

30 March 2015

NSO/0414(2015)C3B/5066

STANAG 5066 C3B (EDITION 3) – PROFILE FOR HF RADIO DATA COMMUNICATIONS

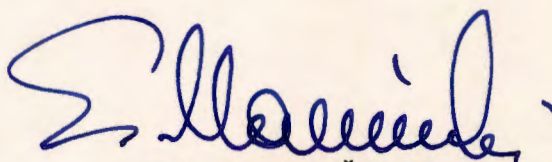
Reference:

AC/322(SC/6)N(2010)0076 dated 8 December 2010

1. The enclosed NATO Standardization Agreement, which has been ratified by nations as reflected in the NATO Standardization Document Database (NSDD), is promulgated herewith.
2. The reference listed above is to be destroyed in accordance with local document destruction procedures.

ACTION BY NATIONAL STAFFS

3. National staffs are requested to examine their ratification status of the STANAG and, if they have not already done so, advise the NATO HQ C3 Staff through their national delegation as appropriate of their intention regarding its ratification and implementation.



Edvardas MAŽEIKIS
Major General, LTUAF
Director, NATO Standardization Office

Enclosure:

STANAG 5066 (Edition 3)

**NORTH ATLANTIC TREATY ORGANIZATION
(NATO)**

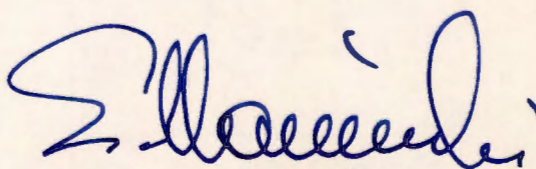


**NATO STANDARDIZATION OFFICE
(NSO)**

**STANDARDIZATION AGREEMENT
(STANAG)**

SUBJECT: PROFILE FOR HF RADIO DATA COMMUNICATIONS

Promulgated on 30 March 2015



Edvardas MAŽEIKIS
Major General, LTUAF
Director, NATO Standardization Office

RECORD OF AMENDMENTS

| No. | Reference/Date of amendment | Date entered | Signature |
|-----|-----------------------------|--------------|-----------|
| | | | |

EXPLANATORY NOTES

AGREEMENT

1. This STANAG is promulgated by the Director NATO Standardization Office under the authority vested in him by the North Atlantic Council.
2. No departure may be made from the agreement without informing the tasking authority in the form of a reservation. Nations may propose changes at any time to the tasking authority where they will be processed in the same manner as the original agreement.
3. Ratifying nations have agreed that national orders, manuals and instructions implementing this STANAG will include a reference to the STANAG number for purposes of identification.

RATIFICATION, IMPLEMENTATION AND RESERVATIONS

4. Ratification, implementation and reservation details are available on request or through the NSO websites (internet <http://nso.nato.int>; NATO Secure WAN <http://nso.hq.nato.int>).

RESTRICTION TO REPRODUCTION

5. No part of this publication may be reproduced, stored in a retrieval system, used commercially, adapted, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior permission of the publisher. With the exception of commercial sales, this does not apply to member and partner nations, or NATO commands and bodies.

FEEDBACK

6. Any comments concerning this publication should be directed to NATO/NSO – Bvd Leopold III - 1110 Brussels - Belgium.

NATO STANDARDIZATION AGREEMENT
(STANAG)

PROFILE FOR HF RADIO DATA COMMUNICATIONS

RELATED DOCUMENTS:

| | |
|------------------|---|
| STANAG 4203 | TECHNICAL STANDARDS FOR SINGLE CHANNEL HF RADIO EQUIPMENT |
| STANAG 4285 | CHARACTERISTICS OF 1200/2400/3600 BITS PER SECOND SINGLE TONE MODULATORS/ DEMODULATORS FOR HF RADIO LINKS |
| STANAG 4529 | CHARACTERISTICS OF SINGLE TONE MODULATORS/ DEMODULATORS FOR HF RADIO LINKS WITH 1240 HZ BANDWIDTH |
| STANAG 4538 | STANAG 4538 TECHNICAL STANDARDS FOR AN AUTOMATIC RADIO CONTROL SYSTEM (ARCS) FOR HF COMMUNICATIONS LINKS |
| STANAG 4539 | TECHNICAL STANDARDS FOR NON-HOPPING HF COMMUNICATIONS WAVEFORMS |
| MIL-STD-188-110B | INTEROPERABILITY AND PERFORMANCE STANDARDS FOR DATA MODEMS |
| CCITT V.41 | CODE INDEPENDENT ERROR CONTROL SYSTEM |
| CCITT V.42 | ERROR-CORRECTING PROCEDURES FOR DCES USING ASYNCHRONOUS-TO-SYNCHRONOUS CONVERSION |

AIM

The aim of this agreement is to define the functions and interfaces required for networked, error-free communication over HF radio channels, nominally for beyond-line-of-sight communications.

AGREEMENT

The participating nations agree to implement the profile defined in this STANAG (including mandatory Annexes) to provide long-haul communications over HF radio circuits.

IMPLEMENTATION OF THE AGREEMENT

This STANAG is implemented by a nation when data communication on long-haul HF radio circuits complies with the characteristics detailed in this agreement.

DEFINITIONS

| | |
|------------|--|
| node | An implementation of the profile described in the main body of and mandatory annexes to this STANAG. The node is generally assumed to include the HF (modem and radio) and cryptographic equipment required for communications. |
| profile | A document describing a set of functions (some or all of which may be defined in separate documents or standards), segregated logically into layers, together with the interfaces, data formats, and procedures required for interoperability. |
| subnetwork | A collection of nodes. As a whole, a subnetwork provides a reliable networked data-transport service for external users or clients. |

1 INTRODUCTION

This document describes a profile for data communication over HF radio, nominally for beyond-line-of-sight circuits. The technical characteristics that are required to ensure interoperability and reliable system operation are described in the main body of and mandatory annexes to the document. Information-only annexes provide information on possible implementation of interfaces and subnetwork clients, and implementation advice based on extensive experience during the development of the protocols.

This document is organised so that the main body gives an overview of the structure of the profile and the capabilities that should be realised when it is implemented. The details of the interfaces, data formats, and procedures are described in a number of mandatory and informational annexes.

The HF profile provides interoperability at the two major interfaces: first, the “common air interface”, describing how information is exchanged between nodes by radio; and second, the non-HF interfaces which allow external users or clients to interact with the subnetwork and with each other over the subnetwork. While physical interfaces are left up to the system implementer (e.g., Ethernet, FDDI, internal bus, or shared memory), the data formats (primitives) and procedures that make up the interface are specified in detail so that client applications can make use of the subnet.

1.1 Common Air Interface: Reliable Data Communications over HF Radio

Reliable data communications over HF radio is provided by using an ARQ data link protocol supported by modern, equalized single-tone HF data modems¹ or other modems using modern modulation and coding techniques.

The data transfer sublayer defined in the profile supports automatic changes of the user data rate (that is, code rate) of the HF modem in response to changing channel conditions (adaptive data rate). This capability requires remote control of the HF modem. The profile is defined so nodes in which remote control of the modem, and hence adaptive data rate, is not available will interoperate with nodes which do have the

¹ Although this document may be used with other HF data modems such as parallel-tone or Orthogonal Frequency Division Multiplexed (OFDM) waveforms, it has been developed and tested for use with the MIL-STD-188-110B, STANAG 4285, STANAG 4529, and STANAG 4539 single-tone waveforms.

capability. The profile includes adaptive data rate with MIL-STD-188-110B, STANAG 4285, STANAG 4529, and STANAG 4539 waveforms.

The HF radio associated with a node is assumed to be an HF SSB radio with specifications appropriate to the modems mentioned above and used within the node. The profile defined here does not assume that any remote control of the radio (specifically, frequency) is available.

1.2 Interoperability With the Subnetwork

The profile also defines the interface between a node and external users of the subnetwork. While physical interfaces are left up to the system implementer (e.g., Ethernet, FDDI, internal bus or shared memory), data formats (primitives) and procedures are defined so that external applications can make use of the subnet. There are two reasons for including these definitions as a mandatory part of the STANAG: first, with a standard interface definition, any vendor can develop an application which makes use of the subnetwork; and second, without such a definition, interoperability is only guaranteed between nodes implemented by the same vendor. Annex F to this STANAG defines the data formats and procedures that will allow interoperation for a limited subset of client applications. Other vendors may define client applications that make use of other procedures; while they should be able to make use of the subnet to communicate with another application of the same type, there is no guarantee of interoperability between vendors. The client application itself is not defined in this document.

For the purposes of clearer discussion, this document divides the functions of the HF profile into a number of sublayers as shown in Figure 1. These sublayers contain the functions which, in terms of the Open Systems Interconnection Reference Model, would be found in the Physical and Link Layers (with a few network layer functions). Figure 1 contains, for completeness, a number of sublayers (shaded) which are not addressed in this document.

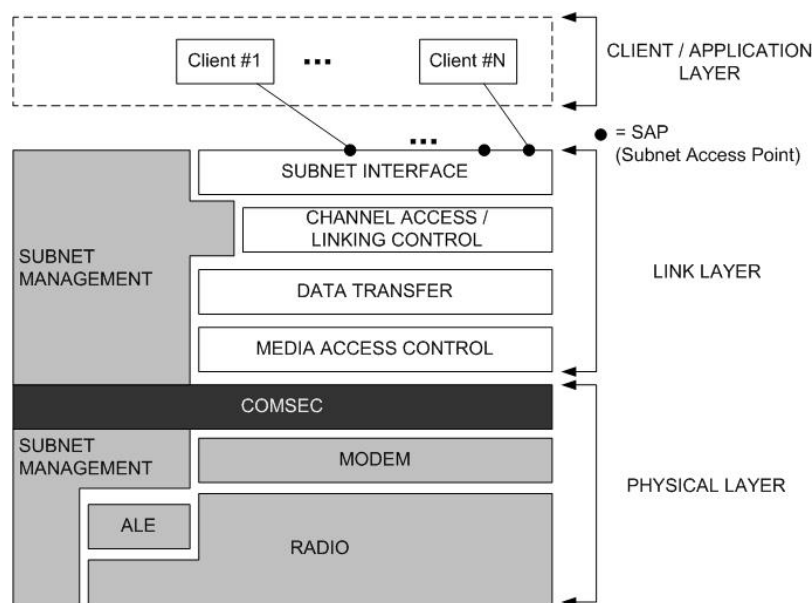


Figure 1. Sublayers within the Profile for HF Data Communication

1.3 Communications between adjacent sublayers and peer sublayers

Communications between adjacent sublayers within a node is done with “primitives”. Primitives at certain sublayer interfaces must be defined to achieve interoperability; the main example is primitives entering the system at the subnet interface. Other primitives that are not required for interoperability are defined here only minimally for information and as an aid to specifying or describing sublayer operation, but are discussed in detail in [1].

Communications between a sublayer and the corresponding sublayer in a different node is done with Protocol Data Units (PDUs) exchanged using the delivery services provided by lower (sub-) layers. For interoperability, PDUs at all sublayers must be defined, together with the protocols for their use. These definitions are given in the annexes to this document.

A brief description of the functions associated with each sublayer follows:

The **Subnetwork Interface Sublayer** provides a common, standard interface to all users. This is the interface between the subnet and the “rest of the world”. Annex A contains a detailed specification of the primitives that are exchanged between this sublayer and external users (clients), and the PDUs that are exchanged (over HF radio) between peer subnetwork interface sublayers.

Annex F provides requirements for the **Client/Application Layer** to ensure that a subset of client applications will be interoperable across vendors. The Client Requirements specify compatibility and interoperability standards for Service-Access-Point Identifiers (SAP ID) that allow simultaneous use of the subnet without interference. Mandatory standards for a Raw-Socket interface are defined over which clients may communicate with the subnet using the S_Primitives of Annex A. Annex F also defines mandatory requirements for an IP-client interface to support IP-over-HF operation. Additional standards are provided for clients — whose implementation is not required for conformance with the STANAG — to ensure their interoperability over the HF subnet; end-to-end interoperability standards for these optional clients are defined for formal and informal messaging as well as for more generalized HF transport services.

The **Channel Access Sublayer** provides additional functionality as needed to allow different forms of channel access in the form of linking control between pairs of nodes; more generalized access is managed by the Media-Access-Control sublayer. For the purposes of this document, this sublayer supports communication over a “dedicated” HF radio channel, under the assumption that the processes required to place the two ends of the link on the proper channel are handled by procedures that are external to this system (mechanisms for handling unintentional interference are provided). The PDUs exchanged (over HF radio) between peer sublayers are defined in Annex B of this document. The primitives exchanged between this sublayer and the Subnetwork Interface sublayers within a node are defined minimally for information purposes and as an aid to specification of the sublayer functions and requirements.

The **Data Transfer Sublayer** provides the various data transfer protocols. These protocols

| provide a reliable (ARQ) data link service, as well as best-effort broadcast services, for regular data-delivery and expedited data-delivery. The PDUs exchanged (over HF radio) between peer sublayers are defined in Annex C of this document. The primitives exchanged between this sublayer and the Channel Access sublayer within a node are defined minimally for information and as an aid to specification of the sublayer requirements. Annex C also contains the protocol for adaptive control of the HF modem data rate and other parameters when using the
| STANAG 4285, STANAG 4529, STANAG 4539, or MIL-STD-188-110B waveforms. The interface between this sublayer and the supporting sublayers below (either communications security sublayer or modem sublayer) is defined in Annex D to this document.

The **Media-Access-Control Sublayer** provides mechanisms for enhanced media-access control capability for HF data communication in multi-node networks. Whereas the Channel-Access Sublayer provides pairwise logical link control mechanisms to establish a point-to-point link (or set of multiple, independent point-to-point links) for data communication, the Media-Access-Control Sublayer introduces optional modes for enhanced media-access control capability for HF data communication in multi-node networks, and the prescribed method in which they may be used with other STANAG 5066 capabilities. These optional channel-access modes extend or modify, but do not replace, the pairwise logical link control mechanisms defined for the Channel Access Sublayer. General requirements for the Media-Access-Control Sublayer are defined in Annex J of this STANAG, with requirements on each of the defined protocols for media-access-control provided in Annexes K, L, and M, for Carrier-Sense-Multiple Access, Wireless-Token-Ring, and Adaptive Time Division Multiple Access².

The **Communications Security Sublayer** provides communications security using hardware crypto equipment. A number of NATO approved cryptos, including BID-950, KG-84C and KIV-7, have been shown to be suitable to provide this function; detailed information may be found in reference [2].

The **Modem Sublayer** provides a means for transmitting digital data over an analogue channel. This STANAG has been developed specifically for use with HF modems defined in STANAG
| 4285 and STANAG 4529 (as well as with STANAG 4539 and MIL-STD-188-110B), though use of other modems that use other waveforms is not precluded. The interface between the modem sublayer and radio equipment is not specified in this document for two reasons: first, it is outside the scope of this document, and second, it is specified in other STANAGs. Current trends in system security indicate that encryption will, in the future, be implemented at or near the application layer. If this change occurs while this STANAG is still in service, the interface between the Data Transfer and modem sublayers shall be as defined in Annex D. This is provided to allow migration toward a more flexible system architecture, in which systems would not be specific to a single vendor's HF modem. Continued requirements for link-level communication security, outside of the scope of this STANAG, may still exist even with application-layer security services.

² N.B.: This STANAG currently retains the title for Annex M on ATDMA as a placeholder grouped with the other MAC-sublayer protocols, if such a protocol is required and produced for future Editions. It is at present an empty Annex.

The **Automatic Link Establishment Sublayer** automates the process of establishing a radio path (link) with one or more remote nodes. This sublayer is not addressed in this document. The system as it is defined is fully compatible with the use of ALE. If in the future ALE is added, no changes to the other sublayers will be required and only minor changes to the implementation will be required (assuming that implementors follow a layered approach in implementing the profile).

The **Radio Equipment Sublayer** comprises the equipment required to establish a radio link between two or more nodes, i.e. transmitters, receivers, transceivers, antennas, etc. This sublayer is not defined in this document. A NATO STANAG (STANAG 4203) exists which specifies minimum standards for transmitters and receivers.

The **Subnet Management Sublayer** is shown in Figure 1 as a vertical column with interfaces to each sublayer. The main subnet management function, in the context of this STANAG, is automatic link maintenance (ALM) in the form of adaptive control of the HF modem. The management sublayer messages and associated procedures which are required for ALM are defined in Annex C of this document, in the context of the MANAGEMENT D_PDU. The other functions of the Subnet Management Sublayer, which may be critically important to a successful implementation, need not be standardized for interoperability and are not addressed further in this document.

Guidance on node address management in STANAG 5066 networks, related to the administration and management of the subnet and address assignments for the Data Transfer Sublayer, is provided in Annex N.

List of Annexes

| | |
|----------|--|
| Annex A: | Subnetwork Interface Sublayer (mandatory) |
| Annex B: | Channel Access Sublayer (mandatory) |
| Annex C: | Data Transfer Sublayer (mandatory) |
| Annex D: | Interface between Data Transfer Sublayer and Communications Equipment (mandatory) |
| Annex E: | HF Modem Remote Control Interface (information only) |
| Annex F: | HF Subnetwork Client Requirements (mandatory) |
| Annex G: | Use of Waveforms at Data Rates Above 2400 bps (information only) |
| Annex H: | Implementation Guide and Notes (information only) |
| Annex I: | Messages and Procedures for Frequency Change (information only) |
| Annex J: | General Requirements for Enhanced Media-Access-Control (MAC) Capabilities in Multi-Node STANAG 5066 Networks (information only) |
| Annex K: | High-Frequency Carrier-Sense Multiple-Access (CSMA) Protocols (information only) |
| Annex L: | High-Frequency Wireless Token-Ring-Protocol (WTRP) Requirements (information only) |
| Annex M: | Adaptive Time-Division Multiple-Access (ATDMA) Protocols using STANAG 5066 DTS Layer Messaging (information only; this annex is currently empty, with the title as a placeholder). |

Annex N: Guidance on Address Management in STANAG 5066 Networks
(information only).

Annexes A through I are unchanged from Edition 2 of this STANAG. This Edition adds
Annexes J through K.

References

1. Clark, D., and N. Karavassillis, "Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio", TM-937, May 1998
2. Miller, T., and P. Reynolds, "Experience with Approved Cryptographic Equipment in HF ARQ Systems", NC3A TN 638, NATO CONFIDENTIAL, November 1996
3. NATO STANDARDIZATION AGREEMENT 4285 Edition 1— Characteristics of 1200/ 2400/ 3600 bps single tone modulators/demodulators for HF radio links
4. NATO STANDARDIZATION AGREEMENT 4529 Edition 1— Characteristics of Single-Tone Modulators/Demodulators for Maritime HF Radio Links with 1240 Hz bandwidth
5. NATO STANDARDIZATION AGREEMENT 4539 Edition 1— Technical Standards for Non-Hopping HF Communications Waveforms
6. DEPARTMENT OF DEFENSE INTERFACE STANDARD 188-110B — Interoperability and Performance Standards for Data Modems, MIL-STD-188-110B, 27 April 2000
7. REQUEST FOR COMMENTS 791 — Postel, J., "Internet Protocol", Internet Engineering Task Force , Network Working Group, RFC 791, September 1981.
8. REQUEST FOR COMMENTS 1661 — W. Simpson, Editor, "The Point-to-Point Protocol (PPP)", Internet Engineering Task Force , Network Working Group, RFC 1661, July 1994
9. REQUEST FOR COMMENTS 2472 — D. Haskin, E. Allen, " IP Version 6 over PPP ", Internet Engineering Task Force , Network Working Group, RFC 2472, December 1998
10. REQUEST FOR COMMENTS 2893 — R. Gilligan, E. Nordmark, "Transition Mechanisms for IPv6 Hosts and Routers ", Internet Engineering Task Force , Network Working Group, RFC 2472, August 2000
11. Mustafa Ergen, Duke Lee et. al., "Wireless Token Ring Protocol", University of California, Berkeley, CA 94720, USA.

Annex A: Subnetwork Interface Sublayer (mandatory)

This annex defines the interface between the users of the HF subnetwork and the computer information system through which the user accesses the subnetwork.

A.1 Subnetwork Service Definition

A client-server relationship governs the interaction between the HF subnetwork and the users of the subnetwork. The users (clients) request the services provided by the HF subnetwork (server). The service provided by the server is application independent and common to all clients irrespective of the task they may perform.

Clients are attached to the Subnetwork Interface Sublayer at Subnetwork Access Points (SAPs). There can be multiple clients simultaneously attached to the Subnetwork Interface Sublayer. Each SAP is identified by its SAP Identifier (SAP ID)¹. The SAP ID is a number in the range 0-15; hence there can be a maximum of 16 clients attached to the Subnetwork Interface Sublayer of a single node.

Annex F contains a recommended definition of the various subnetwork clients. For the purposes of this STANAG Edition, some subnetwork client definitions in Annex F are mandatory. Data submitted by the clients to the Subnetwork Interface Sublayer must be in the form of primitives with the format as described in this document. Clients are responsible for segmenting larger messages into User Protocol Data Units (U_PDUs). A U_PDU format that supports this segmentation is defined in Annex F, but remains outside of the scope of the mandatory requirements on the client-to-subnetwork interface.

The Subnetwork Interface Sublayer treats all clients connected to it in the same manner irrespective of the application performed by these clients. The only distinguishing factor between clients is their **Rank** that is a measure of their importance. See Annex H.5 for further information on the rank of clients. Certain service requests made by higher ranked clients may take precedence over requests made by lower ranked clients.

A.1.1 Initiating Data Exchange Sessions

The Subnetwork Interface Sublayer is responsible for initiating the establishment and termination of Sessions with its peers at remote nodes. There are four types of sessions:

1. Soft Link Data Exchange Session
2. Hard Link Data Exchange Session
3. Broadcast Data Exchange Session
4. Reserved

¹ SAPs are equivalent to the “ports” of the TCP protocol.

All sessions apart from the broadcast data exchange session require the making of a point-to-point physical link with a specified remote node.

Clients for the HF Subnetwork services **may** interleave requests for the various session types in accordance with the capabilities of this standard. Support for only one session type, e.g., restriction to support only a Broadcast Data Exchange Session, **may** be established as part of the local (implementation-dependent) subnetwork management function.

A.1.1.1 Soft Link Data Exchange Session

The establishment of a Soft Link Data Exchange Session **shall** ⁽¹⁾ be initiated unilaterally by the Subnetwork Interface Sublayer which has queued data requiring reliable delivery (i.e., queued ARQ U_PDUs) and from which a client has not requested a Hard Link Data Exchange Session.

The Subnetwork Interface Sublayer **shall** ⁽²⁾ initiate Soft Link Data Exchange Sessions as needed, following the procedure described in Section A.3.2.1.1.

When all data has been transmitted to a node with which a Soft Link Data Exchange Session has been established, the Subnetwork Interface Sublayer **shall** ⁽³⁾ terminate the Soft Link Data Exchange Session after a configurable and implementation-dependent time-out period in accordance with the protocol specified in Section A.3.2.1.2.

Termination of the Soft Link Data Exchange Session **shall** ⁽⁴⁾ be in accordance with the procedure specified in Section A.3.2.1.3. The time out period may be zero. The time out period allows for the possibility of newly arriving U_PDUs being serviced by an existing Soft Link Data Exchange Session prior to its termination.

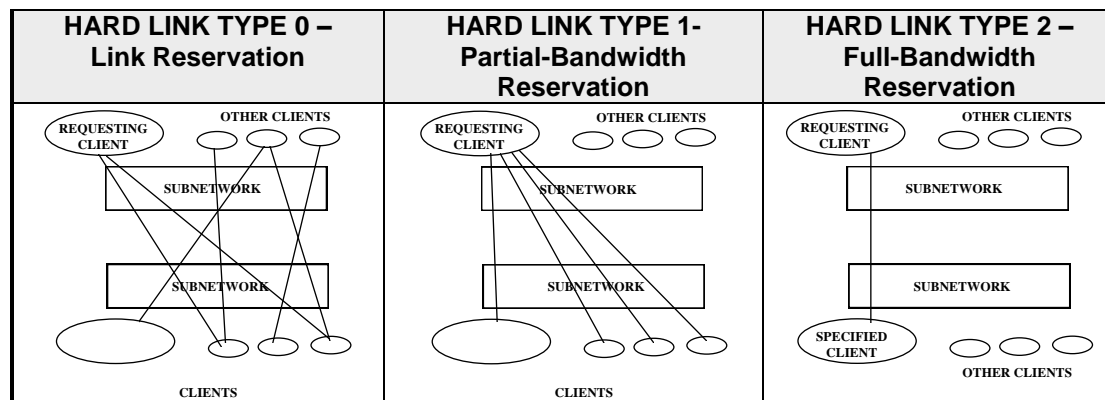
In order to provide “balanced” servicing of the queued U_PDUs, a Soft Link Data Exchange Session **shall** ⁽⁵⁾ not be maintained for a period which exceeds a specified maximum time if U_PDUs of appropriate priorities are queued for different node(s).

The specified maximum time out period **shall** ⁽⁶⁾ be a configurable parameter for the protocol implementation. The specific values of the parameters governing the establishment and termination of Soft Link Data Exchange Sessions (e.g. time-out periods etc.) must be chosen in the context of a particular configuration (i.e. size of network, etc).

A.1.1.2 Hard Link Data Exchange Sessions

The second type of data exchange session is the Hard Link Data Exchange Session. A Hard Link Data Exchange Session **shall** ⁽¹⁾ be initiated at the explicit request of a client in accordance with the procedures for establishing and terminating hard link sessions specified in Sections A.3.2.2.1 and A.3.2.2.2.

A client may request a Hard Link Data Exchange Session in order to ensure that a physical link to a specified node is maintained (irrespective of the destinations of other queued U_PDUs) and optionally to partially or fully reserve the capacity of such a link. The three types of Hard Links that may be established are depicted below in the following figure:



Hard-Link Data-Exchange Session Types

A.1.1.2.1 Type-0 Hard Link: Physical Link Reservation

A Hard Link of Type-0, also called a Hard Link with Link Reservation, **shall** ⁽¹⁾ maintain a physical link between two nodes.

The Type-0 Hard Link capacity **shall** ⁽²⁾ not be reserved for any given client on the two nodes.

Any client on nodes connected by a Hard Link of Type 0 **shall** ⁽³⁾ be permitted to exchange data over the Hard Link.

Any client on either node other than the client that requested the Hard Link **shall** ⁽⁴⁾ gain access to the link only as a Soft-Link Data Exchange Session and may lose the link when the originating client terminates its Hard Link Data Exchange Session.

A.1.1.2.2 Type-1 Hard Link: Partial-Bandwidth Reservation

A Hard Link of Type 1, also called a Hard Link with Partial Bandwidth Reservation, **shall** ⁽¹⁾ maintain a physical link between two nodes.

The Type 1 Hard Link capacity **shall** ⁽²⁾ be reserved only for the client that requested the Type 1 Hard Link between the two nodes. The requesting client may send user data to any client on the remote node, and may receive user data from any client on the remote node only as a Soft-Link Data Exchange Session.

Clients that are not sending data to or receiving data from the client that requested the Type 1 Hard Link **shall** ⁽³⁾ be unable to use the Hard Link. Any client using the link may lose the link when the originating client terminates its Hard Link Session.

A.1.1.2.3 Type-2 Hard Link: Full-Bandwidth Reservation

A Hard Link of Type 2, also called a Hard Link with Full Bandwidth Reservation, **shall** ⁽¹⁾ maintain a physical link between two nodes.

The Type 2 Hard Link capacity **shall** ⁽²⁾ be reserved only for the client that requested the Type 2 Hard Link and a specified remote client. No clients other than the requesting client and its specified remote client **shall** ⁽³⁾ exchange data on a Type-2 Hard Link.

A.1.1.3 Broadcast Data Exchange Session

The third type of data exchange session is the Broadcast Data Exchange Session. The subnetwork **shall** ⁽¹⁾ service only clients with service requirements for non-ARQ U_PDUs during a Broadcast Data Exchange Session. [Note: Clients with service requirements for non-ARQ U_PDUs may be serviced during other session types, however, in accordance with the session's service characteristics.] A Broadcast Data Exchange Session can be initiated and terminated by a management process, e.g., a local or network administrator management client.

The procedures that initiate and terminate broadcast data exchange sessions **shall** ⁽²⁾ be as specified in Annex C.

A node configured to be a broadcast-only node **shall** ⁽³⁾ use a "permanent" Broadcast Data Exchange Session during which the Subnetwork Interface Sublayer **shall** ⁽⁴⁾ service no hard link requests or ARQ Data U_PDUs. Alternatively the Subnetwork Interface Sublayer can unilaterally initiate and terminate Broadcast Data Exchange Sessions.

A.2 Primitives Exchanged with Clients

Communication between the client and the Subnetwork Interface Sublayer uses the interface primitives listed in Table A-1 and defined in the following subsections. The names of these primitives are prefixed with an "S_" to indicate that they are exchanged across the interface between the subnetwork interface sublayer and the subnetwork clients. This table is intended to provide a general guide and overview to the primitives. For detailed specification of the primitives, the later sections of this Annex **shall** ⁽¹⁾ apply.

Table A-1. Primitives Exchanged with Clients

| CLIENT -> SUBNETWORK INTERFACE | SUBNETWORK INTERFACE -> CLIENT |
|--|---|
| S_BIND_REQUEST (Service Type, Rank, SAP ID) | S_BIND_ACCEPTED (SAP ID, MTU) |
| | S_BIND_REJECTED (Reason) |
| S_UNBIND_REQUEST () | S_UNBIND_INDICATION (Reason) |
| S_HARD_LINK_ESTABLISH (Link Priority, Link Type, Remote Node Address, Remote SAP ID) | S_HARD_LINK_ESTABLISHED (Remote Node Status, Link Priority, Link Type, Remote Node Address, Remote SAP ID) |
| | S_HARD_LINK_REJECTED (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID) |
| S_HARD_LINK_ACCEPT (Link Priority, Link Type, Remote Node Address, Remote SAP ID) | S_HARD_LINK_INDICATION (Remote Node Status, Link Priority, Link Type, Remote Node Address, Remote SAP ID) |
| S_HARD_LINK_REJECT (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID) | |
| S_HARD_LINK_TERMINATE (Remote Node Address) | S_HARD_LINK_TERMINATED (Reason, Link Priority, Link Type, Remote Node Address, Remote SAP ID) |
| | |
| | S_SUBNET_AVAILABILITY (Subnet Status, Reason) |
| | |
| S_UNIDATA_REQUEST (Destination Node Address, Destination SAP ID, Priority, TimeToLive, Delivery Mode, U_PDU) | S_UNIDATA_REQUEST_CONFIRM (Destination Node Address, Destination SAP ID, Size of confirmed U_PDU, U_PDU) |
| | S_UNIDATA_REQUEST_REJECTED (Reason, Destination Node Address, Destination SAP ID, Size of Rejected U_PDU, U_PDU) |
| | S_UNIDATA_INDICATION (Source Node Address, Source SAP ID, Destination Node Address, Destination SAP ID, Priority, Transmission Mode, <i>transmission-mode conditional parameters</i> , U_PDU) |
| | |
| S_EXPEDITED_UNIDATA_REQUEST (Destination Node Address, Destination SAP ID, TimeToLive, Delivery Mode, U_PDU) | S_EXPEDITED_UNIDATA_REQUEST_CONFIRM (Destination Node Address, Destination SAP ID, Size of confirmed U_PDU, U_PDU) |
| | S_EXPEDITED_UNIDATA_REQUEST_REJECTED (Reason, Destination Node Address, Destination SAP ID, Size of Rejected U_PDU, U_PDU) |
| | S_EXPEDITED_UNIDATA_INDICATION (Source Node Address, Source SAP ID, Destination Node Address, Destination SAP ID, Transmission Mode, <i>transmission-mode conditional parameters</i> , U_PDU) |
| | |
| | S_DATA_FLOW_ON() |
| | S_DATA_FLOW_OFF () |
| | |
| S_MANAGEMENT_MSG_REQUEST (MSG TYPE, MSG BODY) | S_MANAGEMENT_MSG_INDICATION (MSG TYPE, MSG BODY) |
| | |
| S_KEEP_ALIVE () | S_KEEP_ALIVE () |

A.2.1 Content Specification and Use of Primitives

The content specification and use of the Subnetwork Interface Sublayer primitives **shall** ⁽¹⁾ be as specified in the following subsections.

A.2.1.1 S_BIND_REQUEST Primitive

Name :

S_BIND_REQUEST ()

Arguments :

1. SAP ID,
2. RANK,
3. Service Type

Direction :

Client -> Subnetwork Interface

Description :

The S_BIND_REQUEST primitive **shall** ⁽¹⁾ be issued by a new client when it first connects to the subnetwork. Unless this primitive is issued the client can not be serviced. With this primitive the client uniquely identifies and declares that it is “on-line” and ready to be serviced by the subnetwork.

The first argument of this primitive **shall** ⁽²⁾ be the “SAP ID” which the client wishes to be assigned. The SAP ID **shall** ⁽³⁾ be node-level unique, i.e. not assigned to another client connected to the Subnetwork Interface Sublayer for a given node.

The second argument of this primitive **shall** ⁽⁴⁾ be “Rank”. This is a measure of the importance of a client; the subnetwork uses a client’s rank to allocate resources. A description of the use of the Rank argument may be found in Annex H and [1]. The range of values for the rank argument **shall** ⁽⁵⁾ be from 0 to 15. Clients that are not authorised to make changes to a node or subnetwork configuration **shall** ⁽⁶⁾ not bind with rank of 15.

The last argument of this primitive **shall** ⁽⁷⁾ be “Service Type” and identifies the default type of service requested by the client. The Service Type argument **shall** ⁽⁸⁾ apply to all data units submitted by the client unless explicitly overridden by client request when submitting a U_PDU to the subnetwork. The “Service Type” argument is a complex argument and has a number of attributes that are encoded as specified in Section A.2.2.3.

A.2.1.2 S_UNBIND_REQUEST Primitive

Name :

S_UNBIND_REQUEST ()

Arguments :

NONE

Direction :

Client -> Subnetwork Interface ()

Description :

The S_UNBIND_REQUEST primitive **shall** ⁽¹⁾ be issued by a client in order to declare itself “off-line”. The Subnetwork Interface Sublayer **shall** ⁽²⁾ release the SAP ID allocated to the client

from which it receives the S_UNBIND_REQUEST and the SAP_ID allocated to this client **shall** ⁽³⁾ then be available for allocation to another client that may request it.

A client that went off-line by issuing the S_UNBIND_REQUEST primitive can come on-line again by issuing a new S_BIND_REQUEST.

A client can also go off-line by physically disconnecting itself (e.g. powering down the computer which runs the client program) or disconnecting the physical cable (RS232, Ethernet, etc.) which may connect the client to the node.

The Subnetwork Interface Sublayer can sense whether a client is physically disconnected in order to unilaterally declare this client as off-line; the S_KEEP_ALIVE primitive specified in Section A.2.1.17 provides this capability, though other implementation-dependent methods may be used in addition to this primitive.

[Note: The omission of SAP ID as an argument in this and other primitives implies a requirement on the stack supporting this connection to associate a SAP ID with a lower level connection (i.e., socket) and maintain this association.]

A.2.1.3 S_BIND_ACCEPTED Primitive

Name :

S_BIND_ACCEPTED ()

Arguments :

1. SAP ID
2. Maximum Transmission Unit (MTU)

Direction :

Subnetwork Interface -> Client

Description :

The S_BIND_ACCEPTED primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer as a positive response to a client's S_BIND_REQUEST.

The *SAP ID* argument of the S_BIND_ACCEPTED primitive **shall** ⁽²⁾ be the SAP ID assigned to the client and **shall** ⁽³⁾ be equal to the *SAP ID* argument of the S_BIND_REQUEST to which this primitive is a response.

The *MTU* argument **shall** ⁽⁴⁾ be used by the subnetwork interface sublayer to inform the client of the maximum size U_PDU (in bytes or octets) which will be accepted as an argument of the S_UNIDATA_REQUEST primitive. S_UNIDATA_REQUEST primitives containing U_PDUs larger than the MTU **shall** ⁽⁵⁾ be rejected by the subnetwork interface. Note that this restriction applies only to U_PDUs received through the subnetwork interface. U_PDUs which are received from the lower HF sublayers (i.e., received by radio) **shall** ⁽⁶⁾ be delivered to clients regardless of size.

For general-purpose nodes, the MTU value **shall** ⁽⁷⁾ be 2048 bytes. For broadcast-only nodes, the MTU **shall** ⁽⁸⁾ be configurable by the implementation up to a maximum that **shall** ⁽⁹⁾ not exceed 4096 bytes.

A.2.1.4 S_BIND_REJECTED Primitive

Name :

S_BIND_REJECTED ()

Arguments :

1. Reason

Direction :

Subnetwork Interface -> Client

Description :

The S_BIND_REJECTED primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer as a negative response to a client's S_BIND_REQUEST. If certain conditions are not met then the Subnetwork Interface Sublayer rejects the client's request.

The *Reason* argument of the S_BIND_REJECTED primitive **shall** ⁽²⁾ specify the reason why the client's request was rejected. Valid *Reason* values **shall** ⁽³⁾ be as specified in the table below.

| Reason | Value |
|---|-------|
| Not Enough Resources | 1 |
| Invalid SAP ID | 2 |
| SAP ID already allocated | 3 |
| ARQ Mode unsupportable during Broadcast Session | 4 |

The binary representation of the value in the table **shall** ⁽⁴⁾ be encoded in the Reason field of the primitive by placing the LSB of the value into the LSB of the encoded field for the primitive as specified in Section A.2.2.

A.2.1.5 S_UNBIND_INDICATION Primitive

Name :

S_UNBIND_INDICATION ()

Arguments :

1. Reason

Direction :

Subnetwork Interface->Client

Description :

The S_UNBIND_INDICATION primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer to unilaterally declare a client as off-line. If the client wants to come on-line again, it must issue a new a S_BIND_REQUEST primitive as specified in Section A.2.1.1.

The S_UNBIND_INDICATION primitive provides a means for the Subnetwork Interface Sublayer to manage the clients connected to it. As an implementation dependent example, if a new "High Ranked" client submits a S_BIND_REQUEST to come on-line but not enough resources are available, the Subnetwork Interface Sublayer may unilaterally declare a "Lower

Ranked” client off-line. In such a case, the sublayer will send the lower-ranked client an S_UNBIND_INDICATION in order to release resources for the Higher-Ranked client.

The *Reason* argument of the S_UNBIND_INDICATION primitive **shall** ⁽²⁾ specify why the client was declared off-line. The binary representation of the value in the table **shall** ⁽³⁾ be mapped into the Reason field of the primitive by placing the LSB of the value into the LSB of the encoded field for the primitive as specified in section A.2.2.

| Reason | Value |
|---|-------|
| Connection pre-empted by higher ranked client | 1 |
| Inactivity (failure to respond to “Keep alive”) | 2 |
| Too many invalid primitives | 3 |
| Too many expedited data request primitives | 4 |
| ARQ Mode Unsupportable during Broadcast Session | 5 |

A.2.1.6 S_UNIDATA_REQUEST Primitive

Name :

S_UNIDATA_REQUEST()

Arguments :

1. Priority
2. Destination SAP ID
3. Destination Node Address
4. Delivery Mode
5. TimeToLive (TTL)
6. Size of U_PDU
7. U_PDU (User Protocol Data Unit)

Direction :

Client->Subnet Interface

Description :

The S_UNIDATA_REQUEST primitive **shall** ⁽¹⁾ be used by connected clients to submit a U_PDU to the HF subnetwork for delivery to a receiving client.

The argument *Priority* **shall** ⁽²⁾ represent the priority of the U_PDU. The U_PDU priority **shall** ⁽⁵⁾ take a value in the range 0-15. The processing by HF protocol sublayers **shall** ⁽⁶⁾ make a “best effort” to give precedence to high priority U_PDUs over lower priority U_PDUs which are queued in the system.

The argument *Destination SAP ID* **shall** ⁽³⁾ specify the SAP ID of the receiving client. Note that as all nodes will have uniquely specified SAP IDs for clients, the Destination SAP ID distinguishes the destination client from the other clients bound to the destination node.

The argument *Destination Node Address* **shall** ⁽⁴⁾ specify the HF subnetwork address of the physical HF node to which the receiving client is bound.

The argument *Delivery Mode* **shall** ⁽⁵⁾ be a complex argument with a number of attributes, as specified by the encoding rules of Section A.2.2.28.2. This argument can be given the value of "DEFAULT" which means that the delivery mode associated with the U_PDU will be the delivery mode specified by the client during "binding" (i.e., the value DEFAULT is equal to the *Service Type* argument of client's original S_BIND_REQUEST). Values other than DEFAULT for the *Delivery Mode* can be used to override the default delivery mode for this U_PDU.

The argument *TimeToLive (TTL)* **shall** ⁽⁶⁾ specify the maximum amount of time the submitted U_PDU is allowed to stay in the HF Subnetwork before it is delivered to its destination. If the TTL is exceeded the U_PDU **shall** ⁽⁷⁾ be discarded. A TTL value of 0 **shall** ⁽⁸⁾ define an infinite TTL, i.e. the subnetwork should try *forever* to deliver the U_PDU.

The subnetwork **shall** ⁽⁹⁾ have a default maximum TTL. The default maximum TTL **shall** ⁽¹⁰⁾ be configurable as an implementation-dependent value. As soon as the Subnetwork Interface Sublayer accepts a S_UNIDATA_REQUEST primitive, it **shall** ⁽¹¹⁾ immediately calculate its *TimeToDie (TTD)* by adding the specified TTL (or the default maximum value if the specified TTL is equal to 0) to the current Time of Day, e.g. GMT. The TTD attribute of a U_PDU **shall** ⁽¹²⁾ accompany it during its transit within the subnetwork. [Note that the TTD is an absolute time while the TTL is a time interval relative to the instant of the U_PDU submission.]

The *Size of U_PDU* argument **shall** ⁽¹³⁾ be the size of the U_PDU that is included in this S_UNIDATA_REQUEST Primitive.

The final argument, *U_PDU*, **shall** ⁽¹⁴⁾ be the actual Data Unit submitted by the client to the HF Subnetwork.

A.2.1.7 S_UNIDATA_REQUEST_CONFIRM Primitive

Name :

S_UNIDATA_REQUEST_CONFIRM

Arguments :

1. Destination Node Address
2. Destination SAP ID
3. Size of Confirmed U_PDU
4. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->Client

Description :

The S_UNIDATA_REQUEST_CONFIRM primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer to acknowledge the successful delivery of a S_UNIDATA_REQUEST submitted by the client.

This primitive **shall** ⁽²⁾ be issued only if the client has requested Data Delivery Confirmation (either during binding or for this particular data unit).

The *Destination Node Address* argument in the S_UNIDATA_REQUEST_CONFIRM Primitive **shall** ⁽³⁾ have the same meaning and be equal in value to the *Destination Node Address* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_CONFIRM Primitive is the response.

The *Destination SAP_ID* argument in the S_UNIDATA_REQUEST_CONFIRM Primitive **shall** ⁽⁴⁾ have the same meaning and be equal in value to the *Destination SAP_ID* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_CONFIRM Primitive is the response.

The *Size of Confirmed U_PDU* argument **shall** ⁽⁵⁾ be the size of the U_PDU or part that is included in this S_UNIDATA_REQUEST_CONFIRM Primitive.

The *U_PDU* argument in the S_UNIDATA_REQUEST_CONFIRM Primitive **shall** ⁽⁶⁾ be a copy of the whole or a fragment of the *U_PDU* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_CONFIRM Primitive is the response.

Using these arguments, the client **shall** ⁽⁷⁾ be able to uniquely identify the U_PDU that is being acknowledged. Depending on the implementation of the protocol, the last argument, *U_PDU*, may not be a complete copy of the original U_PDU but only a partial copy, i.e., only the first X bytes are copied for some value of X. If a partial U_PDU is returned, *U_PDU_response_frag_size* bytes **shall** ⁽⁹⁾ be returned to the client starting with the first byte of the U_PDU so that the client will have the U_PDU segment information. The number of bytes returned, *U_PDU_response_frag_size*, **shall** ⁽¹⁰⁾ be a configurable parameter in the implementation..

A.2.1.8 S_UNIDATA_REQUEST_REJECTED Primitive

Name :

S_UNIDATA_REQUEST_REJECTED

Arguments :

1. Reason
2. Destination Node Address
3. Destination SAP ID
4. Size of Rejected U_PDU (or part)
5. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->Client

Description :

The S_UNIDATA_REQUEST_REJECTED primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer to inform a client that a S_UNIDATA_REQUEST was not delivered successfully.

This primitive **shall** ⁽²⁾ be issued if the client has requested Data Delivery Confirmation (either during Binding or for this particular U_PDU) and the data was unsuccessfully delivered. This primitive also **shall** ⁽³⁾ be issued to a client if a U_PDU larger than the MTU is submitted.

The argument *Reason* **shall** ⁽⁴⁾ specify why the delivery failed, using the encoding given in the table below:

| Reason | Value |
|---------------------------------|-------|
| TTL Expired | 1 |
| Destination SAP ID not bound | 2 |
| Destination node not responding | 3 |
| U_PDU larger than MTU | 4 |
| Tx Mode not specified | 5 |

The binary representation of the value in the table **shall** ⁽⁵⁾ be mapped into the Reason argument of the primitive by placing the LSB of the value into the LSB of the encoded argument for the primitive as specified in section A.2.2

The *Destination Node Address* argument in the S_UNIDATA_REQUEST_REJECTED Primitive **shall** ⁽⁶⁾ have the same meaning and be equal in value to the *Destination Node Address* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Destination SAP_ID* argument in the S_UNIDATA_REQUEST_REJECTED Primitive **shall** ⁽⁷⁾ have the same meaning and be equal in value to the *Destination SAP_ID* argument of the S_UNIDATA_REQUEST Primitive for which the S_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Size of Rejected U_PDU* argument **shall** ⁽⁸⁾ be the size of the U_PDU or part that is included in this S_UNIDATA_REQUEST_REJECTED Primitive.

Just as specified for the S_UNIDATA_REQUEST_CONFIRM primitive, the *U_PDU* argument in the S_UNIDATA_REQUEST_REJECTED primitive may only be a partial copy of the original U_PDU, depending on the implementation of the protocol. If a partial U_PDU is returned, *U_PDU_response_frag_size* bytes **shall** ⁽⁹⁾ be returned to the client starting with the first byte of the U_PDU so that the client will have the U_PDU segment information. The number of bytes returned, *U_PDU_response_frag_size*, **shall** ⁽¹⁰⁾ be a configurable parameter in the implementation.

A.2.1.9 S_UNIDATA_INDICATION Primitive

Name :

S_UNIDATA_INDICATION

Arguments :

1. Priority
2. Destination SAP ID
3. Destination Node Address
4. Transmission Mode
5. Source SAP ID
6. Source Node Address
7. Size of U_PDU
8. Number of Blocks in Error
9. Array of Block-Error Pointers
10. Number of Non-Received Blocks
11. Array of Non-Received-Block Pointers
12. U_PDU

Direction :

Subnetwork Interface->client

Description :

The S_UNIDATA_INDICATION primitive **shall** ⁽¹⁾ be used by the Subnetwork Interface Sublayer to deliver a received U_PDU to the client.

The *Priority* argument **shall** ⁽²⁾ be the priority of the PDU.

The *Destination SAP ID* argument **shall** ⁽³⁾ be the SAP ID of the client to which this primitive is delivered.

The *Destination Node Address* argument **shall** ⁽⁴⁾ be the address assigned by the sending node to the U_PDU contained within this primitive. This normally will be the address of the local (i.e., receiving) node. It may however be a “group” address to which the local node has subscribed (Group Addresses and their subscribers are defined during configuration) and to which the source node addressed the U_PDU.

The *Transmission Mode* argument **shall** ⁽⁵⁾ be the mode by which the U_PDU was transmitted by the remote node and received by the local node; ie, ARQ, Non-ARQ (Broadcast) transmission, Non-ARQ w/ Errors, etc., encoded as per section A.2.2.28.3

The *Source SAP ID* **shall** ⁽⁶⁾ be SAP ID of the client that sent the U_PDU.

The *Source Node Address* **shall** ⁽⁷⁾ represent the node address of the client that sent the U_PDU.

The *Size of U_PDU* argument **shall** ⁽⁸⁾ be the size of the U_PDU that was sent and delivered in this S_UNIDATA_INDICATION S_Primitive.

The following four arguments **shall** ⁽⁹⁾ be present in the S_UNIDATA_INDICATION S_Primitive if and only if the Transmission Mode for the U_PDU is equal to Non-ARQ w/ Errors:

- a) The *Number of Blocks in Error* argument **shall** ⁽¹⁰⁾ equal the number of data blocks in the U_PDU that were received in error by the lower layers of the subnetwork and that were passed on to the Subnetwork Interface Sublayer. This argument **shall** ⁽¹¹⁾ specify the number of ordered pairs in the *Array of Block-Error Pointers* argument.
- b) The *Array of Block-Error Pointers* argument **shall** ⁽¹²⁾ consist of a an array of ordered pairs, the first element in the pair equal to the location within the U_PDU of the data block with errors, and the second element equal to the size of the data block with errors.
- c) The *Number of Non-Received Blocks* argument **shall** ⁽¹³⁾ equal the number of data blocks missing from the U_PDU because they were not received. This argument **shall** ⁽¹⁴⁾ specify the number of ordered pairs in the *Array of Non-Received-Block Pointers* argument.
- d) The *Array of Non-Received-Block Pointers* **shall** ⁽¹⁵⁾ consist of an array of ordered pairs, the first element in the pair equal to the location of the missing data block in the U_PDU and the second element equal to the size of the missing data block.

The final argument, *U_PDU*, **shall** ⁽¹⁶⁾ contain the actual received user data for delivery to the client.

A.2.1.10 S_EXPEDITED_UNIDATA_REQUEST Primitive

Name :

S_EXPEDITED_UNIDATA_REQUEST

Arguments :

1. Destination SAP ID
2. Destination Node Address
3. Delivery Mode
4. TimeToLive (TTL)
5. Size of U_PDU
6. U_PDU (User Protocol Data Unit)

Direction :

Client->Subnet Interface

Description :

The S_EXPEDITED_UNIDATA_REQUEST primitive **shall** ⁽¹⁾ be used to submit a U_PDU to the HF Subnetwork for Expedited Delivery to a receiving client.

The argument *Destination SAP ID* **shall** ⁽²⁾ specify the SAP ID of the receiving client. Note that as all nodes will have uniquely specified SAP IDs for clients, the Destination SAP ID distinguishes the destination client from the other clients bound to the destination node.

The argument *Destination Node Address* **shall** ⁽³⁾ specify the HF subnetwork address of the physical HF node to which the receiving client is bound.

The argument *Delivery Mode* **shall** ⁽⁴⁾ be a complex argument with a number of attributes, as specified by the encoding rules of Section A.2.2.28.2. This argument can be given the value of “Default” which means that the delivery mode associated with the U_PDU will be the delivery mode specified by the client during “binding” (*Service Type* argument of S_BIND_REQUEST). The other values of the *Delivery Mode* can be used to override the default delivery mode for this U_PDU.

The argument *TimeToLive (TTL)* **shall** ⁽⁵⁾ specify the maximum amount of time the submitted U_PDU is allowed to stay in the HF Subnetwork before it is delivered to its final destination. If the TTL is exceeded the U_PDU **shall** ⁽⁶⁾ be discarded. A TTL value of 0 **shall** ⁽⁷⁾ define an infinite TTL, i.e. the subnetwork should try *forever* to deliver the U_PDU.

As soon as the Subnetwork Interface Sublayer accepts a S_EXPEDITED_UNIDATA_REQUEST primitive, it **shall** ⁽⁸⁾ immediately calculate its *TimeToDie (TTD)* by adding the specified TTL (or the default maximum TTL value if the specified TTL is equal to 0) to the current Time of Day, e.g. GMT. The TTD attribute of a U_PDU **shall** ⁽⁹⁾ accompany it during its transit within the subnetwork. [Note that the TTD is an absolute time while the TTL is a time interval relative to the instant of the U_PDU submission.]

The *Size of U_PDU* argument **shall** ⁽¹⁰⁾ be the size of the U_PDU that is included in this S_UNIDATA_REQUEST Primitive.

The final argument, *U_PDU*, **shall** ⁽¹¹⁾ be the actual User Data Unit (U_PDU) submitted by the client to the HF Subnetwork for expedited delivery service.

[Note: There is no *Priority* argument in the S_EXPEDITED_UNIDATA_REQUEST primitive. Although seemingly equivalent, there are a important differences between a S_UNIDATA_REQUEST primitive of the highest priority and a S_EXPEDITED_UNIDATA_REQUEST primitive. S_UNIDATA_REQUEST primitives of all priority levels are processed according to a set of rules that apply to Normal Data. U_PDUs submitted using S_EXPEDITED_UNIDATA_REQUEST primitives are treated differently, e.g. expedited U_PDUs should be queued separately from normal U_PDUs. When an expedited U_PDU is received, the transmission of normal data is halted and the expedited data is transmitted. When the expedited data has been sent the transmission of normal data is resumed again.]

The 5066 node management **shall** ⁽³⁾ track the number of S_EXPEDITED_UNIDATA_REQUEST primitives submitted by various clients. If the number of S_EXPEDITED_UNIDATA_REQUEST primitives for any client exceeds a configurable, implementation dependent parameter, node management **shall** ⁽⁴⁾ unilaterally disconnect the client using a S_UNBIND_INDICATION primitive with REASON = 4 = “Too many expedited-data request primitives”.

A.2.1.11 S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive

Name :

S_EXPEDITED_UNIDATA_REQUEST_CONFIRM

Arguments :

1. Destination Node Address
2. Destination SAP ID

3. Size of Confirmed U_PDU (or part)
4. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->Client

Description :

The S_EXPEDITED_UNIDATA_REQUEST_CONFIRM primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer to acknowledge the successful delivery of a S_EXPEDITED_UNIDATA_REQUEST primitive.

This primitive **shall** ⁽²⁾ be issued only if the client has requested Data Delivery Confirmation (either during Binding or for this particular U_PDU).

The *Destination Node Address* argument in the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive **shall** ⁽³⁾ have the same meaning and be equal in value to the *Destination Node Address* argument of the S_EXPEDITED_UNIDATA_REQUEST Primitive for which the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive is the response.

The *Destination SAP_ID* argument in the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive **shall** ⁽⁴⁾ have the same meaning and be equal in value to the *Destination SAP_ID* argument of the S_EXPEDITED_UNIDATA_REQUEST Primitive for which the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive is the response.

The *Size of Confirmed U_PDU* argument **shall** ⁽⁵⁾ be the size of the U_PDU or part that is included in the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive.

Just as specified for the S_UNIDATA_REQUEST_CONFIRM primitive, the *U_PDU* argument in the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM primitive may only be a partial copy of the original U_PDU, depending on the implementation of the protocol. If a partial U_PDU is returned, *U_PDU_response_frag_size* bytes **shall** ⁽⁶⁾ be returned to the client starting with the first byte of the U_PDU so that the client will have the U_PDU segment information. The number of bytes returned, *U_PDU_response_frag_size*, **shall** ⁽⁷⁾ be a configurable parameter in the implementation.

A.2.1.12 S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive

Name :

S_EXPEDITED_UNIDATA_REQUEST_REJECTED

Arguments :

1. Reason
2. Destination Node Address
3. Destination SAP ID
4. Size of Rejected U_PDU (or part)
5. U_PDU (User Protocol Data Unit or part of it)

Direction :

Subnetwork Interface->Client

Description :

The S_EXPEDITED_UNIDATA_REQUEST_REJECTED primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer to inform a client that a S_EXPEDITED_UNIDATA_REQUEST was not delivered successfully.

This primitive **shall** ⁽²⁾ be issued if the client has requested Data Delivery Confirmation (either during Binding or for this particular U_PDU), or if a U_PDU larger than the MTU is submitted.

The argument *Reason* **shall** ⁽³⁾ specify why the delivery failed with values defined for this field as specified in the table below.

| Reason | Value |
|---------------------------------|-------|
| TTL Expired | 1 |
| Destination SAP ID not bound | 2 |
| Destination node not responding | 3 |
| U_PDU larger than MTU | 4 |
| I Tx Mode not specified | 5 |

The binary representation of the value in the table **shall** ⁽⁴⁾ be mapped into the Reason field of the primitive by placing the LSB of the value into the LSB of the encoded field for the primitive as specified in section A.2.2.1.

The *Destination Node Address* argument in the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive **shall** ⁽⁵⁾ have the same meaning and be equal in value to the *Destination Node Address* argument of the S_EXPEDITED_UNIDATA_REQUEST Primitive for which the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Destination SAP_ID* argument in the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive **shall** ⁽⁶⁾ have the same meaning and be equal in value to the *Destination SAP_ID* argument of the S_EXPEDITED_UNIDATA_REQUEST Primitive for which the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive is the response.

The *Size of Rejected U_PDU* argument **shall** ⁽⁷⁾ be the size of the U_PDU or part that is included in the S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive.

Just as specified for the S_EXPEDITED_UNIDATA_REQUEST_CONFIRM primitive, the *U_PDU* argument in the S_EXPEDITED_UNIDATA_REQUEST_REJECTED primitive may only be a partial copy of the original U_PDU, depending on the implementation of the protocol. If a partial U_PDU is returned, *U_PDU_response_frag_size* bytes **shall** ⁽⁸⁾ be returned to the client starting with the first byte of the U_PDU so that the client will have the U_PDU segment information. The number of bytes returned, *U_PDU_response_frag_size*, **shall** ⁽⁹⁾ be a configurable parameter in the implementation.

A.2.1.13 S_EXPEDITED_UNIDATA_INDICATION Primitive

Name :

S_EXPEDITED_UNIDATA_INDICATION

Arguments :

1. Destination SAP ID
2. Destination Node Address
3. Transmission Mode
4. Source SAP ID
5. Source Node Address
6. Size of U_PDU
7. Number of Blocks in Error
8. Array of Block-Error Pointers
9. Number of Non-Received Blocks
10. Array of Non-Received-Block Pointers
11. U_PDU

Direction :

Subnetwork Interface->Client

Description :

The S_EXPEDITED_UNIDATA_INDICATION primitive **shall** ⁽¹⁾ be used by the Subnetwork Interface Sublayer to deliver an Expedited U_PDU to a client.

The *Destination SAP ID* argument **shall** ⁽²⁾ be the SAP ID of the client to which this primitive is delivered.

The *Destination Node Address* argument **shall** ⁽³⁾ be the address assigned by the sending node to the U_PDU contained within this primitive. This normally will be the address of the local (i.e., receiving) node. It may however be a “group” address to which the local node has subscribed (Group Addresses and their subscribers are defined during configuration) and to which the source node addressed the U_PDU.

The *Transmission Mode* argument **shall** ⁽⁴⁾ be the mode by which the U_PDU was transmitted by the remote node and received by the local node; ie, ARQ, Non-ARQ (Broadcast) transmission, Non-ARQ w/ Errors, etc.

The *Source SAP ID* **shall** ⁽⁵⁾ be SAP ID of the client that sent the U_PDU.

The *Source Node Address* **shall** ⁽⁶⁾ represent the node address of the client that sent the U_PDU.

The *Size of U_PDU* argument **shall** ⁽⁷⁾ be the size of the U_PDU that was sent and delivered in this S_EXPEDITED_UNIDATA_INDICATION S_Primitive.

The following four arguments **shall** ⁽⁸⁾ be present in the S_EXPEDITED_UNIDATA_INDICATION S_Primitive if and only if the Transmission Mode for the U_PDU is equal to Non-ARQ w/ Errors:

a) The *Number of Blocks in Error* argument **shall** ⁽⁹⁾ equal the number of data blocks in the U_PDU that were received in error by the lower layers of the subnetwork and that were passed on to the Subnetwork Interface Sublayer. This argument **shall** ⁽¹⁰⁾ specify the number of ordered pairs in the *Array of Block-Error Pointers* argument.

b) The *Array of Block-Error Pointers* argument **shall** ⁽¹¹⁾ consist of an array of ordered pairs, the first element in the pair equal to the location within the U_PDU of the data block with errors, and the second element equal to the size of the data block with errors.

c) The *Number of Non-Received Blocks* argument **shall** ⁽¹²⁾ equal the number of data blocks missing from the U_PDU because they were not received. This argument **shall** ⁽¹³⁾ specify the number of ordered pairs in the *Array of Non-Received-Block Pointers* argument.

d) The *Array of Non-Received-Block Pointers* **shall** ⁽¹⁴⁾ consist of an array of ordered pairs, the first element in the pair equal to the location of the missing data block in the U_PDU and the second element equal to the size of the missing data block.

The final argument, *U_PDU*, **shall** ⁽¹⁵⁾ contain the actual received user data for delivery to the client.

A.2.1.14 Interface Flow Control Primitives: S_DATA_FLOW_ON and S_DATA_FLOW_OFF

Name :

S_DATA_FLOW_ON
S_DATA_FLOW_OFF

Arguments :

NONE

Direction :

Subnetwork Interface-> Client

Description :

The S_DATA_FLOW_ON and S_DATA_FLOW_OFF primitives **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer to control the transfer of U_PDUs submitted by a client.

On receipt of an _DATA_FLOW_OFF primitive, the client **shall** ⁽²⁾ cease transferring U_PDUs over the interface.

Transfer over the interface of U_PDUs by the client **shall** ⁽³⁾ be enabled following receipt of an S_DATA_FLOW_ON primitive.

Depending on the implementation, the physical connection between the client(s) and the Subnetwork Interface Sublayer may provide an implicit flow-control mechanism that would make the use of these primitives unnecessary. For example, if the connection is implemented as TCP/IP Berkeley Sockets, the implicit flow-control mechanism of the TCP protocol may be utilized in which case these two primitives are redundant.

The Subnetwork Interface Sublayer can use these two primitives (or other mechanisms) to control the flow of data from locally attached clients. U_PDUs from an attached client to which the S_DATA_FLOW_OFF primitive has been sent may be discarded by the Subnetwork Interface Sublayer without acknowledgement, indication, or warning.

A client **shall** ⁽⁴⁾ not control the flow of data *from* the subnetwork by any mechanism, explicit or implicit.

All clients **shall** ⁽⁵⁾ be ready to accept at all times data received by the HF Node to which it is bound; clients not following this rule may be disconnected by the node.

A.2.1.15 S_MANAGEMENT_MSG_REQUEST Primitive

Name :

S_MANAGEMENT_MSG_REQUEST

Arguments :

1. MSG TYPE
2. MSG BODY

Direction :

Client-> Subnet Interface

Description :

The S_MANAGEMENT_MSG_REQUEST primitive **shall** ⁽¹⁾ be issued by a client to submit a "Management" message to the Subnetwork.

The complex argument MSG may be implementation dependent and is not specified in this version of STANAG 5066. At present, a minimally compliant HF subnetwork implementation **shall** ⁽²⁾ be capable of receiving this primitive, without further requirement to process its contents.

The subnetwork **shall** ⁽³⁾ accept this primitive only from clients which have bound with a rank of 15.

Depending on the value of the complex argument MSG, this primitive can take the form of a Command (e.g. Go-To-EMCON, Go-Off-Air, etc.) or of a Request (e.g. Request-For-Subnetwork-Statistics, Request-For-Connected-client-Information, etc.).

Note that this primitive is not intended to allow for the transmission of management coordination messages over the air. This is an interaction between peer subnet management clients and as such shall be accomplished using the UNIDATA or EXPEDITED UNIDATA primitives defined elsewhere in this annex.

A.2.1.16 S_MANAGEMENT_MSG_INDICATION Primitive

Name :

S_MANAGEMENT_MSG_INDICATION

Arguments :

1. MSG TYPE
2. MSG BODY

Direction :

Subnetwork Interface-> Client

Description :

The S_MANAGEMENT_MSG_INDICATION primitive **shall** ⁽¹⁾ be issued by the Subnetwork to send a "Management" message to a client.

The complex argument MSG may be implementation dependent and is not specified in this version of STANAG 5066. At present, a minimally compliant client **shall** ⁽²⁾ be capable of receiving this primitive, without further requirement to process its contents.

As implementation options, the complex argument *MSG* could take several values such as: Subnetwork-Statistics, Connected-client-Information, etc. This primitive could be issued either in response to a S_MANAGEMENT_MSG_REQUEST or asynchronously by the Subnetwork.

A.2.1.17 S_KEEP_ALIVE Primitive

Name :

S_KEEP_ALIVE

Arguments :

NONE

Direction :

Client-> Subnetwork Interface

Subnetwork Interface-> Client

Description :

The S_KEEP_ALIVE primitive can be issued as required (e.g. during periods of inactivity) by the clients and/or the Subnetwork Interface to sense whether the physical connection between the client and the Subnetwork is alive or broken. This primitive may be redundant if the implementation of the physical connection provides an implicit mechanism for sensing the status of the connection.

A minimally compliant implementation of a client or subnetwork interface is not required to generate the S_KEEP_ALIVE primitive except in response to the receipt of an S_KEEP_ALIVE primitive.

When the S_KEEP_ALIVE Primitive is received, the recipient (i.e, client or Subnetwork Interface) **shall** ⁽¹⁾ respond with the same primitive within 10 seconds.

If a reply is not sent within 10 seconds, no reply **shall** ⁽²⁾ be sent.

A client or Subnetwork Interface **shall** ⁽³⁾ not send the S_KEEP_ALIVE Primitive more frequently than once every 120 seconds to the same destination.

A.2.1.18 S_HARD_LINK_ESTABLISH Primitive

Name :

S_HARD_LINK_ESTABLISH

Arguments :

1. Link Priority
2. Link Type
3. Remote Node Address
4. Remote SAP ID

Direction :

Client-> Subnetwork Interface

Description :

The S_HARD_LINK_ESTABLISH primitive **shall** ⁽¹⁾ be used by a client to request the establishment of a Hard Link between the local Node to which it is connected and a specified remote Node.

[Note: Physical Links between Nodes are normally made and broken unilaterally by the HF subnetwork according to the destinations of the queued U_PDUs. Such links are classified as Soft Links. The S_HARD_LINK_ESTABLISH primitive allows a client to override these procedures and request a Physical Link to be made to a specific Node and be maintained until the requesting client decides to break it.]

The argument *Link Priority* **shall** ⁽²⁾ define the priority of the Link. It **shall** ⁽³⁾ take a value in the range 0-3.

An S_HARD_LINK_ESTABLISH primitive with a higher Link Priority value **shall** ⁽⁴⁾ take precedence over a Hard Link established with a lower Link Priority value submitted by a client of the same Rank.

Hard Link requests made by clients with higher Rank **shall** ⁽⁵⁾ take precedence over requests of lower-Ranked clients regardless of the value of the *Link Priority* argument, in accordance with the requirements of Section A.3.2.2.1.

The *Link Type* argument **shall** ⁽⁶⁾ be used by the requesting client to fully or partially reserve the bandwidth of the Link. It **shall** ⁽⁷⁾ take a value in the range 0-2, as specified in Section A.1.1.2, specifying this primitive as one for a Type 0 Hard Link, Type 1 Hard Link, or Type 2 Hard Link, respectively.

The *Remote Node Address* argument **shall** ⁽⁸⁾ specify the physical HF Node Address to which the connection must be established and maintained.

The *Remote SAP ID* argument **shall** ⁽⁹⁾ identify the single client connected to the remote Node, to and from which traffic is allowed. This argument **shall** ⁽¹⁰⁾ be valid only if the *Link Type* argument has a value of 2 (i.e., only if the Hard Link request reserves the full bandwidth of the link for the local and remote client, as specified in section A.1.1.2.3).

A.2.1.19 S_HARD_LINK_TERMINATE Primitive

Name :

S_HARD_LINK_TERMINATE

Arguments :

- | 1. Remote Node Address

Direction :

Client-> Subnetwork Interface

Description :

The S_HARD_LINK_TERMINATE primitive **shall** ⁽¹⁾ be issued by a client to terminate an existing Hard Link.

The subnetwork **shall** ⁽²⁾ terminate an existing Hard Link on receipt of this primitive only if the primitive was generated by the client which requested the establishment of the Hard Link.

The single argument *Remote Node Address* **shall** ⁽³⁾ specify the Address of the Node at the remote end of the Hard Link.

[Note: The *Remote Node Address* argument is redundant in that Hard Links can exist with only one remote node at any time. It may however be used by the subnetwork implementation receiving the primitive to check its validity.]

Upon receiving this primitive, the subnetwork **shall** ⁽⁴⁾ take all necessary steps to terminate the Hard Link, as specified in section A.3.2.2.3².

[Note: The HARD LINK TERMINATE primitive is always accepted, and the subnetwork will terminate the link whether or not the remote node responds to the termination protocol. As specified in section A.3.2.2.3, the subnetwork will issue a S_HARD_LINK_TERMINATED primitive confirming the successful termination of the Link only if the termination protocol ends without confirmation from the remote node and the subnetwork was required to terminate the Hard Link unilaterally.]

A.2.1.20 S_HARD_LINK_ESTABLISHED Primitive

Name :

S_HARD_LINK_ESTABLISHED

Arguments :

- 1. Remote Node Status
- 2. Link Priority
- 3. Link Type
- 4. Remote Node Address
- 5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

² The Link can be terminated immediately or in a “graceful” manner according to the requirements of a specific application and implementation. A graceful termination might, for example, allow completion of the current transmission interval before the link is broken and/or allow transmission of queued high priority U_PDUs from other clients to the same destination to be transmitted before the link is terminated.

Description:

The S_HARD_LINK_ESTABLISHED primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer as a positive response to a client's S_HARD_LINK_ESTABLISH primitive.

This primitive **shall** ⁽²⁾ be issued only after all the negotiations and protocols between the appropriate peer sublayers of the local and remote nodes have been completed and the remote node has accepted the establishment of the Hard Link, in accordance with the protocol specified in Section A.3.2.2.2.

The first argument, *Remote Node Status*, **shall** ⁽³⁾ inform the requesting client of any special status of the remote node, e.g. Remote Node in EMCON, etc. Valid arguments for *Remote Node Status* are given in the table below.

| Remote Node Status | Value |
|--------------------|-------|
| ERROR | 0 |
| OK | >=1 |

Subsequent versions of this STANAG and implementation-dependent options may define additional values for the remote node status, for example, through use of the same set of local-node status codes defined for the S_SUBNET_AVAILABILITY primitive to report the status of a remote node. Successful establishment of a Hard Link **shall** ⁽⁴⁾ always imply a status of "OK" for the remote node; the value OK **shall** ⁽⁵⁾ be indicated by any positive non-zero value in the Remote Node Status field.

The argument *Link Priority* **shall** ⁽⁶⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

The *Link Type* argument **shall** ⁽⁷⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

The *Remote Node Address* argument **shall** ⁽⁸⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

The *Remote SAP ID* argument **shall** ⁽⁹⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_ESTABLISHED Primitive is the response.

A.2.1.21 S_HARD_LINK_REJECTED Primitive

Name :

S_HARD_LINK_REJECTED

Arguments :

1. Reason
2. Link Priority
3. Link Type

4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

The S_HARD_LINK_REJECTED primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer as a negative response to a client's S_HARD_LINK_ESTABLISH primitive.

The *Reason* argument **shall** ⁽²⁾ specify why the Hard Link Request was rejected, with values defined for this argument as specified in the table below:

| Reason | Value |
|-------------------------------|-------|
| Remote-Node-Busy | 1 |
| Higher-Priority-Link-Existing | 2 |
| Remote-Node-Not-Responding | 3 |
| Destination SAP ID not bound | 4 |
| Requested Type 0 Link Exists | 5 |

The argument *Link Priority* **shall** ⁽³⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_REJECTED Primitive is the response.

The *Link Type* argument **shall** ⁽⁴⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_REJECTED Primitive is the response.

The *Remote Node Address* argument **shall** ⁽⁵⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_REJECTED Primitive is the response.

The *Remote SAP ID* argument **shall** ⁽⁶⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_REJECTED Primitive is the response.

A.2.1.22 S_HARD_LINK_TERMINATED Primitive

Name :

S_HARD_LINK_TERMINATED

Arguments :

1. Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

The S_HARD_LINK_TERMINATED primitive **shall** ⁽¹⁾ be issued by the Subnetwork Interface Sublayer to inform a client which has been granted a Hard Link that the Link has been terminated unilaterally by the Subnetwork.

For Hard Link Types 0 and 1, only the client that originally requested the Hard Link **shall** ⁽²⁾ receive this primitive. Other clients sharing the link with Soft-Link Data Exchange Sessions may have the link broken without notification.

For type 2 hard links, both called and calling clients **shall** ⁽³⁾ receive this primitive.

The *Reason* argument **shall** ⁽⁴⁾ specify why the Hard Link was terminated, with values defined for this argument as specified in the table below:

| Reason | Value |
|---------------------------------------|-------|
| Link terminated by remote node | 1 |
| Higher priority link requested | 2 |
| Remote node not responding (time out) | 3 |
| Destination SAP ID unbound | 4 |
| Physical Link Broken | 5 |

The argument *Link Priority* **shall** ⁽⁵⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_TERMINATED Primitive is the response.

The *Link Type* argument **shall** ⁽⁶⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_TERMINATED Primitive is the response.

The *Remote Node Address* argument **shall** ⁽⁷⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_TERMINATED Primitive is the response.

The *Remote SAP ID* argument **shall** ⁽⁸⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_TERMINATED Primitive is the response.

A.2.1.23 S_HARD_LINK_INDICATION Primitive

Name :

S_HARD_LINK_INDICATION

Arguments :

1. Remote Node Status
2. Link Priority
3. Link Type

4. Remote Node Address
5. Remote SAP ID

Direction :

Subnetwork Interface-> Client

Description:

The S_HARD_LINK_INDICATION primitive **shall** ⁽¹⁾ be used only for Hard Link Type 2. With this primitive the Subnetwork Interface Sublayer **shall** ⁽²⁾ signal to one of its local clients that a client at a remote node requested a Hard Link of Type 2 to be established between them.

The first argument, *Remote Node Status*, **shall** ⁽³⁾ inform the local client of any special status of the remote node, e.g. Remote Node in EMCON, etc. Valid arguments currently defined for *Remote Node Status* are given in the table below.

| Remote Node Status | Value |
|--------------------|-------|
| ERROR | 0 |
| OK | >=1 |

Subsequent versions of this STANAG and implementation-dependent options may define additional values for the remote node status. At present, a minimally compliant client implementation may ignore this argument.

The argument *Link Priority* **shall** ⁽⁴⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive generated by the remote-client and for which this S_HARD_LINK_INDICATION Primitive is the result.

The *Link Type* argument **shall** ⁽⁵⁾ have the same meaning and be equal in value to the argument of the S_HARD_LINK_ESTABLISH Primitive generated by the remote-client and for which this S_HARD_LINK_INDICATION Primitive is the result.

The *Remote Node Address* argument **shall** ⁽⁶⁾ be equal in value to the HF subnetwork address of the node to which the remote-client is bound and that originated the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_INDICATION Primitive is the result.

The *Remote SAP ID* argument **shall** ⁽⁷⁾ be equal in value to the SAP_ID that is bound to the remote client that originated the S_HARD_LINK_ESTABLISH Primitive for which this S_HARD_LINK_INDICATION Primitive is the result.

A.2.1.24 S_HARD_LINK_ACCEPT Primitive

Name :

S_HARD_LINK_ACCEPT

Arguments :

1. Link Priority
2. Link Type
3. Remote Node Address
4. Remote SAP ID

Direction :

Client-> Subnetwork Interface

Description:

The S_HARD_LINK_ACCEPT primitive **shall** ⁽¹⁾ be issued by a client as a positive response to a S_HARD_LINK_INDICATION primitive. With this primitive the client tells the Subnetwork Interface Sublayer that it accepts the Hard Link of Type 2 requested by a client at a remote node.

The argument *Link Priority* **shall** ⁽²⁾ have the same meaning and be equal in value to the *Link Priority* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ACCEPT Primitive is the response.

The *Link Type* argument **shall** ⁽³⁾ have the same meaning and be equal in value to the *Link Type* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ACCEPT Primitive is the response.

The *Remote Node Address* argument **shall** ⁽⁴⁾ have the same meaning and be equal in value to the *Remote Node Address* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ACCEPT Primitive is the response.

The *Remote SAP ID* argument **shall** ⁽⁵⁾ have the same meaning and be equal in value to the *Remote SAP ID* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ACCEPT Primitive is the response.

A.2.1.25 S_HARD_LINK_REJECT Primitive

Name :

S_HARD_LINK_REJECT

Arguments :

1. Reason
2. Link Priority
3. Link Type
4. Remote Node Address
5. Remote SAP ID

Direction :

Client-> Subnetwork Interface

Description:

The S_HARD_LINK_REJECT primitive **shall** ⁽¹⁾ be issued by a client as a negative response to a S_HARD_LINK_INDICATION primitive. With this primitive the client tells the Subnetwork Interface Sublayer that it rejects the Hard Link of Type 2 requested by a client at a remote node.

The *Reason* argument **shall** ⁽²⁾ specify why the hard link is rejected. Possible values of this argument are Mode-Not-Supported (for Link Type 2), I-Have-Higher-Priority-Data, etc.

The argument *Link Priority* **shall** ⁽³⁾ have the same meaning and be equal in value to the *Link Priority* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_REJECT Primitive is the response.

The *Link Type* argument **shall** ⁽⁴⁾ have the same meaning and be equal in value to the *Link Type* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_REJECT Primitive is the response.

The *Remote Node Address* argument **shall** ⁽⁵⁾ have the same meaning and be equal in value to the *Remote Node Address* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_REJECT Primitive is the response.

The *Remote SAP ID* argument **shall** ⁽⁶⁾ have the same meaning and be equal in value to the *Remote SAP ID* argument of the S_HARD_LINK_INDICATION Primitive received by the client from the Subnetwork for which this S_HARD_LINK_ACCEPT Primitive is the response.

A.2.1.26 S_SUBNET_AVAILABILITY Primitive

Name :

S_SUBNET_AVAILABILITY

Arguments :

1. Node Status
2. Reason

Direction :

Subnetwork Interface-> Client

Description:

The S_SUBNET_AVAILABILITY primitive may be sent asynchronously to all or selected clients connected to the Subnetwork Interface Sublayer to inform them of changes in the status of the node to which they are attached. For example, clients can be informed using this primitive that available resources (e.g., bandwidth) have been temporarily reserved by a high ranked client. Alternatively, this primitive could be used to inform clients that the node has entered an EMCON state and as a result they should only expect to receive Data and will not be allowed to transmit data.

The contents of this primitive are implementation dependent and not specified in this version of STANAG 5066. At present, a minimally compliant client implementation **shall** ⁽¹⁾ be capable of receiving this primitive, without further requirement to process its contents.

As implementation options, the *Node Status* argument could specify the new Status of the node. Possible values of this argument could be ON, OFF, Receive-Only, Transmit-Only-to-Specific-Destination-Node/SAP, etc.

| Node Status | Value |
|---------------|-------|
| OFF | 0 |
| ON | 1 |
| Receive-Only | 2 |
| Half-Duplex | 3 |
| Full-Duplex | 4 |
| Transmit-Only | 5 |

If the Subnetwork Status is other than ON, the *Reason* argument explains why. Values of this argument shall be as specified below.

| Reason | Value |
|--------------------------------|-------|
| unspecified | 0 |
| Local Node in EMCON | 1 |
| Higher priority link requested | 2 |
| ... | |

A.2.2 Encoding of Primitives

The encoding of the S_Primitives for communication across the Subnetwork Interface Sublayer **shall** ⁽¹⁾ be in accordance with text and figures in the subsections below.

A.2.2.1 Generic Field Encoding Requirements

Unless noted otherwise, the bit representation for argument values in an S_Primitive **shall** ⁽¹⁾ be encoded into their corresponding fields in accordance with CCITT V.42, 8.1.2.3, which states that:

- when a field is contained within a single octet (i.e, eight bit group), the lowest bit number of the field **shall** ⁽²⁾ represent the lowest-order (i.e., least-significant-bit) value;
- | when a field spans more than one octet, the order of bit values within each octet **shall** ⁽³⁾ progressively decrease as the octet number increases. The lowest bit number associated with the field represents the lowest-order value.

The 4-byte address field in the S_primitives **shall** ⁽⁴⁾ carry the 3.5-byte address and address-size information defined in A.2.2.28.1. The lowest order bit of the address shall be placed in the lowest order bit position of the field (generally bit 0 of the highest byte number of the field), consistent with the mapping specified in Section C.3.2.6 for D_PDUs.

A.2.2.2 S_Primitive Generic Elements and Format

As shown in Figure A-1(a), all primitives **shall** ⁽¹⁾ be encoded as the following sequence of elements:

- a two-byte S_Primitive preamble field, whose value is specified by the 16-bit Maury-Styles sequence below;
- a one-byte version-number field;
- a two-byte Size_of_Primitive field;

- a multi-byte field that contains the encoded S_Primitive.

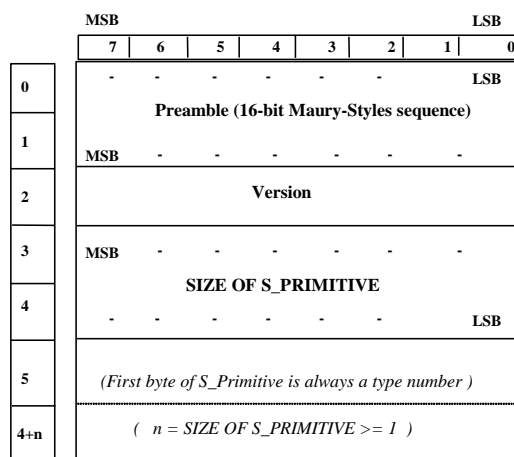


Figure A-1(a): Element-Sequence Encoding of “S_” Primitives

The S_Primitive preamble field **shall** ⁽²⁾ be encoded as the 16-bit Maury-Styles sequence shown below, with the least significant bit (LSB) transmitted first over the interface:

(MSB) 1 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 (LSB)

i.e., with the multi-byte S_Primitive field represented in hexadecimal form as 0xEB90, the least-significant bits of the sequence **shall** ⁽³⁾ be encoded in the first byte (i.e, byte number 0) of the preamble field and the most significant bits of the sequence **shall** ⁽⁴⁾ be encoded in the second byte (i.e, byte number 1) of the preamble field as follows:

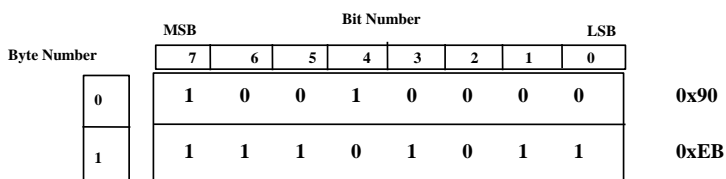


Figure A-1(b): Encoding of Maury-Styles Preamble-Sequence in “S_” Primitives

[**Note:** This encoding of the Maury-Styles preamble sequence is an exception to the general requirement of section 2.2.1 for field encoding.]

Following the Maury-Styles sequence, the next 8 bit (1-byte) field **shall** ⁽⁵⁾ encode the 5066 version number. For this version of STANAG 5066, the version number **shall** ⁽⁶⁾ be all zeros, i.e, the hexadecimal value 0x00, as follows:

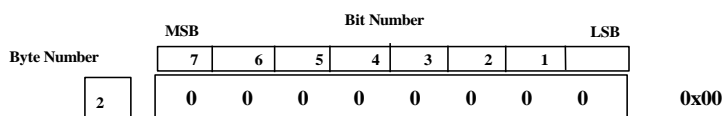


Figure A-1(c): Encoding of Version Number in “S_” Primitives

The next 16 bit (two-byte) field **shall** ⁽⁷⁾ encode the size in bytes of the S_primitive-dependent field to follow, exclusive of the Maury-Styles sequence, version field, and this size field. The LSB of the of the size value **shall** ⁽⁸⁾ be mapped into the low order bit of the low-order byte of the field, as follows:

| Byte Number | | Bit Number | | | | | | | | MSB | LSB |
|-------------|---|------------|---|---|---|---|---|---|-----|-----------------|-----|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 3 | 4 | MSB | - | - | - | - | - | - | - | High-order byte | |
| | | - | - | - | - | - | - | - | LSB | Low-order byte | |

Figure A-1(d): Encoding of Size_of_S_Primitive Element in “S_” Primitives

Unless specified otherwise, the order of bit transmission for each byte in the encoded S_Primitive **shall** ⁽⁹⁾ be as described in CCITT V.42 paragraph 8.1.2.2, which specifies the least significant bit (LSB, bit 0 in the figures below) of byte 0 **shall** ⁽¹⁰⁾ be transmitted first.

The sixth byte (i.e., byte number 5) of the sequence **shall** ⁽¹¹⁾ be the first byte of the encoded primitive and **shall** ⁽¹²⁾ be equal to the S_Primitive type number, with values encoded in accordance with the respective section that follows for each S_primitive

The remaining bytes, if any, in the S_Primitive **shall** ⁽¹³⁾ be transmitted sequentially, also beginning with the LSB of each byte, in accordance with the respective section that follows for each S_primitive.

In the subsections that follow, any bits in a S_Primitive that are specified as NOT USED **shall** ⁽¹³⁾ be encoded with the value “0” unless specified otherwise for the specific S_Primitive being defined.

A.2.2.3 S_BIND_REQUEST Encoding

The S_BIND_REQUEST primitive **shall** ⁽¹⁾ be encoded as a four-byte field as follows:

| MSB | | | | | | | | LSB | |
|-----|-----------------|---|---|-----------|---|---|---|-----------------------------|--|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| 5 | TYPE * | | | | | | | * TYPE = 1 (S_BIND_REQUEST) | |
| 6 | SAP ID | | | RANK | | | | | |
| 7 | MSB - - - - - | | | | | | | ** SERVICE-TYPE SUBFIELD | |
| | SERVICE TYPE ** | | | - - - - - | | | | ATTRIBUTE ENCODING | |
| 8 | - - - | | | LSB | | | | as specified herein | |
| | | | | NOT USED | | | | | |

Figure A-2: Encoding of S_BIND_REQUEST Primitive

The S_BIND_REQUEST SERVICE-TYPE field **shall** ⁽²⁾ be encoded as five subfields as follows:

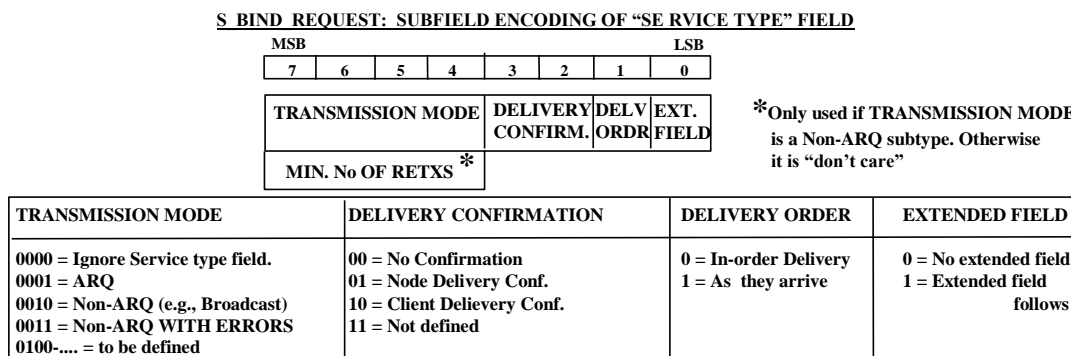


Figure A-3: Sub-field Attribute Encoding of S_BIND_REQUEST SERVICE-TYPE field.

Argument : SERVICE TYPE
Primitive : S_BIND_REQUEST

The SERVICE TYPE argument **shall** ⁽³⁾ specify the default type of service requested by the client. This type of service **shall** ⁽⁴⁾ apply to any U_PDU submitted by the client until the client unbinds itself from the node, unless overridden by the DELIVERY MODE argument of the U_PDU. A client **shall** ⁽⁵⁾ change the default service type only by unbinding and binding again with a new S_BIND_REQUEST.

The SERVICE TYPE argument is complex, consisting of a number of attributes encoded as sub-fields. Although the exact number of attributes and their encoding is left for future definition and enhancement using the Extended Field attribute, the following attributes are mandatory:

1. *Transmission Mode for the Service.* --- ARQ or Non-ARQ Transmission Mode **shall** ⁽⁶⁾ be specified, with one of the Non-ARQ submodes if Non-ARQ was requested. A value of "0" for this attribute **shall** ⁽⁷⁾ be invalid for the SERVICE TYPE argument when binding. Non-ARQ transmission can have submodes such as: *Error-Free-Only* delivery to destination client, delivery to destination client even with *some* errors.
2. *Data Delivery Confirmation for the Service* --- The client **shall** ⁽⁸⁾ request one of the Data Delivery Confirmation modes for the service. There are three types of data delivery confirmation:
 - None
 - Node-to-Node Delivery Confirmation
 - Client-to-Client Delivery Confirmation

The client can request explicit confirmation, i.e, Node-to-Node or Client-to-Client, from the Subnetwork to provide indication that its U_PDUs have been properly delivered to their destination. Explicit delivery confirmation **shall** ⁽⁹⁾ be requested only in combination with ARQ delivery.

[Note: The Node-to-Node Delivery Confirmation does not require any explicit peer-to-peer communication between the Subnetwork Interface Sublayers and hence it does not introduce extra

overhead. It simply uses the ACK (ARQ) confirmation provided by the Data Transfer Sublayer. Client-to-Client Delivery Confirmation requires explicit peer-to-peer communication between the Sublayers and therefore introduces overhead. It should be used only when it is absolutely critical for the client to know whether or not its data was delivered to the destination client (which may, for instance, be disconnected).]

3. *Order of delivery of any U_PDU to the receiving client.* --- A client **shall** ⁽¹⁰⁾ request that its U_PDUs are delivered to the destination client “in-order” (as they are submitted) or in the order they are received by the destination node.
4. *Extended Field* --- Denotes if additional fields in the SERVICE TYPE argument are following; at present this capability of the SERVICE TYPE is undefined, and the value of the Extended Field Attribute **shall** ⁽¹¹⁾ be set to “0”.
5. *Minimum Number of Retransmissions* --- This argument **shall** ⁽¹²⁾ be valid if and only if the Transmission Mode is a Non-ARQ type. If the Transmission Mode is a Non-ARQ type, then the subnetwork **shall** ⁽¹³⁾ retransmit each U_PDU the number of times specified by this argument. This argument may be “0”, in which case the U_PDU is sent only once.
[Note: In non-ARQ Mode, automatic retransmission a minimum number of times may be used to improve the reliability of broadcast transmissions where a return link from the receiver is unavailable for explicit retransmission requests.]

A.2.2.4 S_UNBIND_REQUEST Encoding

The S_UNBIND_REQUEST primitive **shall** ⁽¹⁾ be encoded as a one-byte field as follows:

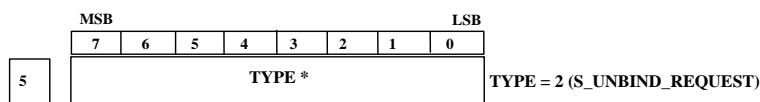


Figure A-4: Encoding of S_UNBIND_REQUEST Primitive

A.2.2.5 S_BIND_ACCEPTED Encoding

The S_BIND_ACCEPTED primitive **shall** ⁽¹⁾ be encoded as a four-byte field as follows:

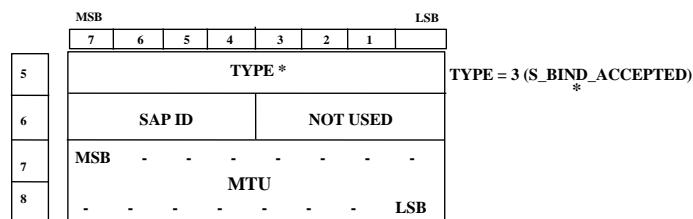


Figure A-5: Encoding of S_BIND_ACCEPTED Primitive

A.2.2.6 S_BIND_REJECTED Encoding

The S_BIND_REJECTED primitive **shall** ⁽¹⁾ be encoded as a two-byte field as follows:

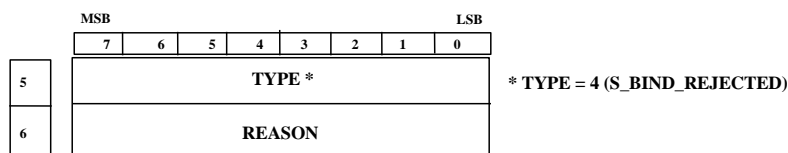


Figure A-6: Encoding of S_BIND_REJECTED Primitive

A.2.2.7 S_UNBIND_INDICATION Encoding

The S_UNBIND_INDICATION primitive **shall** ⁽¹⁾ be encoded as a two-byte field as follows:

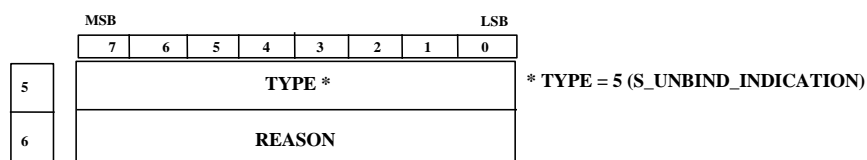


Figure A-7: Encoding of S_UNBIND_INDICATION Primitives

A.2.2.8 S_HARD_LINK_ESTABLISH Encoding

The S_HARD_LINK_ESTABLISH primitive **shall** ⁽¹⁾ be encoded as a six-byte field as follows:

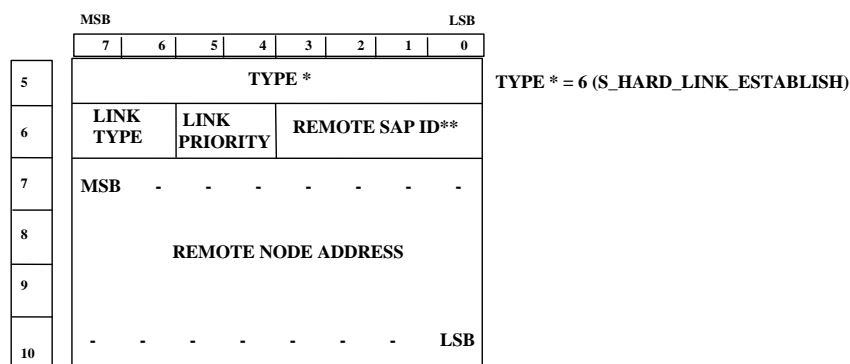


Figure A-8: Encoding of S_HARD_LINK_ESTABLISH Primitives

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.9 S_HARD_LINK_TERMINATE Encoding

The S_HARD_LINK_TERMINATE primitive **shall** ⁽¹⁾ be encoded as a five-byte field as follows:

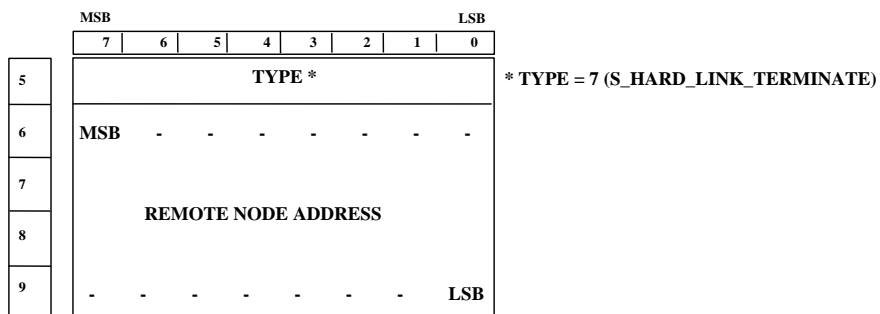


Figure A-9: Encoding of S_HARD_LINK_TERMINATE Primitives

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.10 S_HARD_LINK_ESTABLISHED Encoding

The S_HARD_LINK_ESTABLISHED primitive **shall** ⁽¹⁾ be encoded as a seven-byte field as follows:

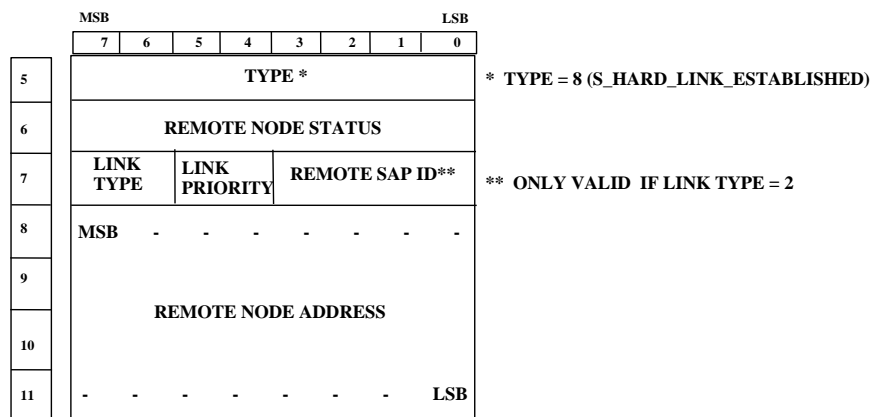


Figure A-10: Encoding of S_HARD_LINK_ESTABLISHED Primitives.

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.11 S_HARD_LINK_REJECTED Encoding

The S_HARD_LINK_REJECTED primitive **shall** ⁽¹⁾ be encoded as a seven-byte field as follows:

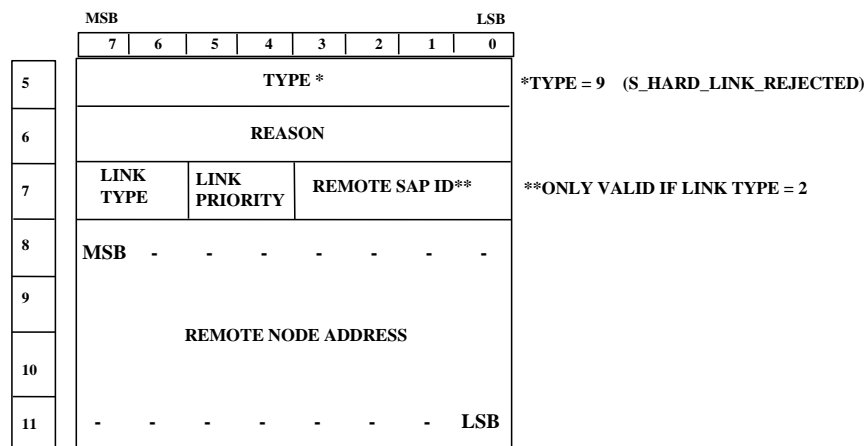


Figure A-11: Encoding of S_HARD_LINK_REJECTED Primitives.

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.12 S_HARD_LINK_TERMINATED Encoding

The S_HARD_LINK_TERMINATED primitive **shall** ⁽¹⁾ be encoded as a seven-byte field as follows:

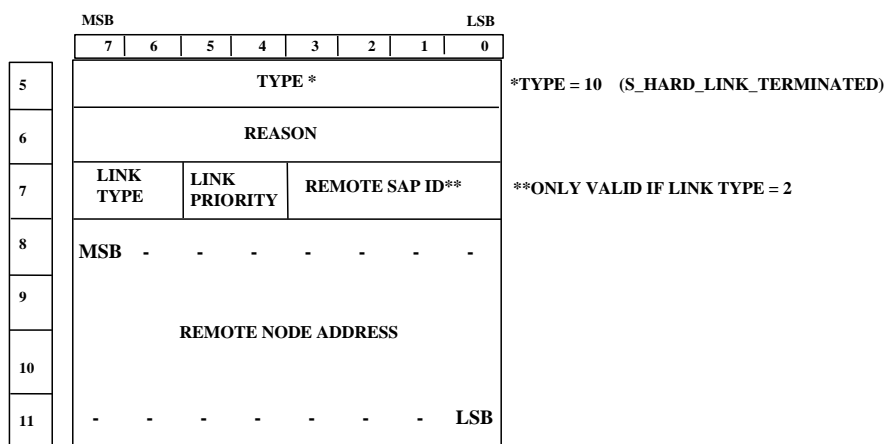


Figure A-12: Encoding of S_HARD_LINK_TERMINATED Primitives.

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.13 S_HARD_LINK_INDICATION Encoding

The S_HARD_LINK_INDICATION primitive **shall** ⁽¹⁾ be encoded as a seven-byte field as follows:

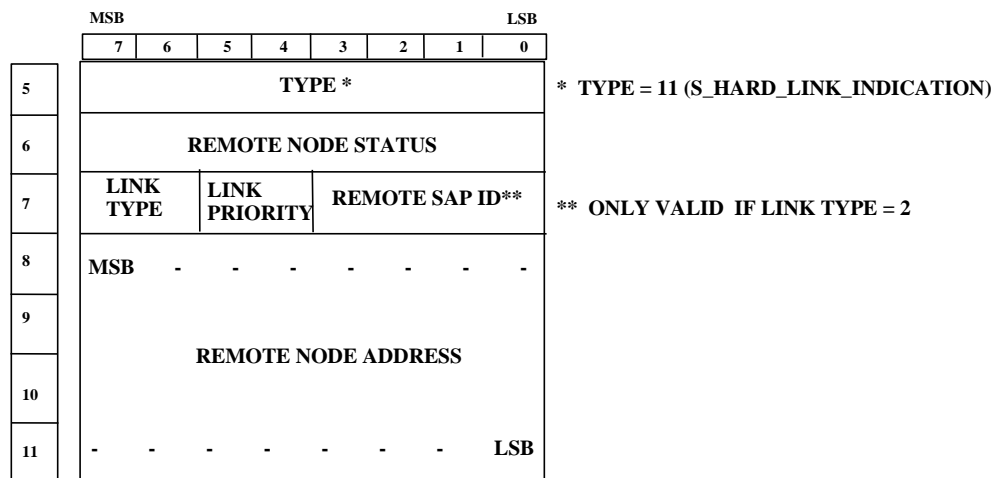


Figure A-13: Encoding of S_HARD_LINK_INDICATION Primitives.

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.14 S_HARD_LINK_ACCEPT Encoding

The S_HARD_LINK_ACCEPT primitive **shall** ⁽¹⁾ be encoded as a six-byte field as follows:

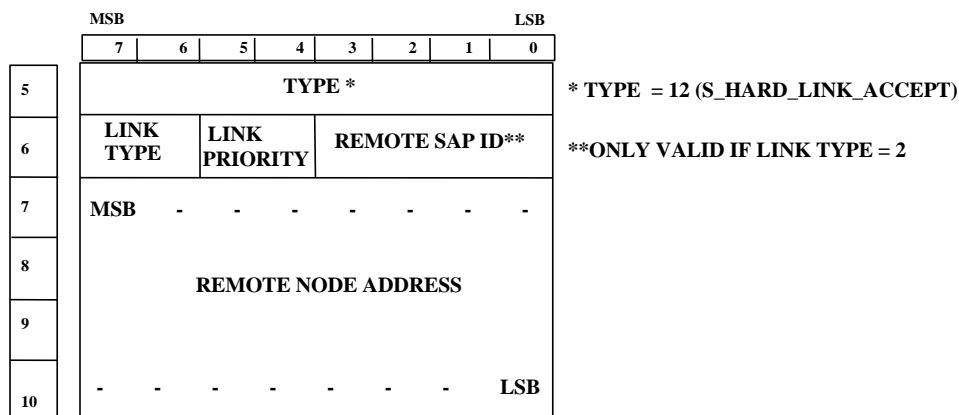


Figure A-14: Encoding of S_HARD_LINK_ACCEPT Primitives

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.15 S_HARD_LINK_REJECT Encoding

The S_HARD_LINK_REJECT primitive **shall** ⁽¹⁾ be encoded as a seven-byte field as follows:

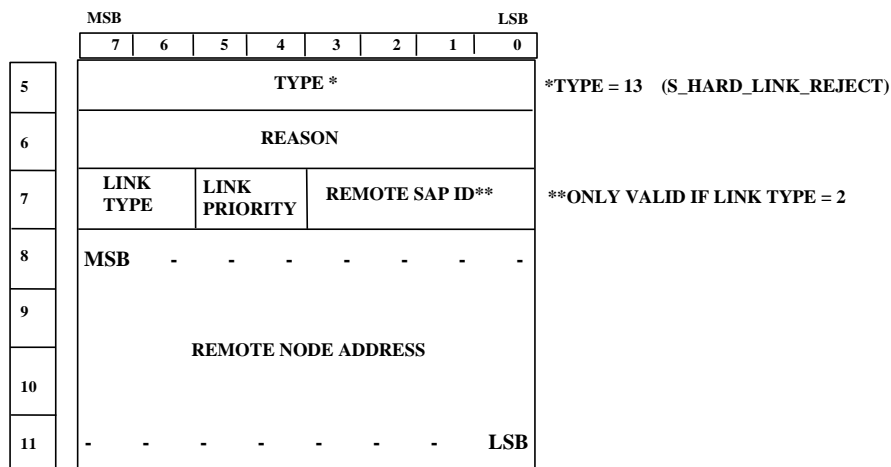


Figure A-15: Encoding of S_HARD_LINK_REJECT Primitives.

The REMOTE NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The LINK TYPE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.4.

A.2.2.16 S_SUBNET_AVAILABILITY Encoding

The S_SUBNET_AVAILABILITY primitive **shall** ⁽¹⁾ be encoded as a three-byte field as follows:

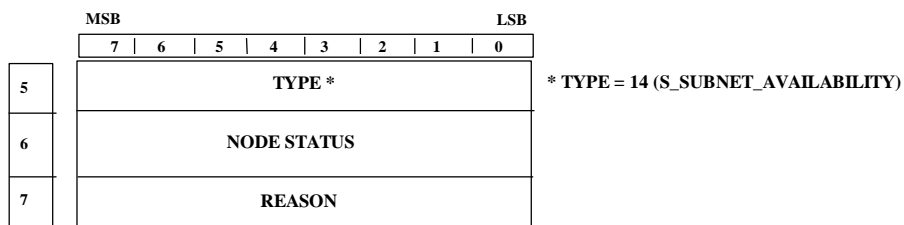


Figure A-16: Encoding of S_SUBNET_AVAILABILITY Primitives.

The encoding of the NODE STATUS and REASON fields is implementation dependent.

A.2.2.17 S_DATA_FLOW_ON and S_DATA_FLOW_OFF Encoding

The S_DATA_FLOW_ON and S_DATA_FLOW_OFF primitives **shall** ⁽¹⁾ be encoded as one-byte fields as follows:

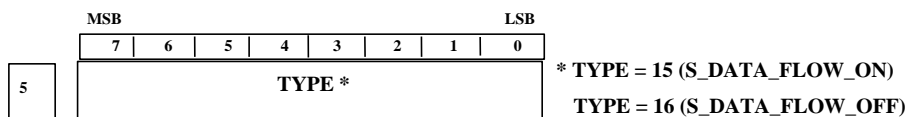


Figure A-17: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

A.2.2.18 S_KEEP_ALIVE Encoding

The S_KEEP_ALIVE primitive **shall** ⁽¹⁾ be encoded as a one-byte field as follows:



Figure A-18: Encoding of S_DATA_FLOW_ON and S_DATA_FLOW_OFF Primitives.

A.2.2.19 S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_INDICATION Encoding

The S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_INDICATION primitives **shall** ⁽¹⁾ be encoded as implementation-dependent variable-length fields as follows:

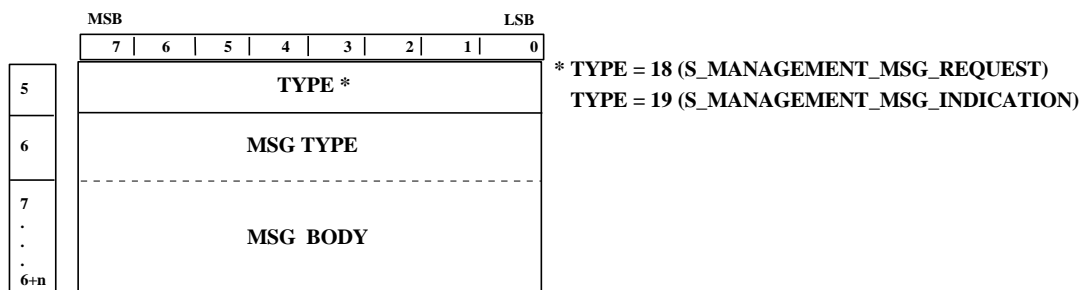


Figure A-19:: Encoding of S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_INDICATION Primitives.

The encoding of the MSG TYPE and MSG BODY fields is implementation dependent.

A.2.2.20 S_UNIDATA_REQUEST Encoding

The S_UNIDATA_REQUEST primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

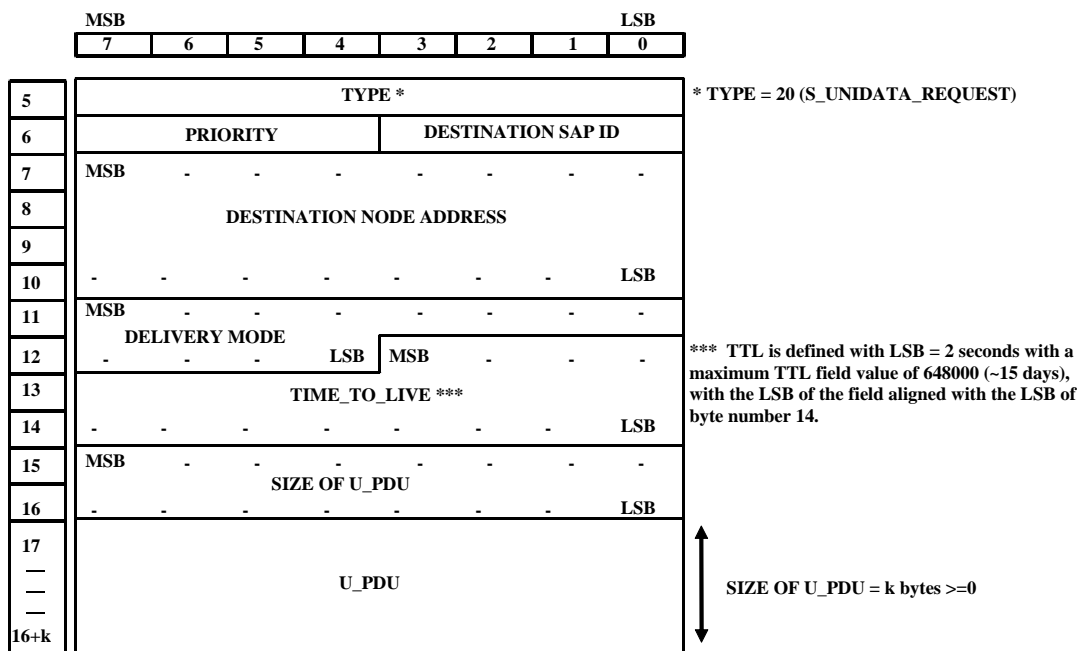


Figure A-20: Encoding of S_UNIDATA_REQUEST Primitives.

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The DELIVERY MODE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.2.

A.2.2.21 S_UNIDATA_INDICATION Encoding

The S_UNIDATA_INDICATION primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

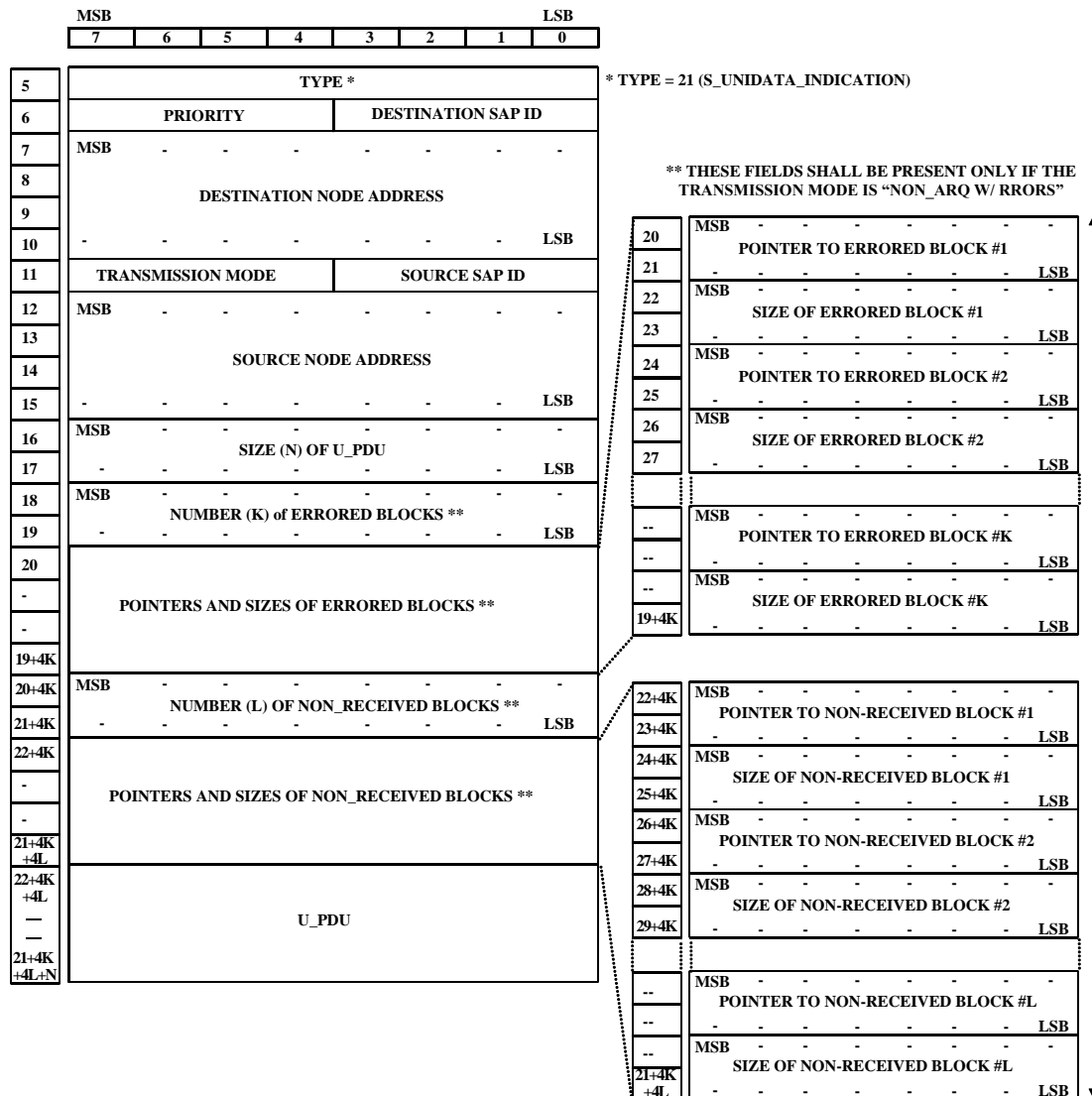


Figure A-21: Encoding of S_UNIDATA_INDICATION Primitives

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The TRANSMISSION MODE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.3.

A.2.2.22 S_UNIDATA_REQUEST_CONFIRM Encoding

The S_UNIDATA_REQUEST_CONFIRM primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

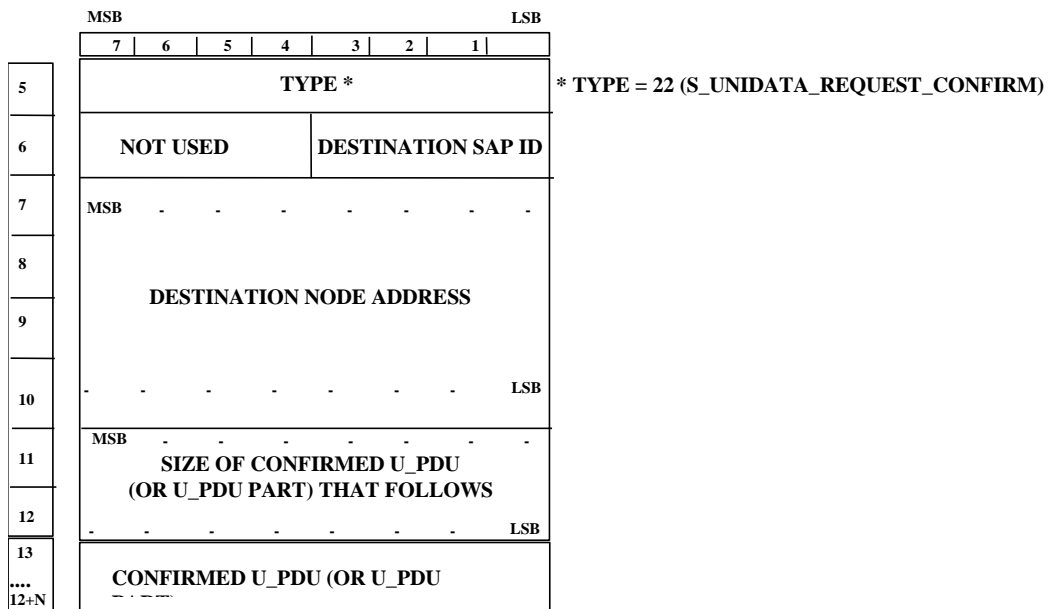


Figure A-22: Encoding of S_UNIDATA_REQUEST_CONFIRM Primitives.

The DESTINATION NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

A.2.2.23 S_UNIDATA_REQUEST_REJECTED Encoding

The S_UNIDATA_REQUEST_REJECTED primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

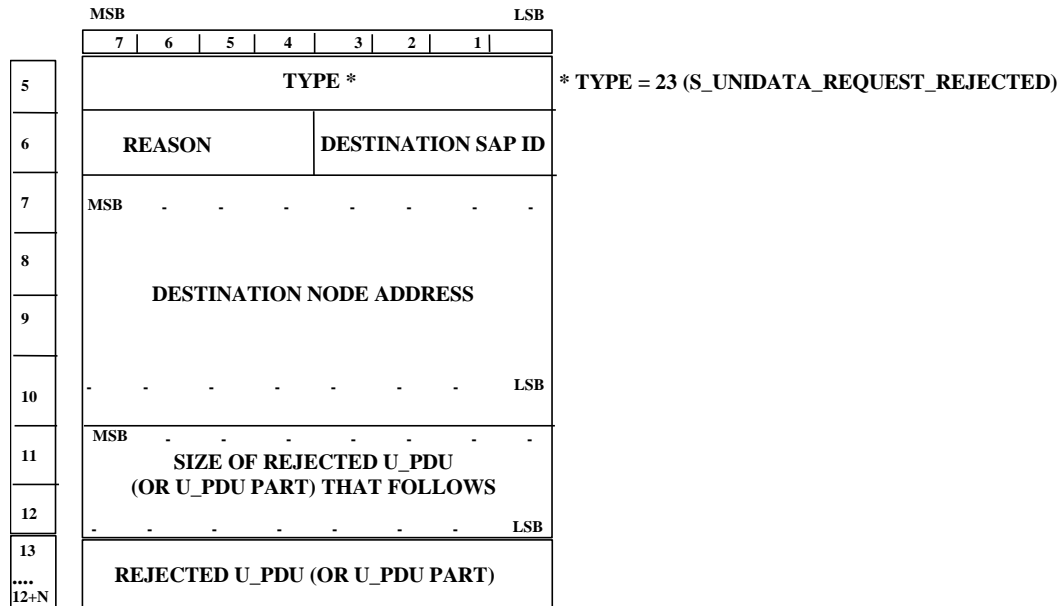


Figure A-23: Encoding of S_UNIDATA_REQUEST_REJECTED Primitives.

The DESTINATION NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

A.2.2.24 S_EXPEDITED_UNIDATA_REQUEST Encoding

The S_EXPEDITED_UNIDATA_REQUEST primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

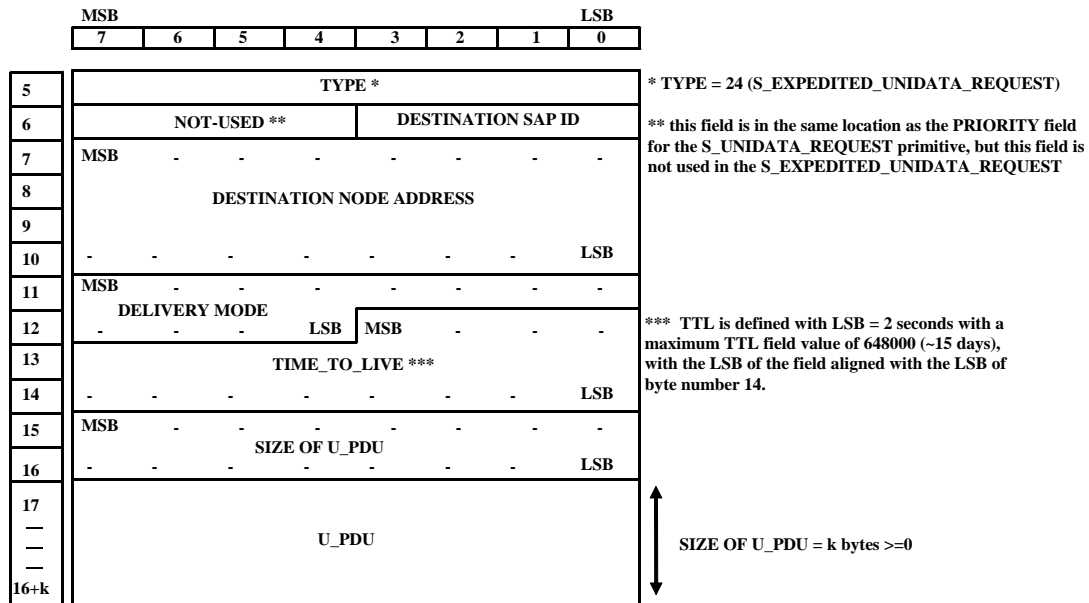


Figure A-24: Encoding of S_EXPEDITED_UNIDATA_REQUEST Primitives.

The DESTINATION NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The DELIVERY MODE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.2.

A.2.2.25 S_EXPEDITED_UNIDATA_INDICATION Encoding

The S_EXPEDITED_UNIDATA_INDICATION primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

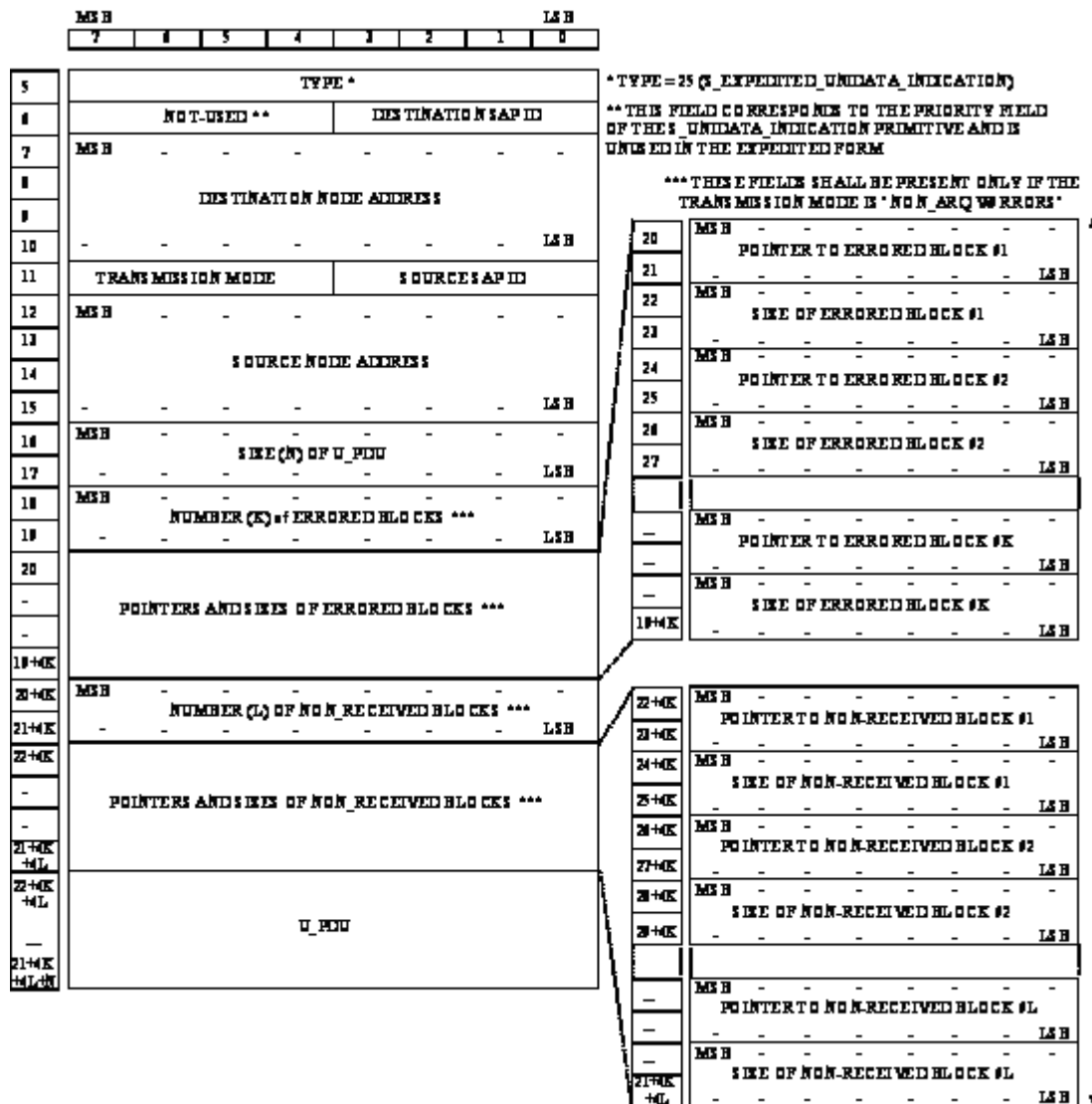


Figure A-25: Encoding of S_EXPEDITED_UNIDATA_INDICATION Primitives

The SOURCE NODE ADDRESS and DESTINATION NODE ADDRESS fields **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

The TRANSMISSION MODE field **shall** ⁽³⁾ be encoded as specified in Section A.2.2.28.3.

A.2.2.26 S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Encoding

The S_EXPEDITED_UNIDATA_REQUEST_CONFIRM primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

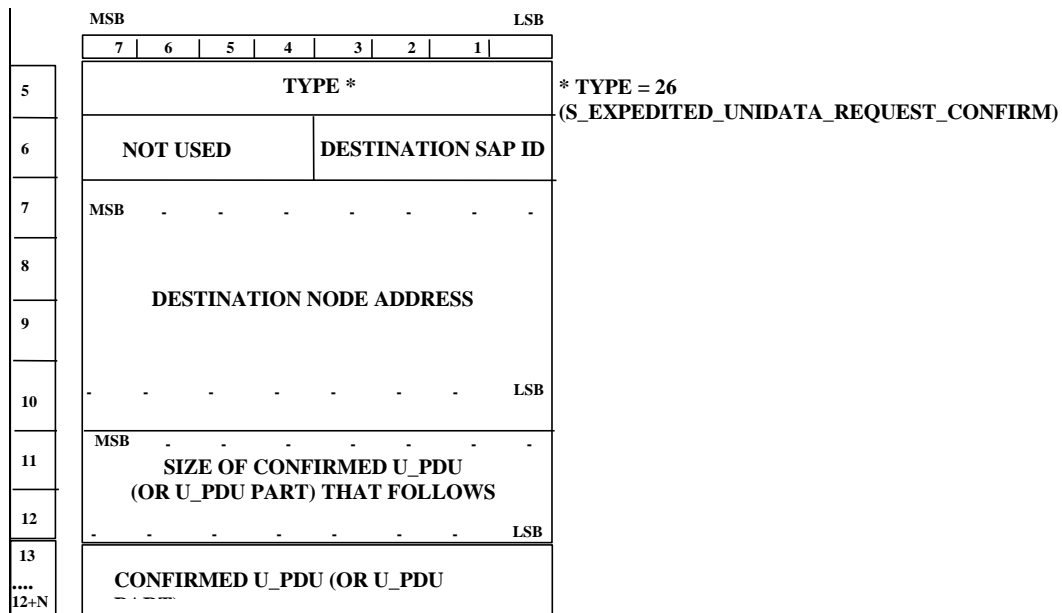


Figure A-26: Encoding of S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitives.

The DESTINATION NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

A.2.2.27 S_EXPEDITED_UNIDATA_REQUEST_REJECTED Encoding

The S_EXPEDITED_UNIDATA_REQUEST_REJECTED primitive **shall** ⁽¹⁾ be encoded as a variable-length field as follows:

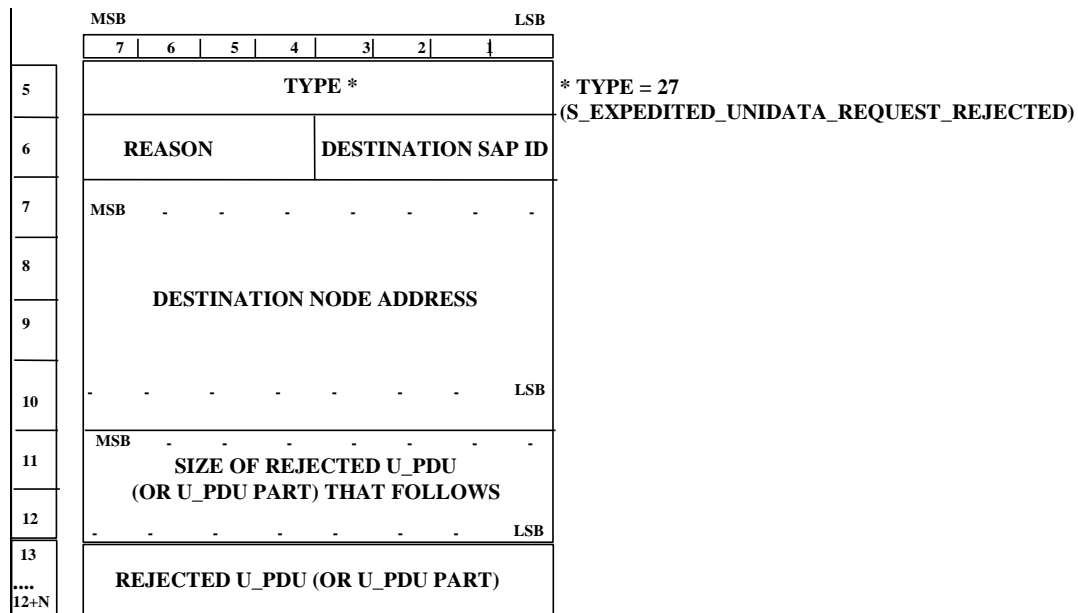


Figure A-27: Encoding of S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitives.

The DESTINATION NODE ADDRESS field **shall** ⁽²⁾ be encoded as specified in Section A.2.2.28.1.

A.2.2.28 Additional S_Primitive Encoding Requirements: Encoding of Common Fields

In order to clarify some of the procedures and tasks executed by the sublayers, additional details concerning some of the arguments of the Primitives described in previous sections are provided below.

A.2.2.28.1 Node ADDRESS Encoding for all Primitives

Arguments : SOURCE NODE ADDRESS, DESTINATION NODE ADDRESS, or REMOTE NODE ADDRESS

Primitives : ALL "UNIDATA" primitives and "S_HARD_LINK" primitives.

For reduced overhead in transmission, node addresses **shall** ⁽¹⁾ be encoded in one of several formats that are multiples of 4-bits (“half-bytes”) in length, as specified in Figure A-28.

Addresses that are encoded as Group node addresses **shall** ⁽²⁾ only be specified as the Destination Node address of Non-ARQ PDUs.

Destination SAP IDs and destination node addresses of ARQ PDUs and source SAP IDs and source node addresses of all PDUs **shall** ⁽³⁾ be individual SAP IDs and individual node addresses respectively.

Remote node addresses and remote SAP IDs of all "S_HARD_LINK" primitives **shall** ⁽³⁾ be individual SAP IDs and individual node addresses respectively.

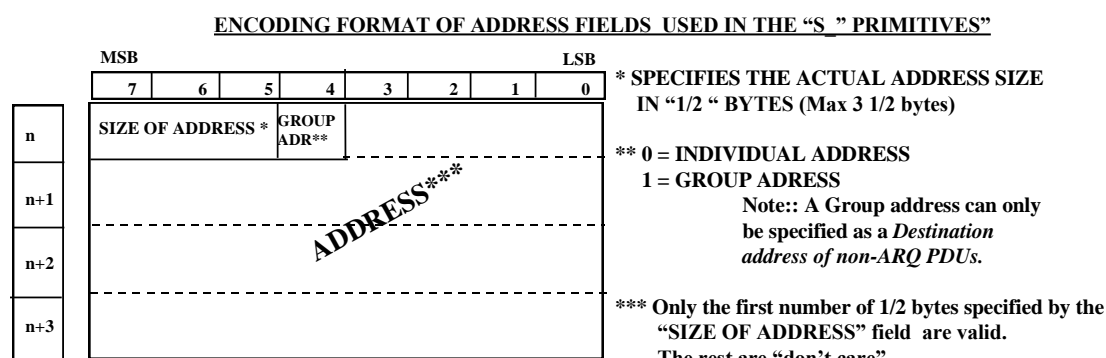


Figure A-28: Encoding of Address Fields in S_Primitives.

A.2.2.28.2 Delivery-Mode Encoding for the S_UNIDATA_REQUEST and S_EXPEDITED_UNIDATA_REQUEST Primitives

Argument : DELIVERY MODE

Primitive : S_UNIDATA_REQUEST , S_EXPEDITED_UNIDATA_REQUEST

The DELIVERY MODE is a complex argument consisting of a number of attributes, as specified here. The DELIVERY MODE argument **shall** ⁽¹⁾ be encoded as shown in Figure A-29.

The value of the DELIVERY MODE argument can be "DEFAULT", as encoded by the Transmission Mode attribute. With a value of "DEFAULT", the delivery mode for this U_PDU **shall** ⁽²⁾ be the delivery mode specified in the *Service Type* argument of the S_BIND_REQUEST. A non-DEFAULT value **shall** ⁽³⁾ override the default settings of the Service Type for this U_PDU.

The attributes of this argument are similar to those described in the *Service Type* argument of the S_BIND_REQUEST:

6. *Transmission Mode of this U_PDU*. --- ARQ or Non-ARQ Transmission can be requested. A value of "0" for this attribute **shall** ⁽⁴⁾ equal the value "DEFAULT" for the Delivery Mode. If the DELIVERY MODE is "DEFAULT", all other attributes encoded in the argument **shall** ⁽⁵⁾ be ignored.
7. *Data Delivery Confirmation for this PDU* --- None, node-to-node, or client-to-client.
8. *Order of delivery of this PDU to the receiving client*. --- A client may request that its U_PDUs are delivered to the destination client "in-order" (as they are submitted) or in the order they are received by the destination node.

9. *Extended Field* --- Denotes if additional fields in the DELIVERY MODE argument are following; at present this capability of the DELIVERY MODE is undefined, and the value of the Extended Field Attribute **shall** ⁽⁶⁾ be set to "0".
10. *Minimum Number of Retransmissions* --- This argument **shall** ⁽⁷⁾ be valid if and only if the Transmission Mode is a Non-ARQ type or sub-type. If the Transmission Mode is a Non-ARQ type or subtype, then the subnetwork **shall** ⁽⁸⁾ retransmit each U_PDU the number of times specified by this argument. This argument may be "0", in which case the U_PDU is sent only once.

[Note: In non-ARQ Mode, automatic retransmission a minimum number of times may be used to improve the reliability of broadcast transmissions where a return link from the receiver is unavailable for explicit retransmission requests.]

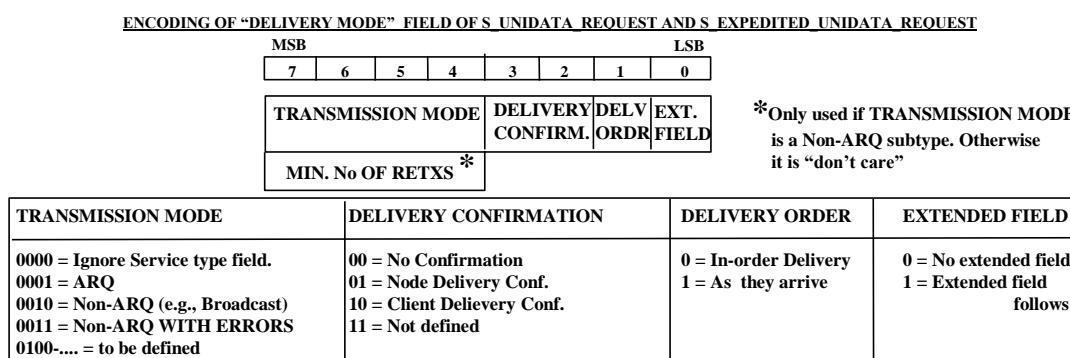


Figure A-29: Encoding of the Delivery Mode field in the S_UNIDATA_REQUEST and S_EXPEDITED_UNIDATA_REQUEST primitives

A.2.2.28.3 TRANSMISSION-MODE Encoding for the S_UNIDATA_INDICATION and S_EXPEDITED_UNIDATA_INDICATION Primitives

Argument: TRANSMISSION-MODE

S_Primitives: S_UNIDATA_INDICATION, S_EXPEDITED_UNIDATA_INDICATION

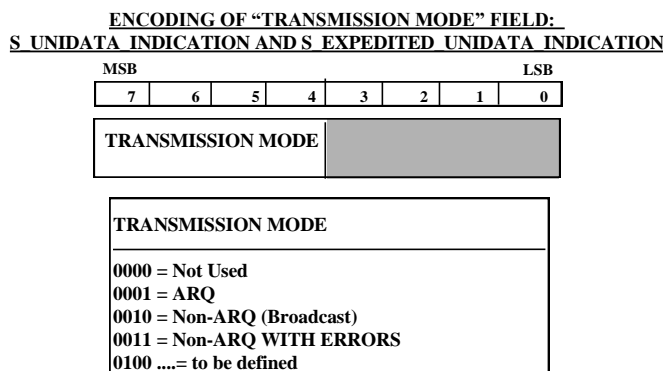


Figure A-30: Encoding of Transmission Mode Field in S_UNIDATA_INDICATION or S_EXPEDITED_UNIDATA_INDICATION primitive.

The subnetwork notifies a client of the transmission-mode used to deliver a U_PDU or Expedited U_PDU Argument with the TRANSMISSION-MODE argument. The TRANSMISSION-MODE argument in the S_UNIDATA_INDICATION and S_EXPEDITED_UNIDATA_INDICATION Primitives **shall** ⁽¹⁾ be encoded as shown in Figure A-30.

[Note: The unused bits in this argument are allocated to the SOURCE SAP_ID argument encoding for both the S_UNIDATA_INDICATION and S_EXPEDITED_UNIDATA_INDICATION Primitives.]

A.2.2.28.4 Link-Type Encoding in S_Primitives

Argument: LINK TYPE

S_Primitives: S_HARD_LINK_ESTABLISH, S_HARD_LINK_ESTABLISHED,
S_HARD_LINK_REJECTED, S_HARD_LINK_ACCEPT, S_HARD_LINK_INDICATION,
S_HARD_LINK_REJECT, S_HARD_LINK_TERMINATED

A client uses the Link-Type argument to reserve partially or fully the capacity of the Hard Link. This argument can have three values:

- A value of 0 **shall** ⁽¹⁾ indicate that the physical link to the specified node address is a Type 0 Hard Link. The Type 0 Hard Link must be maintained, but all clients connected to the two nodes can make use of the link capacity according to normal procedures, i.e. there is no bandwidth reservation.
- A value of 1 **shall** ⁽²⁾ indicate that the physical link to the specified node address is a Type 1 Hard Link. The Type 1 Hard Link must be maintained and traffic is only allowed between the requesting client and any of the clients on the remote Node, i.e., there is partial bandwidth reservation.
- A value of 2 indicates that the physical link to the specified node address must be maintained and traffic is only allowed between the requesting Client and the specific Client on the remote node specified by the remote SAP ID argument, i.e. full bandwidth reservation.

A.3 Peer-to-Peer Communication Protocols and S_PDUs

Peer Subnetwork Interface Sublayers, generally in different nodes, **shall** ⁽¹⁾ communicate with each other by the exchange of Subnetwork Interface Sublayer Protocol Data Units (S_PDUs).

For the Subnetwork configurations currently defined in STANAG 5066, Peer-to-Peer Communication **shall** be ⁽²⁾ required for the:

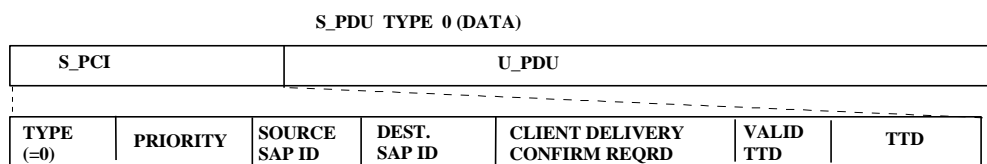
1. Establishment and Termination of Hard Link Data Exchange Sessions
2. Exchange of Client Data

Explicit Peer-to-Peer communication **shall** ⁽³⁾ not be required for the establishment or termination of Soft Link or Broadcast Data Exchange sessions.

The Peer-to-Peer communication required for the exchange of Client Data is similar for all Data exchange sessions, using the facilities of lower sublayers in the protocol profile. The encoding of the S_PDUs and the protocol governing the Peer-to-Peer Communication are described in the following sections.

A.3.1 Subnetwork Interface Sublayer Protocol Data Units (S_PDUS) and Encoding Requirements

There are currently eight types of S_PDUs. Additional S_PDU types may be defined in the future. The generic encoding of the eight S_PDU types showing the fields and subfields of the S_PDUs is shown in Figure A-31.



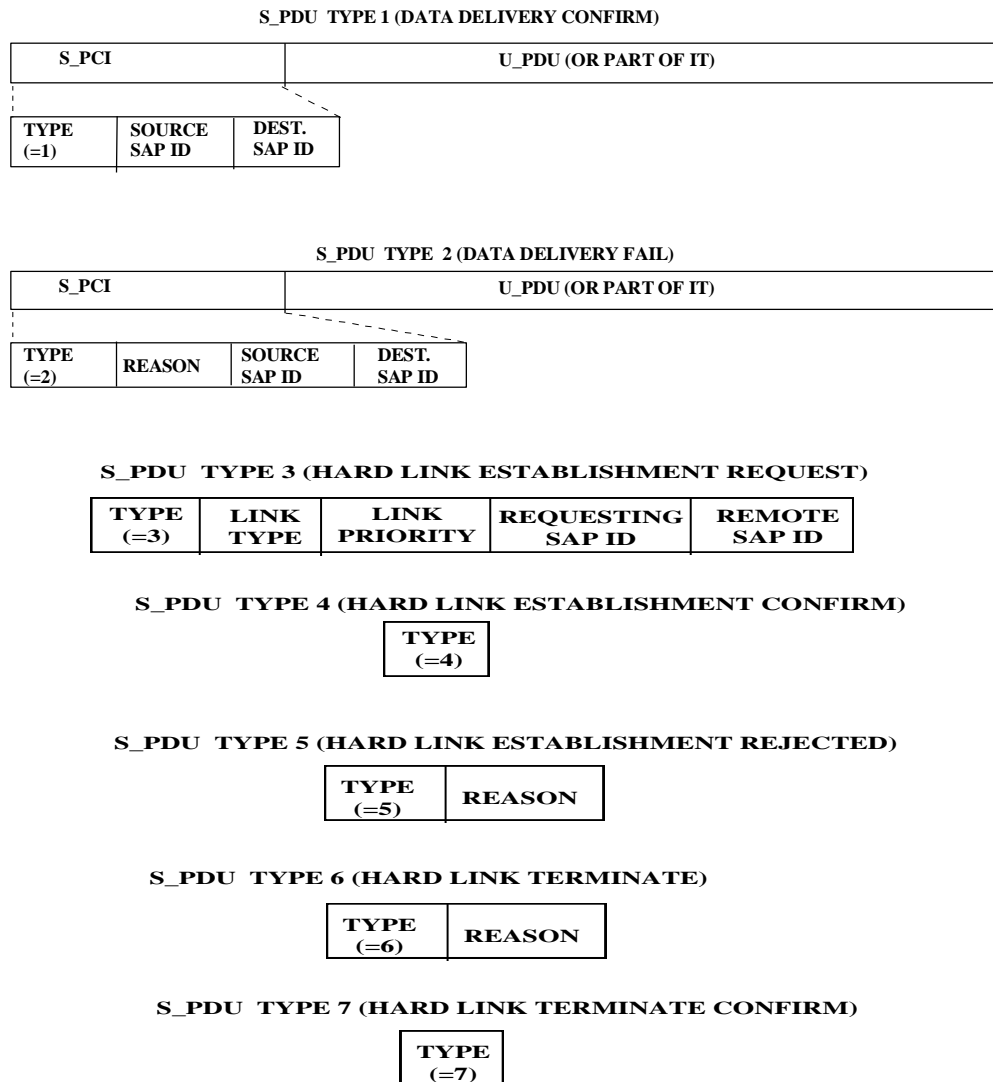


Figure A-31: Generic Encoding of S_PDUs

The first encoded field **shall** ⁽¹⁾ be common to all S_PDU. It is called “TYPE” and **shall** ⁽²⁾ encode the type value of the S_PDU as follows:

| S_PDU TYPE field value | S_PDU Name |
|---------------------------|----------------------------------|
| 0 | DATA |
| 1 | DATA DELIVERY CONFIRM |
| 2 | DATA DELIVERY FAIL |
| 3 | HARD LINK ESTABLISHMENT REQUEST |
| 4 | HARD LINK ESTABLISHMENT CONFIRM |
| 5 | HARD LINK ESTABLISHMENT REJECTED |
| 6 | HARD LINK TERMINATE |
| 7 | HARD LINK TERMINATE CONFIRM |

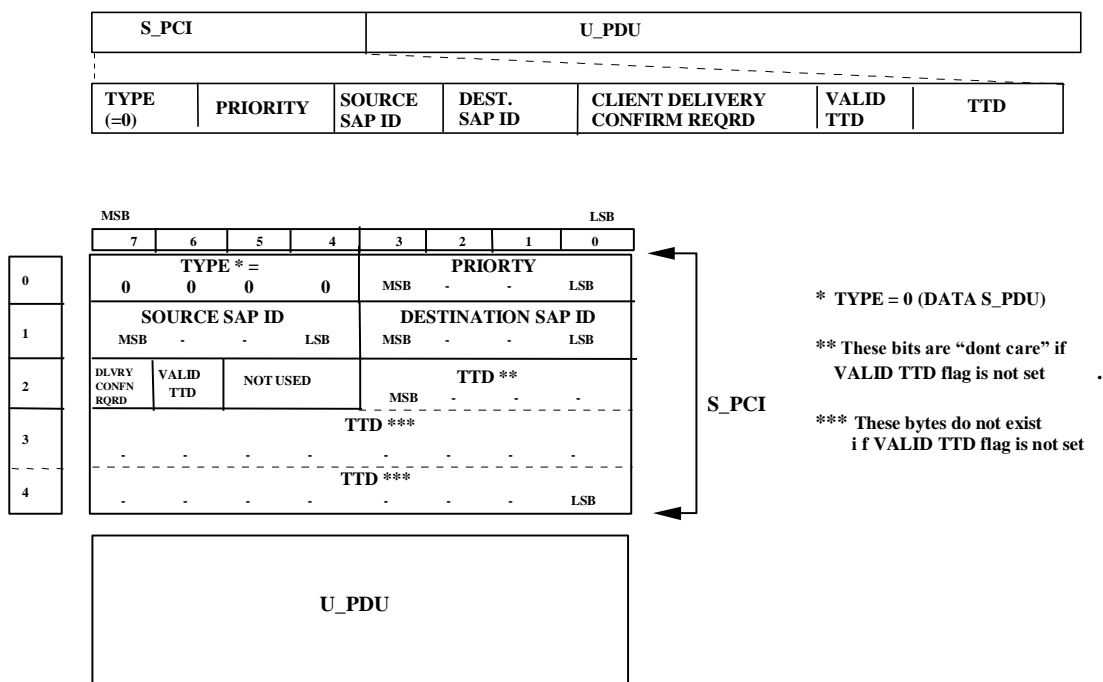
The meaning and encoding of the remaining fields, if any, in an S_PDU **shall** ⁽³⁾ be as specified in the subsection below corresponding to the S_PDU type.

A.3.1.1 DATA S_PDU

Type :

“0” = DATA S_PDU

Encoding :



**Figure A-32: Generic Encoding and Bit-Field Map of the
DATA S_PDU**

Description :

The DATA S_PDU **shall** ⁽¹⁾ be transmitted by the Subnetwork Interface Sublayer in order to send client data to a remote peer sublayer.

The DATA S_PDU **shall** ⁽²⁾ be encoded as specified in Figure A-32 and in the paragraphs below.

This S_PDU **shall** ⁽³⁾ consist of two parts:

- a) the first part **shall** ⁽⁴⁾ be the S_PCI (Subnetwork Interface Sublayer Protocol Control Information) and represents the overhead added by the sublayer;
- b) the second part **shall** ⁽⁵⁾ be the actual client data (U_PDU).

The first field of four bits the S_PCI part **shall** ⁽⁶⁾ be "TYPE". Its value **shall** ⁽⁷⁾ be equal to 0 and identifies the S_PDU as being of type DATA.

The second field of four bits **shall** ⁽⁸⁾ be "PRIORITY" and represents the priority of the client's U_PDU. The "PRIORITY" field **shall** ⁽⁹⁾ be equal in value to the corresponding argument of the S_UNIDATA_REQUEST primitive submitted by the client. For U_PDUs submitted with an S_EXPEDITED_UNIDATA_REQUEST, the PRIORITY field **should** be set to 0.

The third field of four bits of the S_PCI **shall** ⁽¹⁰⁾ be the "SOURCE SAP ID" and identifies the client of the transmitting peer which sent the data.

The fourth field of four bits **shall** ⁽¹¹⁾ be the "DESTINATION SAP ID" and identifies the client of the receiving peer which must take delivery of the data. There is no need to encode the source and destination node addresses in the S_PDU as this information is relayed between the peers by the underlying sublayers. The "DESTINATION SAP ID" **shall** ⁽¹²⁾ be equal in value to the corresponding argument of the S_UNIDATA_REQUEST or S_EXPEDITED_UNIDATA_REQUEST primitive submitted by the client

The fifth field of the S_PCI **shall** ⁽¹³⁾ be "CLIENT DELIVERY CONFIRM REQUIRED", and is encoded as a single bit that can take the values "YES" (=1) or "NO" (=0). The value of this bit **shall** ⁽¹⁴⁾ be set according to the *Service Type* requested by the sending client during binding (see S_BIND_REQUEST primitive) or according to the *Delivery Mode* requested explicitly for this U_PDU (see S_UNIDATA_REQUEST Primitive or S_EXPEDITED_UNIDATA_REQUEST Primitive).

The sixth field **shall** ⁽¹⁵⁾ be the VALID TTD field, and is encoded as a single bit that can take the values "YES" (=1) or "NO" (=0), indicating the presence of a valid TTD within the S_PCI.

The seventh field of the S_PCI **shall** ⁽¹⁴⁾ be two unused bits that are reserved for future use.

The eighth and last field of the S_PCI **shall** ⁽¹⁵⁾ be "TTD" and represents the TimeToDie for this U_PDU. The first four bits of this field **shall** ⁽¹⁶⁾ have meaning if and only if the VALID TTD field equals "YES", the remaining 16 bits of the field **shall** ⁽¹⁷⁾ be present in the S_PCI if and only if the VALID TTD field equals "YES".

The TTD field encodes the Julian date³ modulo 16, and the GMT in seconds after which time the S_PDU must be discarded if it has not yet been delivered to the client. The Julian date modulo 16 part of the TTD **shall** ⁽¹⁸⁾ be mapped into the first four bits of the TTD field (i.e., bits 0-3 of byte 2 of the S_PDU).

The 16 high bits of the GMT part of the TTD shall be mapped into the 2 remaining bytes of the TTD field; the LSB of the GMT shall be discarded. If the “VALID TTD” flag bit of a DATA S_PDU is set (=1) then the complete TTD 20-bit field is present and its value must be used. If this flag bit is not set (=0), the last two bytes of the TTD field are not present (to conserve overhead) and the TTD must not be used. The “VALID TTD” flag bit allows the transmitting peer to specify whether the receiving peer should discard the S_PDU by based on TTD or it delivered the U_PDU to the client without consideration of the TTD.

A.3.1.2 DATA DELIVERY CONFIRM S_PDU

Type :

“1” = DATA DELIVERY CONFIRM

Encoding :

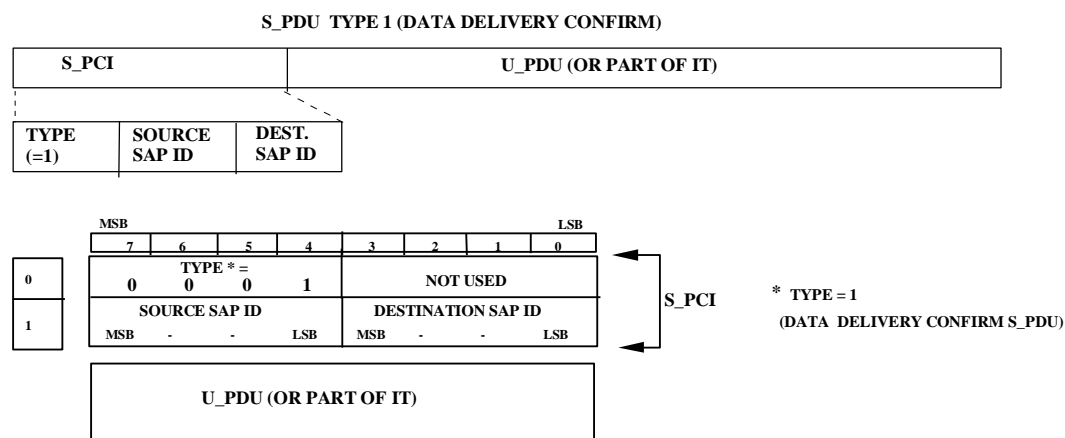


Figure A-33: Generic Encoding and Bit-Field Maps of the DATA DELIVERY CONFIRM S_PDU

Description :

The DATA DELIVERY CONFIRM S_PDU **shall** be ⁽¹⁾ transmitted in response to a successful delivery to a Client of a U_PDU which was received in a DATA type S_PDU in which the “CLIENT DELIVERY CONFIRM REQUIRED” field was set to “YES”. The DATA DELIVERY CONFIRM S_PDU **shall** be ⁽²⁾ transmitted by the Subnetwork Interface Sublayer to the peer sublayer which originated the DATA type S_PDU.

³ The simple Julian date system, which numbers the days of the year consecutively starting with 001 on 1 January and ending with 365 on 31 December (or 366 on leap years).

The first part of the DATA DELIVERY CONFIRM S_PDU **shall** ⁽³⁾ be the S_PCI, while the second part **shall** ⁽⁴⁾ be a full or partial copy of the U_PDU that was received and delivered to the destination Client.

The first field of the S_PCI part **shall** ⁽⁵⁾ be "TYPE" and its value **shall** ⁽⁶⁾ equal 1 to identify the S_PDU as being of type DATA DELIVERY CONFIRM.

The remaining fields and their values for the S_PCI part of the DATA DELIVERY CONFIRM S_PDU **shall** ⁽⁷⁾ be equal in value to the corresponding fields of the DATA S_PDU for which this DATA DELIVERY CONFIRM S_PDU is a response.

The peer sublayer that receives the DATA DELIVERY CONFIRM **shall** ⁽⁸⁾ inform the client which originated the U_PDU that its data has been successfully delivered to its Destination by issuing a S_UNIDATA_REQUEST_CONFIRM or a S_EXPEDITED_UNIDATA_REQUEST_CONFIRM in accordance with the data exchange protocol of Section A.3.2.4.

A.3.1.3 DATA DELIVERY FAIL S_PDU

Type :

"2" = DATA DELIVERY FAIL

Encoding :

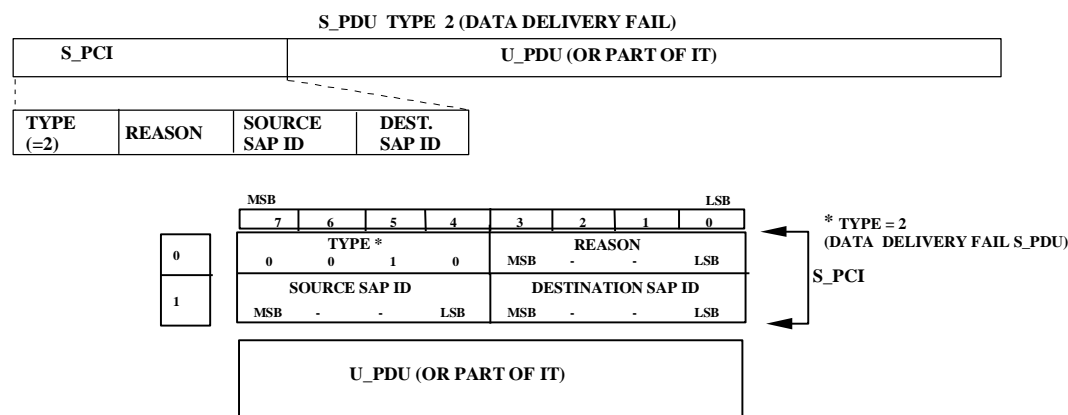


Figure A-34: Generic Encoding and Bit-Field Map of the DATA DELIVERY FAIL S_PDU

Description :

The DATA DELIVERY FAIL S_PDU **shall** ⁽¹⁾ be transmitted in response to a failed delivery to a Client of a U_PDU that was received in a DATA type S_PDU with the "CLIENT DELIVERY CONFIRM REQUIRED" field set to "YES".

The first part of this S_PDU **shall** ⁽²⁾ be the S_PCI.

The second part **shall** ⁽³⁾ be a full or partial copy of the U_PDU that was received but not delivered to the destination Client.

The first field of the S_PCI **shall** ⁽⁴⁾ be "TYPE". Its value **shall** ⁽⁵⁾ be equal to 2 and identifies the S_PDU as being of type DATA DELIVERY FAIL.

The second field **shall** ⁽⁶⁾ be "REASON" and explains why the U_PDU failed to be delivered. It can take a value in the range 0-15; valid reasons are defined in the table below.

| Reason | Value |
|--------------------------------|-------|
| <i>Unassigned and reserved</i> | 0 |
| Destination SAP ID not bound | 1 |
| <i>Unassigned and reserved</i> | 2-15 |

The SOURCE SAP_ID and DESTINATION SAP_ID fields of the S_PCI **shall** ⁽⁷⁾ be equal in value to the corresponding fields of the DATA S_PDU for which the DATA DELIVERY FAIL S_PDU is a response.

The peer sublayer that receives the DATA DELIVERY FAIL S_PDU, **shall** ⁽⁸⁾ inform the client which originated the U_PDU that its data was not delivered to the destination by issuing a S_UNIDATA_REQUEST_REJECTED primitive or a S_EXPEDITED_UNIDATA_REQUEST_REJECTED primitive, in accordance with the data exchange protocol of Section A.3.2.4.

A.3.1.4 HARD LINK ESTABLISHMENT REQUEST S_PDU

Type :

"3" = HARD LINK ESTABLISHMENT REQUEST

Encoding :

| TYPE (=3) | LINK TYPE | LINK PRIORITY | REQUESTING SAP ID | REMOTE SAP ID |
|--------------|--------------|------------------|----------------------|------------------|
|--------------|--------------|------------------|----------------------|------------------|

| MSB | | | | | LSB | | | | |
|-----|----------------------------|---|---|---|-----|----------------------------|---|---------------|---|
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | TYPE * | | | | | LINK TYPE | | LINK PRIORITY | |
| | 0 0 1 1 | | | | | | | | |
| 1 | REQUESTING SAP ID | | | | | REMOTE SAP ID** | | | |
| | MSB - - LSB | | | | | MSB - - LSB | | | |

* TYPE = 3
(HARD LINK ESTABLISHMENT REQUEST S_PDU)

** VALID ONLY IF LINK TYPE = 2

* TYPE = 3
(HARD LINK ESTABLISHMENT REQUEST S_PDU)

** VALID ONLY IF LINK TYPE = 2

Figure A-35: Generic Encoding and Bit-Field Map of the HARD LINK ESTABLISHMENT REQUEST S_PDU

Description :

The HARD LINK ESTABLISHMENT REQUEST S_PDU **shall** ⁽¹⁾ be transmitted by a Peer in response to a Client's request for a Hard Link. Since the establishment of a Hard Link overrides the normal procedures of making Links based on the destinations of the queued U_PDUs (i.e., over Soft Link Data Exchange), it is important that both peers use a handshake procedure in order to confirm the successful Hard Link establishment

The first field of the S_PDU **shall** ⁽²⁾ be "TYPE". It **shall** ⁽³⁾ be equal to 3 and identifies the S_PDU as being of type HARD LINK ESTABLISHMENT REQUEST.

The "LINK TYPE" and "LINK PRIORITY" fields **shall** ⁽⁴⁾ be equal in value to the corresponding arguments of the S_HARD_LINK_ESTABLISH Primitive submitted by the client to request the link.

The "REQUESTING SAP ID" field **shall** ⁽⁵⁾ be the SAP ID of the client that requested the Hard Link Establishment.

This "REMOTE SAP ID" field **shall** ⁽⁶⁾ be valid if and only if the "LINK TYPE" field has a value of 2, denoting a Type 2 Hard Link w/ Full-Bandwidth Reservation. The "REMOTE SAP ID" field **shall** ⁽⁷⁾ identify the single client connected to the remote node to and from which traffic is allowed for Hard Links w/ Full-Bandwidth Reservation. The REMOTE SAP ID field may take any implementation-dependent default value for Hard Links without Full Bandwidth Reservation.

A.3.1.5 HARD LINK ESTABLISHMENT CONFIRM S_PDU

Type :

"4" = HARD LINK ESTABLISHMENT CONFIRM

Encoding :

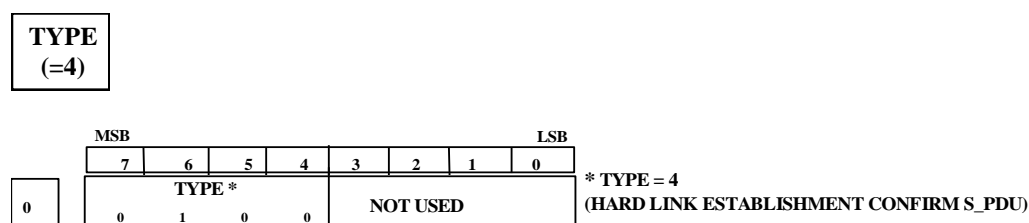


Figure A-36: Generic Encoding and Bit-Field Map of the HARD LINK ESTABLISHMENT CONFIRM S_PDU

Description :

The HARD LINK ESTABLISHMENT CONFIRM S_PDU **shall** ⁽¹⁾ be transmitted as a positive response to the reception of a HARD LINK ESTABLISHMENT REQUEST S_PDU.

Its only field **shall** ⁽²⁾ be “TYPE”, which value **shall** ⁽³⁾ be equal to 4 and identifies the S_PDU as being of type HARD LINK ESTABLISHMENT CONFIRM.

The peer which receives this S_PDU **shall** ⁽⁴⁾ inform its appropriate client accordingly with a S_HARD_LINK_ESTABLISHED Primitive in accordance with the Hard Link Establishment Protocol specified in Section A.3.2.2.2.

A.3.1.6 HARD LINK ESTABLISHMENT REJECTED S_PDU

Type :

“5” = HARD LINK ESTABLISHMENT REJECTED

Encoding :

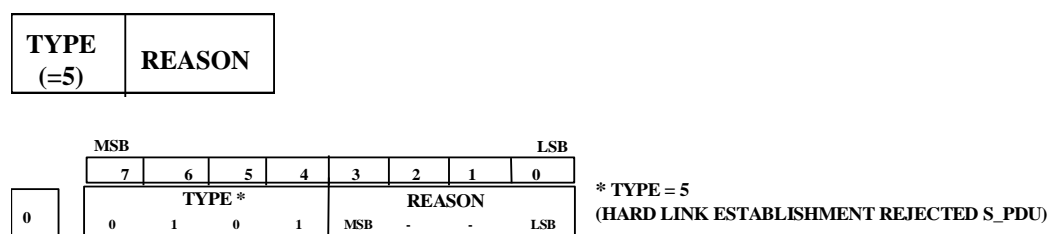


Figure A-37: Generic Encoding and Bit-Field Map of the HARD LINK ESTABLISHMENT REJECTED S_PDU

Description :

This S_PDU **shall** ⁽¹⁾ be transmitted as a negative response to the reception of a HARD LINK ESTABLISHMENT REQUEST S_PDU.

The first field **shall** ⁽²⁾ be “TYPE” and its value **shall** ⁽³⁾ be equal to 5 to identify the S_PDU as being of type HARD LINK ESTABLISHMENT REJECTED.

The second field **shall** ⁽⁴⁾ be “REASON” and explains why the Hard Link request was rejected. The sublayer that receives this S_PDU should inform its appropriate client accordingly with a S_HARD_LINK_REJECTED Primitive in accordance with the Hard Link Establishment Protocol specified in Section A.3.2.2.2. The “REASON” field **shall** ⁽⁵⁾ take a value in the range 0-15. Hard Link reject reasons and their corresponding values **shall** ⁽⁶⁾ be as defined in the following table.

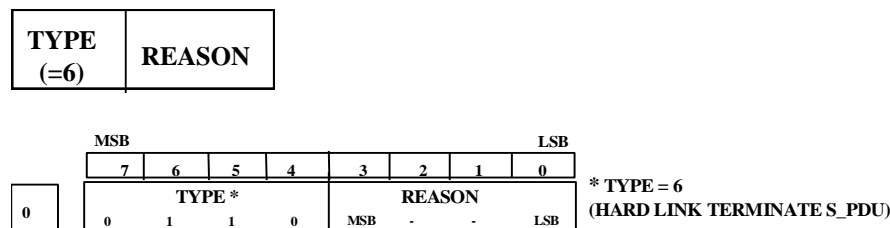
| REASON | Field Value |
|--------------------------------|-------------|
| <i>unassigned</i> | 0 |
| Remote-Node-Busy | 1 |
| Higher-Priority-Link-Existing | 2 |
| Remote-Node-Not-Responding | 3 |
| Destination SAP ID not bound | 4 |
| <i>Reserved for future use</i> | 5-15 |

A.3.1.7 HARD LINK TERMINATE S_PDU

Type :

“6” = HARD LINK TERMINATE

Encoding :



**Figure A-38: Generic Encoding and Bit-Field Map of the
HARD LINK TERMINATE S_PDU**

Description :

Under normal circumstances a Hard Link is terminated at the request of the Client which originated it or as a result of a request by another Client to establish a higher priority Hard Link. Either of the two Peer sublayers involved in a Hard Link session and that wishes to terminate the Hard Link **shall** ⁽¹⁾ transmit a HARD LINK TERMINATE S_PDU to request termination of the Hard Link.

The first four-bit field **shall** ⁽²⁾ be “TYPE” and its value **shall** ⁽³⁾ be set equal to 6 to identify the S_PDU as being of type HARD LINK TERMINATE.

The second four-bit field **shall** ⁽⁴⁾ be “REASON” and explains why the Hard Link is being terminated. Hard Link terminate reasons and their corresponding values **shall** ⁽⁵⁾ be as defined in the following table.

| Reason | Value |
|--------------------------------|-------|
| <i>unassigned</i> | 0 |
| Client request | 1 |
| Higher priority link requested | 2 |
| <i>reserved</i> | 3 |
| Destination SAP ID unbound | 4 |
| <i>Reserved for future use</i> | 5-15 |

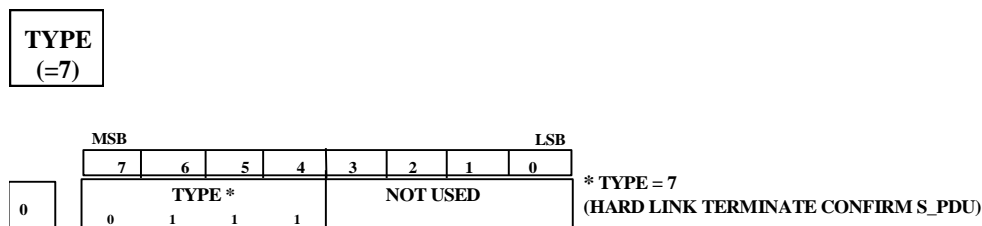
In order to ensure a graceful termination of the Hard Link, the Peer which sent the HARD LINK TERMINATE must await a TIMEOUT period for confirmation of its Peer before it declares the Link as terminated. This TIMEOUT period **shall** ⁽⁶⁾ be configurable by the protocol implementation.

A.3.1.7 HARD LINK TERMINATE CONFIRM S_PDU

Type :

“7” = HARD LINK TERMINATE CONFIRM

Encoding :



**Figure A-39: Generic Encoding and Bit-Field Map of the
HARD LINK TERMINATE CONFIRM S_PDU**

Description :

The HARD LINK TERMINATE CONFIRM S_PDU **shall** ⁽¹⁾ be transmitted in response to the reception of a HARD LINK TERMINATE S_PDU.

The first four-bit field of this S_PDU **shall** ⁽²⁾ be “TYPE”. A value of 7 **shall** ⁽³⁾ identify that the S_PDU is of type HARD LINK TERMINATE CONFIRM.

The second four-bit field of this S_PDU **shall** ⁽³⁾ be not used in this implementation of the protocol. The values of these bits may be implementation dependent.

A.3.2 Peer-to-Peer Communication Protocol

This section specifies the protocols governing the Peer-to-Peer communication for Establishing and Terminating Soft Link Data Exchange Sessions, Establishing and Terminating Hard Link Data Exchange Sessions, Establishing and Terminating Broadcast Data Exchange Sessions and Exchanging Client Data. In these specifications, the node whose local client or Subnetwork Interface Sublayer first requests a Data Exchange Session is denoted as the caller or calling node and the remote node is denoted as the called node.

A.3.2.1 Soft Link Data Exchange Session

In the absence of a hard link request by a client, the Subnetwork Interface Sublayer initiates Soft Link Data Exchange Sessions with remote peers based on the destinations of queued client U_PDUs. In particular, sublayer management algorithms must be established to initiate the protocols for establishment or termination a Soft Link Data Exchange Session. This STANAG allows these sublayer management algorithms to be based on implementation dependent criteria and factors. The use of comparative U_PDU queue-length for given clients, source-destination sets and priority levels for any implementation is allowed and expected (even if the algorithms are trivial) but remain beyond the scope of this STANAG.

A.3.2.1.1 Protocol for Establishing a Soft Link Data Exchange Session

In contrast with the establishment of a Hard Link Session, the establishment of Soft Link Data Exchange Sessions **shall** ⁽¹⁾ not require explicit peer-to-peer handshaking within the Subnetwork Interface Sublayer.

The calling peer **shall** ⁽²⁾ implicitly establish a Soft Link Data Exchange Session by requesting its Channel Access Sublayer to make a physical link to the required remote node, using the procedure for making physical links specified in Annex B. In accordance with these procedures, both peer Subnetwork Interface Sublayers (i.e., the calling and called sublayers) are informed about the successful making of a physical link between their nodes by their respective Channel Access Sublayers.

After the physical link is made, both peer Subnetwork Interface Sublayers **shall** ⁽³⁾ declare that the Soft Link Data Exchange Session has been established between the respective source and destination nodes. Data may then be exchanged in accordance with the protocols specified in Section A.3.2.4.

A.3.2.1.2 Protocol for Terminating a Soft Link Data Exchange Session

No peer-to-peer communication by the Subnetwork Interface Sublayer **shall** ⁽¹⁾ be required to terminate a Soft Link Data Exchange Session.

A Soft Link Data Exchange Session **shall** ⁽²⁾ be terminated by either of the two peers by a request to its respective Channel Access Sublayer to break the Physical Link in accordance with the procedure specified in Annex B. Both Subnetwork Interface sublayers will be informed about the breaking of the Physical link by their respective Channel Access Sublayers.

Since a called peer can terminate a Soft Link Data Exchange Session if it has higher priority data destined for a different Node, called peers **shall** ⁽³⁾ wait a configurable minimum time before unilaterally terminating sessions, to prevent unstable operation of the protocol.

Note: The caller sublayer normally initiates the termination of the session (by breaking the physical link) based on the destinations of its queued U_PDUs, and on any ongoing communication with the distant node. The inter-layer signaling for coordination would normally be carried out via the subnetwork management sublayer. The called sublayer can also terminate the session if it has high priority data destined for a different node. However, called sublayers should wait a configurable minimum time before unilaterally terminating sessions, otherwise an unstable condition may arise if all nodes in the network have data to transmit and called sublayers immediately close sessions in order to establish other sessions as callers. If such a situation arises, the efficiency of a subnetwork will deteriorate as a result of nodes continuously establishing and terminating sessions without actually transmitting data. The minimum amount of time that a called sublayer should wait before it attempts to terminate a Soft Link Session must be carefully chosen and will depend on a number of factors such as the subnetwork size and configuration. Specification of this and other parameters as a configurable but required value allows implementations of the STANAG to be tuned for specific network, with the values for these parameters distributed as part of the standard operating procedures for a given network.

After the Subnetwork Interface Sublayer has been notified that the Physical Link has been broken, the Subnetwork Interface Sublayer **shall** ⁽⁴⁾ declare the Soft Link Exchange Session as terminated.

A.3.2.2 Hard Link Data Exchange Session

The rules governing establishment and termination of Hard Links are straightforward but complicated by the fact that new Hard Link requests could be satisfied (at least in part) by an existing Hard Link. Comparison and evaluation factors for Hard Links include the ranks of the clients, the link priorities, the sets of source and destination nodes, and the Hard-Link types. In particular, various service models could be specified in this STANAG for the management of multiple Hard Links of Types 0 or Type 1 simultaneously, but with different levels of protocol complexity to track the potentially overlapping and independent requests from multiple clients. Note that, since Type 2 Hard Links reserve the full bandwidth and use of a physical link for two specific bound clients, the subnetwork cannot support multiple Type 2 Hard Links with any assumed service model.

This STANAG assumes a simple model for the management of Hard Links based on maintenance of at most a single Hard Link between nodes, while still allowing Type 0 and Type 1 Hard Links between the nodes to be shared by other clients using Soft Link Data Exchange. This management model satisfies the following requirements:

- a node's sublayer **shall** ⁽¹⁾ maintain at most one Hard Link at any time;
- a sublayer **shall** ⁽²⁾ accept a Hard Link request when no Hard Link currently exists;
- the comparative precedence of new requests and any existing hard link **shall** ⁽³⁾ be evaluated in accordance with section A.3.2.2.1 to determine if the new request can be accepted or rejected by the sublayer;
- requests of higher precedence **shall** ⁽⁴⁾ be accepted and will result in the termination of an existing Hard Link;
- an existing Hard Link of higher precedence **shall** ⁽⁵⁾ result in the rejection of the request;
- if an existing Type 0 Hard Link can satisfy a request that has been rejected, the sublayer **shall** ⁽⁶⁾ note this as the reason for rejecting the request.; the requesting client may then submit data for transmission using a Soft Link Data Exchange Session.

[Note: While this model has some limitations, notably that clients sharing a Hard Link with the originator will lose it when the originator terminates the link, it supports the essential service characteristics desired, and presents a simpler protocol than others that were considered.]

Unless noted otherwise, any data structures and variables used to manage Hard Link establishment and termination are implementation dependent and beyond the scope of this STANAG.

Further requirements controlling the establishment and termination of Hard Links are specified below.

A.3.2.2.1 Priority and Precedence Rules for Hard Links

Establishment and Termination of Hard Links **shall** ⁽¹⁾ be controlled in accordance with the following set of precedence rules:

- a) A Hard Link request for a client with greater rank **shall** ⁽²⁾ take precedence over an existing or requested Hard Link established for a client of lower rank, regardless of other factors.
- b) A Hard Link request of greater priority **shall** ⁽³⁾ take precedence over an existing or requested hard link of lower priority, regardless of other factors.

c) For Hard Links of equal priority and rank, and with different sets of source and destination nodes, the Hard Link request processed first by the Subnetwork Interface Sublayer (i.e., the Hard Link currently established) **shall** ⁽⁴⁾ take precedence.

d) For Hard Links (i.e., requests and existing hard links) from clients of equal priority and rank, and with equal sets of source and destination nodes:

- a Hard Link with greater Link Type value **shall** ⁽⁵⁾ take precedence over one with lower value;
- an existing Hard Link **shall** ⁽⁶⁾ take precedence over subsequent Hard Link requests of equal Link Type.

A.3.2.2.2 Protocol for Establishing a Hard Link Data Exchange Session

Upon receiving a S_HARD_LINK_ESTABLISH Primitive from a client, the Subnetwork Interface Sublayer **shall** ⁽¹⁾ first check that it can accept the request from the client in accordance with the precedence and priority rules of Section A.3.2.2.1.

If the Hard Link request is of lower precedence than any existing Hard Link, then the establishment protocol proceeds as follows:

- the request **shall** ⁽²⁾ be denied by the Subnetwork Interface Sublayer,
- the sublayer **shall** ⁽³⁾ issue a S_HARD_LINK_REJECTED Primitive to the requesting client with REASON = "Higher-Priority-Link-Existing", and
- the sublayer **shall** ⁽⁴⁾ terminate the Hard Link establishment protocol.

Otherwise, if a Type 0 Hard Link request is of the same priority, same client-rank, and with the same set of source and destination nodes as an existing Hard Link, then the establishment protocol proceeds as follows:

- the Subnetwork Interface Sublayer **shall** ⁽⁵⁾ reject the Type 0 Hard Link request with the REASON = "Requested Type 0 Hard Link Exists"; a client receiving this rejection may submit data requests for transmission using a Soft-Link Data Exchange Session to the remote peer;
- the sublayer **shall** ⁽⁶⁾ take no further action to establish or change the status of the existing Type 0 Hard Link (Note: since the sublayer has already determined that the existing link satisfies the requirements of the request), and; the sublayer **shall** ⁽⁷⁾ terminate the Hard Link establishment protocol.

Otherwise, the establishment protocol proceeds in accordance with the following requirements.

If the Subnetwork Interface Sublayer can accept the Hard Link request it **shall** ⁽⁸⁾ first terminate any existing Hard Link of lower precedence using the peer-to-peer communication protocol for terminating an existing hard link specified in Section A.3.2.2.3.

The Subnetwork Interface Sublayer then **shall** ⁽⁹⁾ request the Channel Access Sublayer to make a physical link to the node specified by the client, following procedure for making the physical link specified in Annex B.

After the physical link has been made, the caller's Subnetwork Interface Sublayer **shall** ⁽¹⁰⁾ send a "HARD LINK ESTABLISHMENT REQUEST" (TYPE 3) S_PDU to its called peer at the remote node. To ensure that the S_PDU will overtake all routine DATA S_PDUs which may be queued and in various stages of processing by the lower sublayers, the "HARD LINK ESTABLISHMENT REQUEST" S_PDU **shall** ⁽¹¹⁾ be sent to the Channel Access Sublayer using a C_EXPEDITED_UNIDATA_REQUEST Primitive and use the subnetwork's expedited data service.

After it sends the "HARD LINK ESTABLISHMENT REQUEST" (TYPE 3) S_PDU, the caller's Subnetwork Interface Sublayer **shall** ⁽¹²⁾ wait a configurable time-out period for a response from the called peer, and proceed as follows:

- during the waiting-period for the response,
 - if the caller's sublayer receives a HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU from the called peer, the sublayer **shall** ⁽¹³⁾ notify the requesting client that the Hard Link request has failed by sending the client an S_HARD_LINK_REJECTED Primitive to the requesting client with the REASON field of the Primitive set to the corresponding value received in the S_PDU's REASON field,
 - if the caller's sublayer receives a HARD LINK ESTABLISHMENT CONFIRM" (TYPE 4) S_PDU from the called peer, the sublayer **shall** ⁽¹⁴⁾ notify the requesting client that the Hard Link request has succeeded by sending the client an S_HARD_LINK_ESTABLISHED Primitive;
- otherwise, if the waiting-period for the response expires without receipt of a valid response from called node, the caller's sublayer **shall** ⁽¹⁵⁾ notify the requesting client that the Hard Link request has failed by sending the client an S_HARD_LINK_REJECTED Primitive to the requesting client with REASON = "Remote-Node-Not Responding".

The caller's establishment protocol **shall** ⁽¹⁶⁾ terminate on receipt during the waiting of a valid response from the called node and notification of the client, or on expiration of the waiting period.

For the called Subnetwork Access Sublayer, the Hard Link establishment protocol **shall** ⁽¹⁷⁾ be initiated on receipt of a "HARD LINK ESTABLISHMENT REQUEST" (TYPE 3) S_PDU, and proceeds as follows:

- if no client is bound to the called SAP ID and the caller's request is for a Type 2 Hard Link, then the called sublayer **shall** ⁽¹⁸⁾ reject the request, send a "HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU with REASON = "Destination SAP ID not bound" to the caller, and terminate the establishment protocol;
- otherwise, the called sublayer **shall** ⁽¹⁹⁾ evaluate the precedence of the caller's request in accordance with the precedence and priority rules of Section A.3.2.2.1, using as the client

rank either a configurable default rank for the called SAP_ID for Type 0 and Type 1 Hard Link requests, or the actual rank of the bound client with the called SAP_ID for Type 2 Hard Link requests.

- If the caller's request cannot be accepted by the called peer, a "HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU **shall** ⁽²¹⁾ be sent to the calling peer, with the Reason field set as follows:
 - REASON = "Remote-Node-Busy" if the reason for rejection was the existence of an existing Hard Link of equal rank and priority, or,
 - REASON= "Higher-Priority Link Existing" if the reason for rejection was the existing of a Hard Link with higher priority or rank.
- If the caller's Hard Link request can be accepted and the request is not a Type 2 Hard Link request, the called sublayer **shall** ⁽²²⁾ send a "HARD LINK ESTABLISHMENT CONFIRM" (TYPE 4) S_PDU to the caller sublayer, and terminate the protocol;
- otherwise, the request is for a Type 2 Hard Link and the called sublayer **shall** ⁽²³⁾ send a S_HARD_LINK_INDICATION Primitive to the requested client, and wait for a configurable maximum time-out period for a response:
 - if the called sublayer receives a S_HARD_LINK_ACCEPT Primitive from the requested client prior to the expiration of the timeout, then the called sublayer **shall** ⁽²⁴⁾ send a "HARD LINK ESTABLISHMENT CONFIRM" (TYPE 4) S_PDU to the calling sublayer, and terminate the protocol;
 - otherwise, the called sublayer **shall** ⁽²⁴⁾ send a "HARD LINK ESTABLISHMENT REJECTED" (TYPE 5) S_PDU to the caller sublayer, and terminate the protocol.
- Whenever sent, either the TYPE 4 (HARD LINK ESTABLISHMENT CONFIRM) S_PDU or the TYPE 5 (HARD LINK ESTABLISHMENT REJECTED) S_PDU **shall** ⁽²⁵⁾ be sent to the calling sublayer using the Expedited Data Service provided by lower sublayers in the profile.

The procedures for establishing a hard link from the perspective of both the calling and called peers are depicted in Figure A-40(a) and Figure A-40(b) as a possible implementation that meets the stated requirements. This STANAG acknowledges that other implementations may exist that also meet the stated requirements.

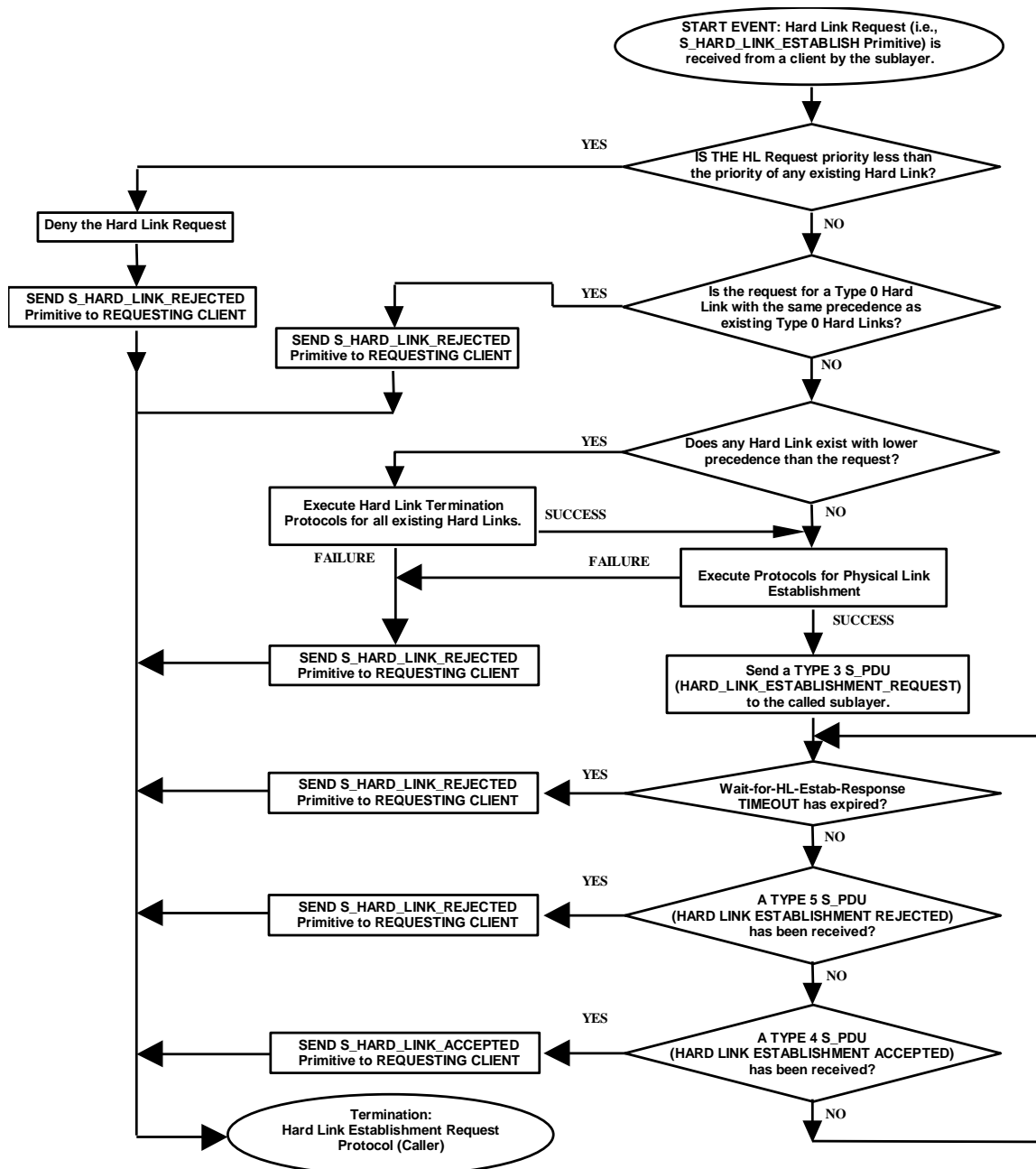


Figure A-40 (a): Procedures for Establishing a Hard Link: CALLER PEER

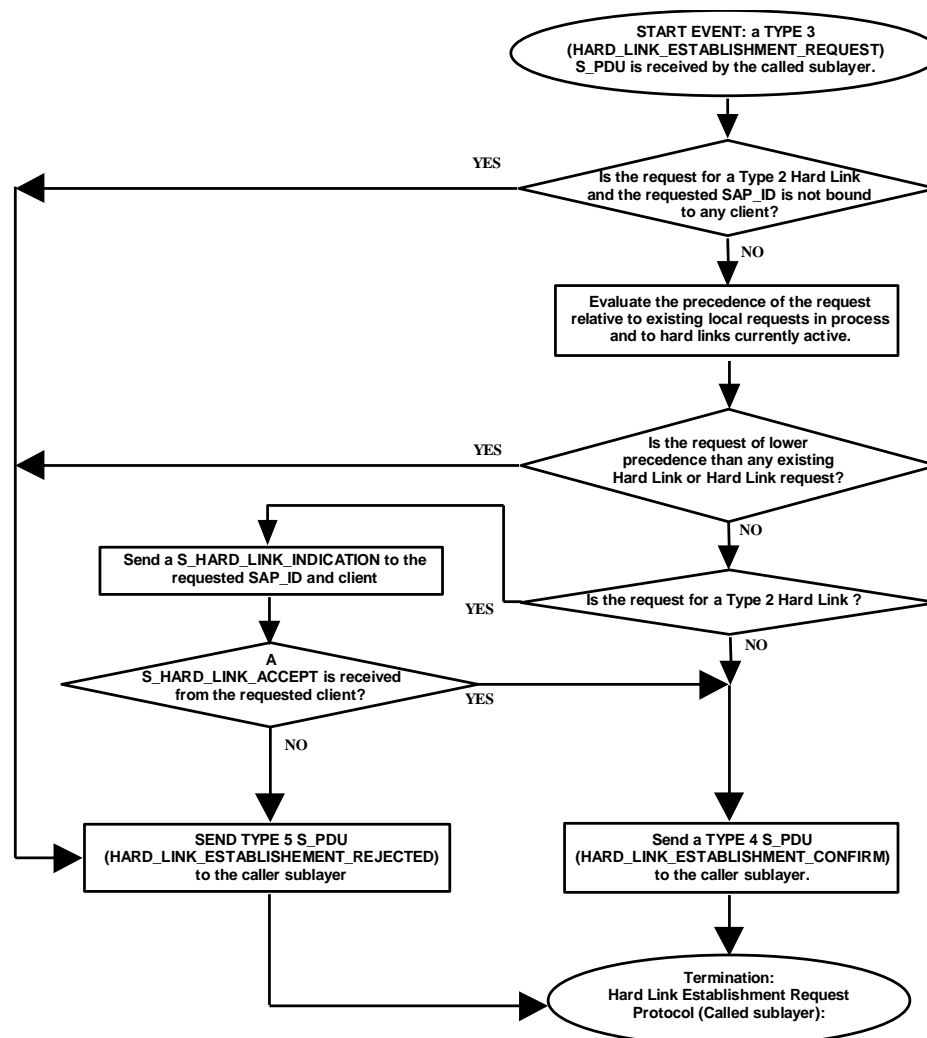


Figure A-40 (b): Procedures for Establishing a Hard Link: CALLED PEER

A.3.2.2.3 Protocol for Terminating a Hard Link Data Exchange Session

The termination of an existing Hard Link can be initiated by either of the two peer sublayers connected by the link. Normally the Hard Link will be terminated by the calling sublayer at the request of the client who initiated it, or by either of the sublayers if it receives a Hard Link request of higher precedence from one of its other clients. Requirements for the Hard Link termination protocol are defined below.

The Hard Link termination protocol **shall** ⁽¹⁾ be initiated when any of the following conditions are met:

- a calling sublayer receives a S_HARD_LINK_TERMINATE Primitive from the client that originated an existing hard link of any type,
- a called sublayer receives a S_HARD_LINK_TERMINATE Primitive from its attached client involved in an existing Type 2 Hard Link,

- either the calling or called sublayer receives from a client a S_HARD_LINK_ESTABLISH Primitive that is of higher precedence than any existing Hard Link.

Any sublayer that must terminate a Hard Link for any of the specified conditions **shall** ⁽²⁾ send a "HARD LINK TERMINATE" (TYPE 6) S_PDU to its peer sublayer.

A sublayer that receives a "HARD LINK TERMINATE" (TYPE 6) S_PDU from its peer **shall** ⁽³⁾ immediately respond with a "HARD LINK TERMINATE CONFIRM" (TYPE 7) S_PDU, declare the Hard Link as terminated, and send a S_HARD_LINK_TERMINATED Primitive to all clients using the Hard Link.

After sending the "HARD LINK TERMINATE" (TYPE 6) S_PDU, the initiating sublayer **shall** ⁽⁴⁾ wait for a response for a configurable maximum time-out period, and proceed.

If the timeout-period expires without receipt by the initiating sublayer of a "HARD LINK TERMINATE CONFIRM" (TYPE 7) S_PDU, the sublayer **shall** ⁽⁵⁾ send a S_HARD_LINK_TERMINATED Primitive to all clients using the Hard Link, with the REASON field set equal to "Remote Node Not Responding (timeout)".

To ensure that any S_PDU used for the termination protocol will overtake all routine DATA S_PDUs that may be queued and in various stages of processing by the lower sublayers, the termination protocol uses the subnetwork's Expedited Data Service. In particular, the "HARD LINK TERMINATE" (TYPE 6) S_PDU and "HARD LINK TERMINATE CONFIRM" (TYPE 7) S_PDU **shall** ⁽⁶⁾ be sent to the Channel Access Sublayer using a C_EXPEDITED_UNIDATA_REQUEST Primitive.

After termination of the Hard Link with a subnetwork client, the Physical Link between the nodes may need to be broken. Normally the breaking of the Physical Link is left to the peer which requested the termination of the Hard Link session. The reason for this is that this peer may want to start another session using the existing Physical Link in which case breaking and making procedures may be avoided. The procedures for breaking a Physical Link are specified in Annex B.

The nominal procedures for Terminating a Hard Link for both the Requesting and Responding Peers are shown in Figure A-41. This STANAG acknowledges that other implementations may exist that meet the requirements stated above.

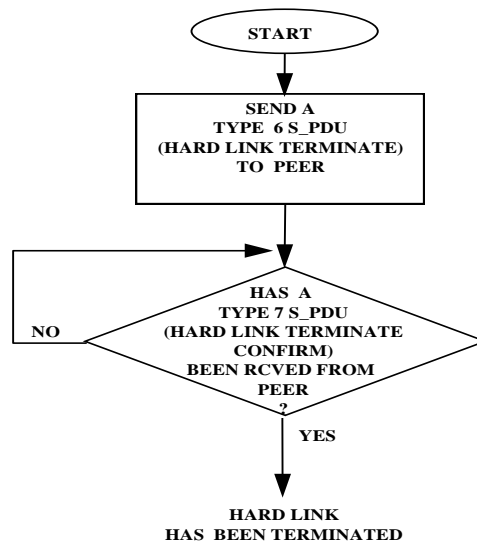


Figure A-41 (a): Procedures for Terminating a Hard Link: REQUESTING PEER

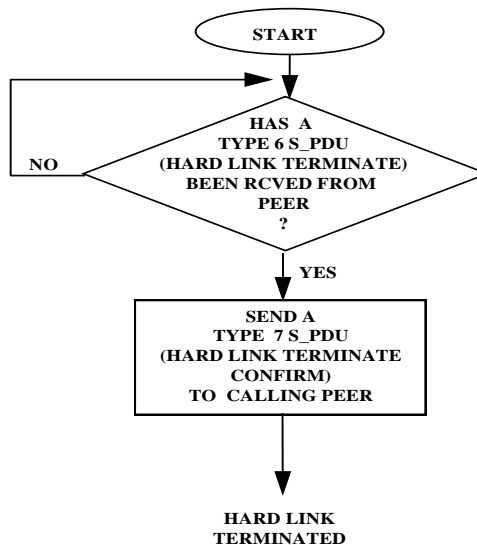


Figure A-42 (b): Procedures for Terminating a Hard Link: RESPONDING PEER

Apart from the procedures above, a sublayer **shall** ⁽⁷⁾ unilaterally declare a Hard Link as terminated if at any time it is informed by the Channel Access Sublayer that the physical link has been permanently broken. In this case, the sublayer **shall** ⁽⁸⁾ send a S_HARD_LINK_TERMINATED Primitive to all clients using the Hard Link, with the REASON field set equal to "Physical Link Broken".

A.3.2.3 Protocol for Establishing and Terminating a Broadcast Data Exchange Session

No explicit peer-to-peer communication **shall** ⁽¹⁾ be required to establish and terminate a Broadcast Data Exchange Session. A Broadcast Data Exchange Session is established and terminated either by a management process or unilaterally by the Subnetwork Interface Sublayer based on a number of criteria as explained in section A.1.1.3.

As noted in section A.1.1, clients may interleave requests for data-exchange sessions. At some point, the subnetwork might also be configured to provide exclusive support for a Broadcast Data Exchange Session. In this case, when the subnetwork is first configured by the local (implementation-dependent) management function to provide exclusive support for a Broadcast Data Exchange Session the Subnetwork Interface Sublayer **shall** ⁽²⁾ send an S_UNBIND_INDICATION to any bound clients that had requested ARQ Delivery Service, with the REASON = "ARQ Mode Unsupportable during Broadcast Session". Subsequent S_BIND requests by clients requesting ARQ service (soft-link or hard-link sessions) **shall** ⁽³⁾ be rejected with the same reason.

A.3.2.4 Protocol for Exchanging Client Data

After a Data Exchange Session of any type has been established, sublayers with client data to exchange **shall** ⁽¹⁾ exchange DATA (TYPE 0) S_PDUs using the protocol specified below and in accordance with the service characteristics of the respective session.

The sublayer **shall** ⁽²⁾ discard any U_PDU submitted by a client where the U_PDU is greater in size than the Maximum Transmission Unit (MTU) size assigned to the client by the S_BIND_ACCEPTED Primitive issued during the client-bind protocol.

If a U_PDU is discarded because it exceeded the MTU size limit and if the DELIVERY CONFIRMATION field for the U_PDU specifies CLIENT DELIVERY CONFIRM or NODE DELIVERY CONFIRM, the sublayer **shall** ⁽³⁾ notify the client that submitted the U_PDU as follows:

- if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive the sublayer **shall** ⁽⁴⁾ send a S_UNIDATA_REQUEST_REJECTED Primitive to the client;
- otherwise, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive, the sublayer **shall** ⁽⁵⁾ send a S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive to the client;
- for either form of the reject primitive, the REASON field **shall** ⁽⁶⁾ be equal to "U_PDU Larger than MTU".

For U_PDUs that have been accepted for transmission, the sending sublayer retrieves client U_PDUs and their associated implementation-dependent service attributes (such as the S_Primitive that encapsulated the U_PDU) from its queues (according to Priority and other implementation-dependent criteria), and proceeds as follows:

- the sending sublayer **shall** ⁽⁷⁾ encode the retrieved U_PDU into a DATA (TYPE 0) S_PDU, transferring any service attributes associated with U_PDU to the S_PDU as required;
- the sending sublayer **shall** ⁽⁸⁾ encode the resulting DATA (TYPE 0) S_PDU in accordance with the C_Primitive interface requirements of the Channel Access Sublayer as specified in Annex B, i.e.,:
 - if the encoded U_PDU was submitted by a client using a S_UNIDATA_REQUEST Primitive, then the sublayer **shall** ⁽⁹⁾ encode the S_PDU as a C_UNIDATA_REQUEST Primitive of the priority corresponding to that initially specified by the client in the S_Primitive, otherwise;

- if the encoded U_PDU was submitted by a client using a S_EXPEDITED_UNIDATA_REQUEST Primitive, then the sublayer **shall** ⁽¹⁰⁾ encode the S_PDU as a C_EXPEDITED_UNIDATA_REQUEST Primitive;
- the sending sublayer then **shall** ⁽¹¹⁾ pass the resulting C_primitive to the Channel Access Sublayer for further processing to send the DATA (TYPE 0) S_PDU to its remote peer.
- if the service attributes for the U_PDU require NODE DELIVERY CONFIRMATION, the sublayer **shall** ⁽¹²⁾ wait for a configurable time for a response as follows:
 - if the sublayer receives a C_UNIDATA_REQUEST_CONFIRM Primitive or a C_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive prior to the end of the waiting time, the sublayer **shall** ⁽¹³⁾ send to the client either a S_UNIDATA_REQUEST_CONFIRM Primitive or S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive, respectively, where the type of C_Primitive expected and S_Primitive sent corresponds to the type of U_PDU delivery service requested;
 - otherwise, if the sublayer receives a C_UNIDATA_REQUEST_REJECTED Primitive or a C_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive prior to the end of the waiting time, the sublayer **shall** ⁽¹⁴⁾ send to the client a either a S_UNIDATA_REQUEST_REJECTED Primitive or S_EXPEDITED_UNIDATA_REQUEST_REJECTED, respectively, where the type of C_Primitive expected and S_Primitive sent corresponds to the type of U_PDU delivery service requested;
 - otherwise, if the waiting time ends prior to receipt of any response indication from the Channel Access sublayer, the Subnetwork Interface sublayer **shall** ⁽¹⁵⁾ | send to the client either a S_UNIDATA_REQUEST_REJECTED Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive, or a | S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive; for either reject S_Primitive, the REASON field shall be set equal to "Destination Node Not Responding".
- if the service attributes for the U_PDU require CLIENT DELIVERY CONFIRMATION, the sending sublayer **shall** ⁽¹⁶⁾ wait for a configurable time for a response as follows:
 - if the Subnetwork Interface sublayer receives a C_Primitive confirming node-node delivery (i.e., either a C_UNIDATA_REQUEST_CONFIRM Primitive or a C_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive) and a "DATA DELIVERY CONFIRM" (TYPE 1) S_PDU is received from the remote sublayer prior to the end of the waiting time, the Subnetwork Interface sublayer **shall** ⁽¹⁷⁾ send to the client either a S_UNIDATA_REQUEST_CONFIRM Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive, or a S_EXPEDITED_UNIDATA_REQUEST_CONFIRM Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive;

- otherwise, if the Subnetwork Interface sublayer receives either a “reject” C_Primitive from the Channel Access Sublayer or a “DATA DELIVERY FAIL” (TYPE 2) S_PDU from the remote peer prior to the end of the waiting time, the Subnetwork Interface sublayer **shall** ⁽¹⁸⁾ send to the client either a S_UNIDATA_REQUEST_REJECTED Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive or a S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive; for either form of the primitive, the REASON field **shall** ^(18.1) be derived from the “DATA DELIVERY FAIL” (TYPE 2) S_PDU or the reject C_Primitive that was received;
- otherwise, if the waiting time ends prior to receipt of a response message, the sublayer **shall** ⁽¹⁹⁾ send to the client either a S_UNIDATA_REQUEST_REJECTED Primitive, if the U_PDU was submitted by a S_UNIDATA_REQUEST Primitive, or a S_EXPEDITED_UNIDATA_REQUEST_REJECTED Primitive, if the U_PDU was submitted by a S_EXPEDITED_UNIDATA_REQUEST Primitive; for either Primitive, the REASON field shall be set equal to “Destination Node Not Responding”.
- On completion of these actions by the sending sublayer the client data delivery protocol terminates for the given DATA (TYPE 0) S_PDU.

A receiving sublayer manages the client data exchange protocol as follows:

- the receiving sublayer **shall** ⁽²⁰⁾ accept encoded DATA (TYPE 0) S_PDUs from the Channel Access Sublayer using C_Primitives in accordance with the interface requirements specified in Annex B.
[Note: in accordance with the interface between the Subnetwork Interface and Channel Access sublayers, there is no explicit indication that the S_PDU is a “normal” or an “expedited” one. Whether the S_PDU is a “normal” or an “expedited” S_PDU is determined by whether the S_PDU is encoded within a C_UNIDATA_INDICATION Primitives or a C_EXPEDITED_UNIDATA_INDICATION Primitives, respectively.]
- the receiving sublayer **shall** ⁽²¹⁾ extract the U_PDU, Destination SAP_ID and the other associated service attributes from the DATA (TYPE 0) S_PDUs as required;
- if there is no client bound to the destination SAP_ID, the receiving sublayer **shall** ⁽²²⁾ discard the U_PDU by; otherwise,
 - if the DATA (TYPE 0) S_PDU was encoded within a C_UNIDATA_INDICATION Primitive, the sublayer **shall** ⁽²³⁾ deliver the extracted U_PDU to the destination client bound to Destination SAP_ID using a S_UNIDATA_INDICATION Primitive;
 - if the DATA (TYPE 0) S_PDU was encoded within a C_EXPEDITED_UNIDATA_INDICATION Primitive, the sublayer **shall** ⁽²⁴⁾ deliver the extracted U_PDU to the destination client bound to Destination SAP_ID using a S_EXPEDITED_UNIDATA_INDICATION Primitive.

- if the received S_PDU has the “CLIENT DELIVERY CONFIRM REQUIRED” field set equal to “YES”, then the sublayer **shall** ⁽²⁵⁾ provide delivery confirmation as follows:
 -
 - if a client was bound to the Destination SAP_ID, the sublayer **shall** ⁽²⁶⁾ encode as required and send a “DATA DELIVERY CONFIRM” (TYPE 1) S_PDU to the sending sublayer; [Note: implementation-dependent methods may be used to provide additional determination that the client data was successfully delivered prior to sending the “DATA DELIVERY CONFIRM” (TYPE 1) S_PDU.],
 - if a client was not bound to the Destination SAP_ID, the sublayer **shall** ⁽²⁷⁾ encode as required and send a “DATA DELIVERY FAIL” (TYPE 2) S_PDU to the sending sublayer. [Note: implementation-dependent methods may be used to provide additional determination that the client data was unsuccessfully delivered | prior to sending the “DATA DELIVERY FAIL” (TYPE 2) S_PDU.]
- On completion of these actions by the receiving sublayer the client data delivery protocol terminates for the given DATA (TYPE 0) S_PDU.

Implementation-dependent queuing disciplines, flow-control procedures, or other characteristics in the sublayer **shall** ⁽²⁸⁾ not preclude the possibility of managing the data exchange protocol for more than one U_PDU at a time. In particular, the Subnetwork Interface Sublayer **shall** ⁽²⁹⁾ be capable of sending a U_PDU, encapsulated in a DATA (TYPE 0) S_PDU and C_Primitive as required, to the Channel Access Sublayer prior to receipt of the data-delivery-confirm response for a U_PDU sent earlier.

[Note: This requirement mitigates the reduction in link throughput that occurs when a subnetwork ceases transmission of any U_PDUs while it awaits confirmation of their delivery. The performance degradation is typical of that which occurs when using a STOP-AND-WAIT form of ARQ protocol anywhere in a communication system.]

The nominal procedures for exchanging DATA S_PDUs for both the Sending and Receiving Peers are shown in Figure A-43(a) and Figure A-43(b). This STANAG acknowledges that other implementations may satisfy the requirements stated above. It should be noted that, as shown in these figures, there is no explicit indication that the S_PDU is a “normal” or an “expedited” one. The reason for this is that the underlying sublayers are expected to treat Expedited S_PDUs differently and implicitly pass the information to the receiving peer by (for example) delivering Expedited S_PDUs as C_EXPEDITED_UNIDATA_INDICATION Primitives rather than normal C_UNIDATA_INDICATION Primitives.

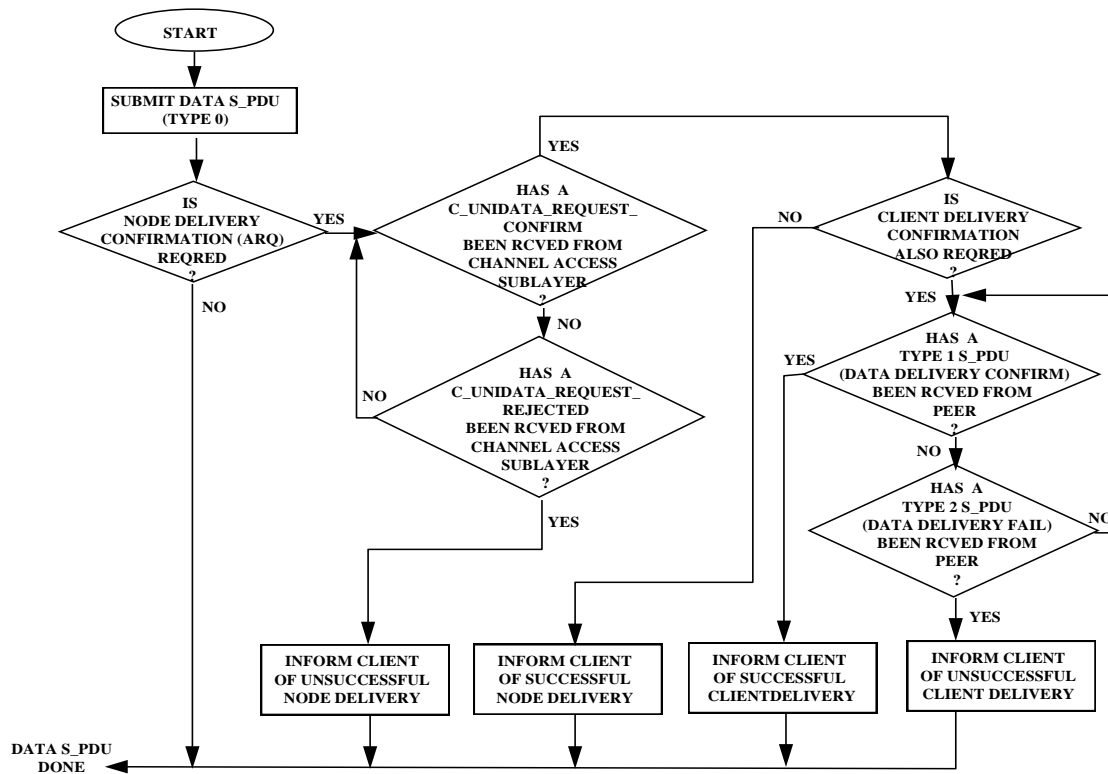


Figure A-43 (a): Data Exchange Procedures: SENDING PEER

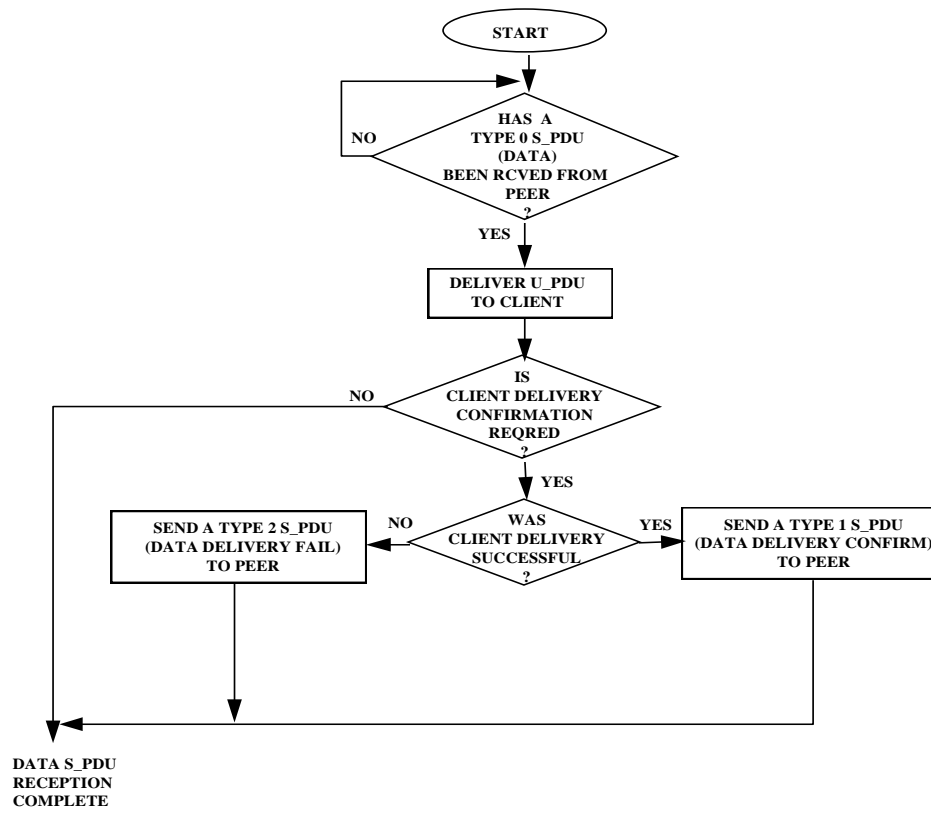


Figure A-43 (b): Data Exchange Procedures: RECEIVING PEER

Left blank intentionally

Annex B: Channel Access Sublayer (mandatory)

The functions required of the channel access sublayer are quite limited in the HF subnetwork.

B.1 Channel Access Sublayer Service Definition

The Channel Access Sublayer provides services to the Subnetwork Interface Sublayer. These services are:

1. Execute requests by the Subnetwork Interface sublayer to “*Make*” and “*Break*” Physical Links
2. Notify the Subnetwork Interface sublayer of changes in the state of a Physical Link.
3. Accept S_PDUs (encapsulated in the appropriate primitive) from the Subnetwork Interface sublayer for transmission on a Physical Link.
4. Deliver S_PDUs (encapsulated in the appropriate primitive) received on a Physical Link to the Subnetwork Interface sublayer.

In order to provide these services, the Channel Access Sublayer implements a protocol that specifies the tasks that must be executed and the rules that must be obeyed by the sublayer. While a number of different channel-access protocols are possible, the one that is suitable for this document is referred to as the Type 1 protocol, and described herein.

B.2 Interface Primitives Exchanged with the Subnetwork Interface Sublayer

The implementation of the interface between the Channel Access Sublayer and the Subnetwork Interface sublayer is not mandated or specified by this STANAG. Since the interface is internal to the subnetwork architecture and may be implemented in a number of ways it is considered beyond the scope of STANAG 5066. A model of the interface has been assumed, however, for the purposes of discussion and specification of other sublayer functions. The STANAG’s presentation of the interface is modelled after that used in Reference 1 [1] to this STANAG, however, the interface defined in the reference is not mandated for use herein.

Despite the advisory nature of the conceptual model of the internal interface between the Subnetwork Interface sublayer and the Channel Access Sublayer, there are some mandatory requirements that are placed on any interface implementation.

The interface must support the service-definition for the Channel Access Sublayer, i.e.:

1. the interface **shall** ⁽¹⁾ enable the Subnetwork Interface sublayer to submit requests to change the state of a physical link, i.e., to make or break a physical link of a specified type (i.e., Exclusive or Nonexclusive, as specified in Section B.____) with a specified node address;
2. the interface **shall** ⁽²⁾ enable the Channel Access sublayer to notify the Subnetwork Interface sublayer of changes in the status of the physical link;

¹ Clark, D., and N. Karavassillis, “Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio”, TM-937, May 1998, pp. 72-80

3. The interface **shall** ⁽³⁾ allow the Channel Access sublayer to accept S_PDUs from the Subnetwork Interface sublayer
4. The interface **shall** ⁽⁴⁾ allow the Channel Access sublayer to deliver S_PDUs to the Subnetwork Interface sublayer.
5. Since S_PDUs have no explicit indication as to whether or not they use Expedited or Normal Data Delivery Service in the subnetwork, the interface **shall** ⁽⁵⁾ indicate the type of delivery service required by or given to the S_PDUs exchanged across the interface.

Additionally, the protocol-control information from the Subnetwork Interface sublayer that is required for the management of the Channel Access sublayer **shall** ⁽⁶⁾ not be derived from knowledge of the contents or format of any client data or U_PDUs encapsulated within the S_PDUs exchanged over the interface.

[Note: user's that encrypt their traffic prior to submittal may use the subnetwork. Subnetwork operation must be possible with client data in arbitrary formats that are unknown to the subnetwork, therefore any service requirements or indications must be provided by interface control information provided explicitly with the user data.]

The interface may use knowledge of the contents of S_PDUs (excluding the contents of any encapsulated U_PDUs) to derive protocol control information for the Channel Access sublayer. This approach is highly discouraged, however. The recommended approach for implementation is that information required for protocol control within the Channel Access sublayer should be provided explicitly in appropriate interface primitives.

In keeping with accepted practice in the definition of layered protocols, and as a means for specifying the operations of the sublayers that are mandated by this STANAG, the communication between the Channel Access Sublayer and the Subnetwork Interface sublayer is described herein with respect to a set of Primitives. The interface Primitives are a set of messages for communication and control of the interface and service requests made between the two layers.

By analogy to the design of the client-subnetwork interface, the technical specification of the Channel Access Sublayer assumes communication with the Subnetwork Interface sublayer using primitives prefixed with a "C_". A minimal set of C_Primitives has been assumed that meet the requirements stated above and the general function for each C_Primitive is given in Table B-1. These C_Primitives are given without benefit of a list of arguments or detailed description of their use. As noted initially, they are offered for information only as a means of describing the interaction between the Channel Access sublayer and the Subnetwork Interface sublayer, and as general guidance for a possible implementation.

Table B-1- Nominal Definition of C_Primitives for the Interface between the Channel Access sublayer and the Subnetwork Interface sublayer (non-mandatory, for information-only)

| NAME OF PRIMITIVE | DIRECTION (see Note) | COMMENTS |
|--------------------------------------|----------------------|---|
| C_PHYSICAL_LINK_MAKE | SIS → CAS | request to make a physical link of a specified type to a specified node |
| C_PHYSICAL_LINK_MADE | CAS → SIS | report that a physical link was made |
| C_PHYSICAL_LINK_REJECTED | CAS → SIS | report that a physical link could not be made and that the initiating request is rejected |
| C_PHYSICAL_LINK_BREAK | SIS → CAS | request to break a physical link |
| C_PHYSICAL_LINK_BROKEN | CAS → SIS | report that a physical has been broken (by request, or unilaterally) |
| C_CHANNEL_AVAILABILITY | CAS → SIS | reports on the availability of a given channel and the underlying subnetwork |
| C_UNIDATA_REQUEST | SIS → CAS | delivers an S_PDU to the Channel Access Sublayer, requesting normal data-delivery service |
| C_UNIDATA_REQUEST_CONFIRM | CAS → SIS | confirms S_PDU delivery to the remote node using the normal data-delivery service |
| C_UNIDATA_REQUEST_REJECTED | CAS → SIS | notifies that an S_PDU could not be delivered to a remote node using the normal data-delivery service |
| C_UNIDATA_INDICATION | CAS → SIS | delivers an S_PDU that had been received using the normal delivery service to the Subnetwork Interface sublayer |
| C_EXPEDITED_UNIDATA_REQUEST | SIS → CAS | delivers an S_PDU to the Channel Access Sublayer, requesting normal data-delivery service |
| C_EXPEDITED_UNIDATA_REQUEST_CONFIRM | CAS → SIS | confirms S_PDU delivery to the remote node using the normal data-delivery service |
| C_EXPEDITED_UNIDATA_REQUEST_REJECTED | CAS → SIS | notifies that an S_PDU could not be delivered to a remote node using the normal data-delivery service |
| C_EXPEDITED_UNIDATA_INDICATION | CAS → SIS | delivers an S_PDU that had been received using the normal delivery service to the Subnetwork Interface sublayer |

[Note: SIS = Subnetwork Interface Sublayer; CAS = Channel Access Sublayer;
→ = direction of message flow from source to destination sublayer]

B.3 Channel Access Protocol Type 1 and C_PDUs

The Type 1 Channel Access Protocol **shall** ⁽¹⁾ support the following subnetwork configuration:

1. Pairs of Nodes **shall** ⁽²⁾ be linked "point-to-point" on a "common" HF frequency channel or on dedicated frequency channels selected from a pool of assigned frequencies by an external process² which is not under the control of any of the sublayers.
(Note: an ALE sublayer is not present or not used in STANAG 5066).
- 1a. The co-ordination of the making and breaking of Physical Links (hereinafter referred to as 'CAS 1 Linking Protocol') between two nodes (after a common frequency has already been selected by an external process) **shall** ⁽³⁾ be performed solely by the Channel Access Sublayer. If it can be reasonably assured that both nodes in a pair of nodes linked "point-to-point" on a common frequency have already entered the "connected" state by virtue of their becoming linked on a common frequency and that they will exit the "connected" state when they are no longer linked on a common frequency, then the CAS 1 Linking Protocol **may** be omitted. If the CAS 1 Linking Protocol is omitted and the response to ARQ-data DPDU is a Warning DPDU indicating that a connection is not made, then the CAS 1 Linking Protocol **shall** ^(3.1) be followed and the data resent.
If a slave node (receiving station) links on a common frequency through an external process (e.g., an ALE protocol) and a CAS-1 Physical-Link Request C_PDU is received, then the receiving node **shall** ^(3.2) respond in accordance with the requirements of the 'CAS 1 Linking Protocol', accepting or rejecting the request as is appropriate for its state.
Physical links established by a process external to the CAS 1 linking protocol **shall** ^(3.3) be broken when the common frequency condition between the linked nodes no longer exists. Conversely, the common frequency condition between two nodes **shall** ^(3.4) be terminated when either node declares its logical physical link to be broken. In these cases, the physical link **shall** ^(3.5) be broken without using the CAS 1 linking protocol.
Physical links established by a process external to the CAS 1 linking protocol **shall** ^(3.6) respond to the reception of a type 4 PHYSICAL_LINK_BREAK C_PDU by sending a type 5 PHYSICAL_LINK_BREAK_CONFIRM C_PDU as specified in Section B.3.2.2. The external process that created the common frequency condition **shall** ^(3.7) terminate that condition and the physical link **shall** ^(3.8) be broken at that point.
It is **strongly recommended** to use the CAS 1 linking protocol if three or more active nodes are on the same frequency. Any node omitting CAS 1 that detects a third node on the same channel **should** enable CAS 1.
[Note: It is possible for this case to occur in the presence of an external frequency coordination protocol (e.g., "NET", "ALL", or "ANY" calls in an ALE protocol). In this case, it is beneficial to arbitrate physical link establishment by using appropriate C_PDU transfer.]
2. Physical Links **shall** ⁽⁴⁾ be of either of two types, Exclusive or Nonexclusive, with properties and service features defined as follows:
 - a) a Node **shall** ⁽⁵⁾ use an Exclusive Physical Link to support control and data exchange for Hard Link Data Exchange Sessions as requested by the Subnetwork Interface Sublayer;

² An appropriate frequency may be simply selected by an external (semi-)automated process or manually by an operator e.g. "available" frequencies are negotiated between a node which needs to establish a link and a master station which "manages" the assigned frequency pool. Protocol type 1 provides no mechanism for automatic interaction between the external process and the Channel Access Sublayer. The Channel Access Sublayer "assumes" that a common frequency channel has already been found.

- b) A Node **shall** ⁽⁶⁾ use a Nonexclusive Physical Link to support control and data exchange for Soft-Link Data Exchange Sessions as requested by the Subnetwork Interface Sublayer.

[Note: the Channel Access sublayer is not explicitly aware of the existence of the Hard-Link or Soft-Link types of the Subnetwork Interface sublayer, nor need it be aware. In practice, and as specified in STANAG 5066 Annex A, the Subnetwork Interface Sublayer will request the making of an Exclusive Physical Link whenever it wishes to establish and use a Hard Link Data Exchange Session. Likewise, the Subnetwork Interface Sublayer will establish only Nonexclusive Physical Links to support Soft-Link Data Exchange Sessions.]

- c) A Node **shall** ⁽⁷⁾ establish at most two Exclusive Physical Links with other nodes.

[Note: This requirement is *not* contradictory, though it may seem so with the nomenclature chosen. A node must accept a request for an Exclusive Physical Link in accordance with the requirements of Section B.3, even if another Exclusive Physical Link is active. The reason is that the newly requested link is required to support peer-to-peer communication at the higher layers where negotiation to establish Hard Link Data Exchange Sessions takes place. Priorities and ranks of clients are not manifest at the Channel Access Sublayer, but only at the Subnetwork Interface Sublayer. Therefore, a new request for an Exclusive Physical Link must be accepted long enough to allow the Subnetwork Interface sublayers of the local and remote nodes to establish the respective priorities of the existing and requested links. Thus, in practice, at most two Exclusive Physical Links can be active at any time, one to support an existing Hard-Link Data Exchange, and the other to support the peer-to-peer communication at the Subnetwork Interface sublayer for an incoming Hard Link Request from a remote node. Since the newly requested Exclusive Physical Link is for a Hard-Link Data Exchange Session, the Subnetwork Interface Sublayer will determine the respective priorities of the active and newly requested links. The Subnetwork Interface Sublayer will either terminate the active link or reject the request, whichever is the loser, and force closure of the corresponding Exclusive Physical Link.]

- d) A Node may have Nonexclusive Physical Links with more than one other node at a time, one Nonexclusive Physical Link per remote node; i.e., a Node may “*Make*” a new Nonexclusive Physical Link with another node before it “*Breaks*” any Nonexclusive Physical links.

4. There **shall** ⁽⁸⁾ be no explicit peer-to-peer communication required to switch from use of one Nonexclusive Physical Link to another.

[Note: A node uses an active Physical Link merely by exchanging data on the link with the other node. With respect to the Channel Access sublayer, this requires no explicit management; lower sublayers, however, may require explicit coordination to switch from use of one link to another based on the destination address of S_PDUs submitted by the Subnetwork Interface Sublayer.]

5. The Nodes may or may not be within ground-wave distances or be able to receive the transmissions of other nodes sharing the channel.

The Type 1 Channel Access Sublayer **shall** ⁽⁹⁾ communicate with peer sublayers in other nodes using the protocols defined here in order to:

1. Make and break physical links
2. Deliver S_PDUs between Subnetwork Interface Sublayers at the local node and remote node(s).

B.3.1 Type 1 Channel Access Sublayer Data Protocol Units (C_PDUs)

The following C_PDUs **shall** ⁽¹⁾ be used for peer-to-peer communication between Channel Access Sublayers in the local and remote node(s):

| <i>C_PDU NAME</i> | <i>Type Code</i> |
|-----------------------------|------------------|
| DATA C_PDU | TYPE 0 |
| PHYSICAL LINK REQUEST | TYPE 1 |
| PHYSICAL LINK ACCEPTED | TYPE 2 |
| PHYSICAL LINK REJECTED | TYPE 3 |
| PHYSICAL LINK BREAK | TYPE 4 |
| PHYSICAL LINK BREAK CONFIRM | TYPE 5 |

The first argument and encoded field of all C_PDUs **shall** ⁽²⁾ be the C_PDU Type

The remaining format and content of these C_PDUs **shall** ⁽³⁾ be as specified in the subsections that follow.

Unless noted otherwise, argument values encoded in the C_PDU bit-fields **shall** ⁽⁴⁾ be mapped into the fields in accordance with CCITT V.42, 8.1.2.3, i.e.:

1. when a field is contained within a single octet, the lowest bit number of the field **shall** ⁽⁵⁾ represent the lowest-order (i.e., least-significant-bit) value;
2. when a field spans more than one octet, the order of bit values within each octet **shall** ⁽⁶⁾ decrease progressively as the octet number increases. The lowest bit number associated with the field **shall** ⁽⁷⁾ represent the lowest-order (i.e., least significant bit) value.

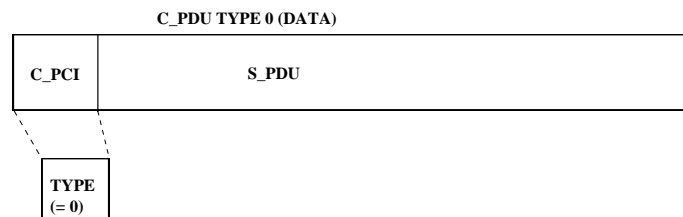
Unless noted otherwise, bit-fields specified as NOT USED **shall** ⁽⁸⁾ be encoded with the value '0' (i.e., zero).

B.3.1.1 DATA C_PDU

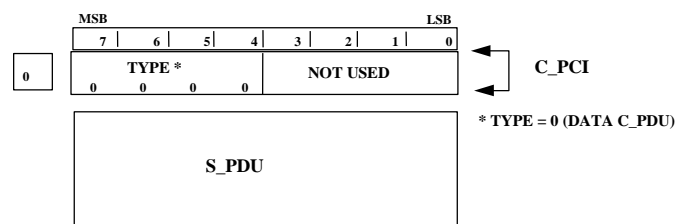
Type :

“0” = DATA C_PDU

Encoding :



(a) Generic Encoding



(b) Bit-Field Map

Figure B-1. Generic Encoding and Bit-Field Map of the DATA C_PDU

Description :

The DATA (TYPE 0) C_PDU **shall** ⁽¹⁾ be used to send an encapsulated S_PDU from the local node to a remote node.

The *Type* argument **shall** ⁽²⁾ be encoded in the first four-bit field of the DATA C_PDU as shown in Figure B-1. The value of the *Type* argument for the DATA C_PDU **shall** ⁽³⁾ be zero.

The remaining octets of the DATA C_PDU **shall** ⁽⁴⁾ contain the encapsulated S_PDU and only the encapsulated S_PDU.

For the Channel Access sublayer request to the lower layers of the subnetwork to deliver a C_PDU, the delivery service requirements for a DATA C_PDU **shall** ⁽⁵⁾ be the same as the S_PDU that it contains, i.e.:

- C_PDUs for which the Subnetwork Interface sublayer requested Expedited Data Delivery service for the encapsulated S_PDU **shall** ⁽⁶⁾ be sent using the Expedited Data Delivery service provided by the lower sublayer, otherwise,
- C_PDUs for which the Subnetwork Interface sublayer requested the normal Data Delivery service for the encapsulated S_PDU **shall** ⁽⁷⁾ be sent using the normal Data Delivery service provided by the lower sublayer;
- the DELIVERY mode specified by the Subnetwork Interface Sublayer for the encapsulated S_PDU (i.e., ARQ, non-ARQ, etc.) also **shall** ⁽⁷⁾ be assigned to the C_PDU by the Channel Access sublayer as the delivery mode to be provided by the lower sublayer.

B.3.1.2 PHYSICAL LINK REQUEST C_PDU

Type :

“1” = PHYSICAL LINK REQUEST

Encoding :

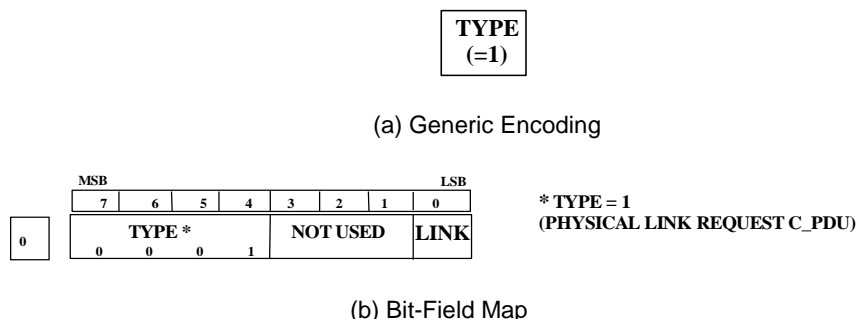


Figure B-2. Generic Encoding and Bit-Field Map of the PHYSICAL LINK REQUEST C_PDU

Description :

The PHYSICAL LINK REQUEST C_PDU **shall** ⁽¹⁾ be transmitted by a Channel Access sublayer to request the making of the Physical Link.

The PHYSICAL LINK REQUEST C_PDU **shall** ⁽²⁾ consist of the arguments *Type* and *Link*.

The value of the *Type* argument for the PHYSICAL LINK REQUEST C_PDU **shall** ⁽³⁾ be ‘1’ (i.e., one), encoded as a four-bit field as shown in Figure B-2.

The three bits not used in the encoding of the PHYSICAL LINK REQUEST C_PDU **shall** ⁽⁴⁾ be reserved for future use and not used by any implementation.

The value of the *Link* argument for the PHYSICAL LINK REQUEST C_PDU **shall** ⁽⁵⁾ be encoded as a one-bit field as shown in Figure B-2, with values as follows:

- a request for a Nonexclusive Physical Link **shall** ⁽⁶⁾ be encoded with the value ‘0’ (i.e., zero);
- a request for an Exclusive Physical Link **shall** ⁽⁷⁾ be encoded with the value ‘1’ (i.e., one).

When a PHYSICAL LINK REQUEST C_PDU is transmitted the local Node is not linked to another Node and therefore the ARQ transmission mode is not supportable by the Data Transfer Sublayer. Therefore, the PHYSICAL LINK REQUEST C_PDU **shall** ⁽⁸⁾ be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

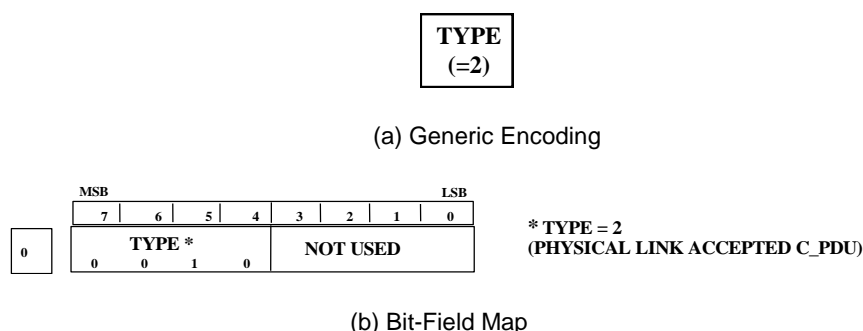
A Channel Access sublayer which receives a PHYSICAL LINK REQUEST C_PDU **shall** ⁽⁹⁾ respond with either a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU or a PHYSICAL LINK REJECTED (TYPE 3) C_PDU, as appropriate.

B.3.1.3 PHYSICAL LINK ACCEPTED C_PDU

Type :

“2” = PHYSICAL LINK ACCEPTED

Encoding :



**Figure B-3. Generic Encoding and Bit-Field Map of the
PHYSICAL LINK ACCEPTED C_PDU**

Description :

The PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU **shall** ⁽¹⁾ be transmitted by a peer sublayer as a positive response to the reception of a TYPE 1 C_PDU (PHYSICAL LINK REQUEST).

PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU **shall** ⁽²⁾ consist only of the argument *Type*.

[Note: A Node can never request the establishment of more than one Physical Link with another given node, therefore, the request for which this message is a response is uniquely identified by the node address of the node to which the PHYSICAL LINK ACCEPTED C_PDU is sent.]

The *Type* argument **shall** ⁽³⁾ be encoded as a four-bit field containing the binary value ‘two’ as shown in Figure B-3.

When a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU is transmitted the local Node is not linked to another Node and therefore the ARQ transmission mode is not supportable by the Data Transfer Sublayer. Therefore, the PHYSICAL LINK ACCEPTED C_PDU **shall** ⁽⁴⁾ be sent by the Channel Access Sublayer requesting the lower-layer’s Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

B.3.1.4 PHYSICAL LINK REJECTED C_PDU

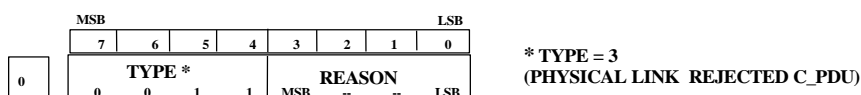
Type :

“3” = PHYSICAL LINK REJECTED

Encoding :

| | |
|---------------------|---------------|
| TYPE (=3) | REASON |
|---------------------|---------------|

(a) Generic Encoding



(b) Bit-Field Map

Figure B-4. Generic Encoding and Bit-Field Map of the PHYSICAL LINK REJECTED C_PDU

Description :

The PHYSICAL LINK REJECTED (TYPE 3) C_PDU **shall** ⁽¹⁾ be transmitted by a peer sublayer as a negative response to the reception of a TYPE 1 C_PDU (PHYSICAL LINK REQUEST).

The PHYSICAL LINK REJECTED (TYPE 3) C_PDU **shall** ⁽²⁾ consist of two arguments: *Type* and *Reason*.

The *Type* argument **shall** ⁽³⁾ be encoded as a four-bit field containing the binary value ‘three’ as shown in Figure B-4.

The *Reason* argument **shall** ⁽⁴⁾ be encoded in accordance with Figure B-4 and the following table³:

| Reason | Value |
|--------------------------------------|-------|
| Reason Unknown | 0 |
| Broadcast-Only-Node | 1 |
| Higher-Priority-Link-Request-Pending | 2 |
| Supporting Exclusive-Link | 3 |
| <i>unspecified</i> | 4-15 |

The value 0, indicating an unknown reason for rejecting the Make request for the physical link, is always a valid reason for rejection. Reasons corresponding to values in the range 4-15 are currently unspecified and unused in the STANAG.

When a PHYSICAL LINK REJECTED C_PDU is transmitted the local Node is not linked to another Node and therefore the ARQ transmission mode is not supportable by the Data

³ “Link busy” is not included in the list of possible reasons for rejecting anode’s request to establish a link since this function is provided by the WARNING frame type of the Data Transfer Sublayer, as specified in Annex C.

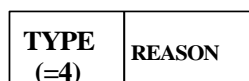
Transfer Sublayer. Therefore, the PHYSICAL LINK REJECTED C_PDU **shall** ⁽⁵⁾ be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

B.3.1.5 PHYSICAL LINK BREAK C_PDU

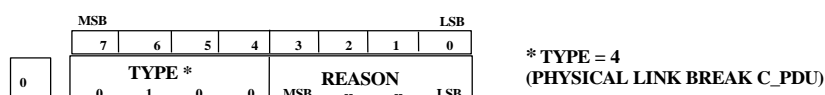
Type :

“4” = PHYSICAL LINK BREAK

Encoding :



(a) Generic Encoding



(b) Bit-Field Map

Figure B-5. Generic Encoding and Bit-Field Map of the PHYSICAL LINK BREAK C_PDU

Description :

The PHYSICAL LINK BREAK C_PDU **shall** ⁽¹⁾ be transmitted by either of the peer Channel Access sublayers involved in an active Physical Link to request the breaking of the link.

The PHYSICAL LINK BREAK C_PDU **shall** ⁽²⁾ consists of two arguments: *Type* and *Reason*.

The *Type* argument **shall** ⁽³⁾ be encoded as a four-bit field containing the binary value ‘four’ as shown in Figure B-5.

The *Reason* argument **shall** ⁽⁴⁾ be encoded in accordance with Figure B-5 and the following table:

| Reason | Value |
|--|-------|
| Reason Unknown | 0 |
| Higher-Layer-Request | 1 |
| Switching to Broadcast-Data-Exchange Session | 2 |
| Higher-Priority-Link-Request-Pending | 3 |
| No More Data | 4 |
| <i>unspecified</i> | 4-15 |

The value 0, indicating an unknown reason for requesting the breaking of a physical link, is always a valid reason. Reasons corresponding to values in the range 4-15 currently are not defined in this STANAG.

A peer sublayer which receives the PHYSICAL LINK BREAK C_PDU **shall** ⁽⁵⁾ immediately declare the Physical Link as broken and respond with a PHYSICAL LINK BREAK (TYPE 5) C_PDU as specified in section B.3.1.6 below.

The PHYSICAL LINK BREAK C_PDU **shall** ⁽⁶⁾ be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.
[Note: The reason the PHYSICAL LINK BREAK C_PDU is sent with the non-ARQ data service, even though a physical link exists at the time that it is sent, is because the receiving peer will immediately declare the Link as broken. The receiving peer will therefore not have time to send ARQ acknowledgements for the C_PDU.]

B.3.1.6 PHYSICAL LINK BREAK CONFIRM C_PDU

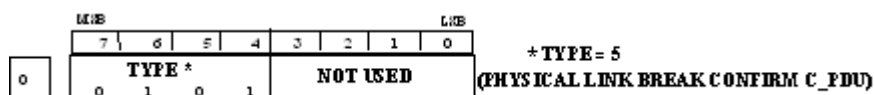
Type :

“5” = PHYSICAL LINK BREAK CONFIRM

Encoding :

| |
|--------------|
| TYPE (=5) |
|--------------|

(a) Generic Encoding



(b) Bit-Field Map

Figure B-6. Generic Encoding and Bit-Field Map of the PHYSICAL LINK BREAK CONFIRM C_PDU

Description :

The PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU **shall** ⁽¹⁾ be transmitted by a Channel Access sublayer as a response to a TYPE 4 “PHYSICAL LINK BREAK” C_PDU.

The PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU **shall** ⁽²⁾ consist of the single argument *Type*.

The *Type* argument **shall** ⁽³⁾ be encoded as a four-bit field containing the binary value ‘five’ as shown in Figure B-6.

The PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU **shall** ⁽⁴⁾ be sent by the Channel Access Sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs) in accordance with the Annex C specification of the Data Transfer Sublayer.

Upon receiving a PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU, the peer which initiated the breaking of the Link **shall** ⁽⁵⁾ declare the Link as broken.

B.3.2 Type 1 Channel Access Sublayer Peer-to-Peer Communication Protocol

Requirements on the Type 1 Channel Access Sublayer Peer-to-Peer Communication Protocols are specified in this section and the subsections below.

The Channel Access sublayer **shall** ⁽¹⁾ perform all peer-to-peer communications for protocol control using the following C_PDUs:

| <i>C_PDUs for Peer-to-Peer Protocol Control</i> | <i>Type Code</i> |
|---|------------------|
| PHYSICAL LINK REQUEST | TYPE 1 |
| PHYSICAL LINK ACCEPTED | TYPE 2 |
| PHYSICAL LINK REJECTED | TYPE 3 |
| PHYSICAL LINK BREAK | TYPE 4 |
| PHYSICAL LINK BREAK CONFIRM | TYPE 5 |

All peer-to-peer communications **shall** ⁽²⁾ be done by the Channel Access sublayer requesting the lower-layer Expedited Non-ARQ Data Service (i.e., transmission using Type 8 D_PDUs in accordance with the Annex C specification of the Data Transfer Sublayer).

The criteria for accepting or rejecting Physical Links Requests **shall** ⁽³⁾ be as follows:

1. A request to establish a Nonexclusive Physical Link **shall** ⁽⁴⁾ be accepted if there are no Exclusive Hard Links (active or with requests pending), and as long as the resulting total number of Nonexclusive Physical Links does not exceed the maximum number of Nonexclusive Physical Links allowed by the current subnetwork configuration.
2. At most one new request to establish an Exclusive Physical Link **shall** ⁽⁵⁾ be accepted as so long as the resulting number of active Exclusive Physical Links is no greater than two.
[Note: As noted earlier in this STANAG, a node must accept a request for an Exclusive Physical Link is, even if another Exclusive Physical Link is active, to support peer-to-peer communication at the higher layers where negotiation to establish Hard Link Data Exchange Sessions takes place. A new request for an Exclusive Physical Link must be accepted long enough to allow the Subnetwork Interface sublayers of the local and remote nodes to establish the respective priorities of any existing and requested links]

B.3.2.1 Protocol for Making a Physical Link

In the specification that follows of the protocol for making a physical link, the node which requests the physical link is referred to as the Caller or Calling node, while the node which receives the request will be referred to as the Called node.

The protocol for making the physical link **shall** ⁽¹⁾ consist of the following steps:

Step 1-Caller:

- a) The Calling Node's Channel Access Sublayer **shall** ⁽²⁾ send a PHYSICAL LINK REQUEST (TYPE 1) C_PDU to initiate the protocol, with the request's *Link* argument set equal to the type of link (i.e., Exclusive or Nonexclusive) requested by the higher sublayer,
- b) Upon sending the PHYSICAL LINK REQUEST (TYPE 1) C_PDU the Channel Access Sublayer **shall** ⁽³⁾ start a timer which is set to a value greater than or equal to the maximum time required by the Called Node to send its response (this time depends on the modem parameters, number of re-transmissions, etc.),
- c) The maximum time to wait for a response to a PHYSICAL LINK REQUEST (TYPE 1) C_PDU **shall** ⁽⁴⁾ be a configurable parameter in the implementation of the protocol.

Step 2-Called:

- a) On receiving a PHYSICAL LINK REQUEST (TYPE 1) C_PDU, a Called node **shall** ⁽⁵⁾ determine whether or not it can accept or reject the request as follows:
 - (1) if the request is for a Nonexclusive Physical Link and the Called node has an active Exclusive Physical Link, the Called node **shall** ⁽⁶⁾ reject the request,
 - (2) otherwise, if the request is for a Nonexclusive Physical Link and the Called node has the maximum number of active Nonexclusive Physical Links, the Called node **shall** ⁽⁷⁾ reject the request,
 - (3) otherwise, the Called node **shall** ⁽⁸⁾ accept the request for a Nonexclusive Physical Link.
 - (4) If the request is for an Exclusive Physical Link and the Called node has either one or no active Exclusive Physical Link, the Called node **shall** ⁽⁹⁾ accept the request,
 - (5) Otherwise, the Called node **shall** ⁽¹⁰⁾ reject the request for an Exclusive Physical Link.
- b) After determining if it can accept or reject the PHYSICAL LINK REQUEST, a Called node **shall** ⁽¹¹⁾ respond as follows:
 - (1) if a Called node accepts the physical link request, it **shall** ⁽¹²⁾ respond with a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU,
 - (2) otherwise, when a Called node rejects the physical link request, it **shall** ⁽¹³⁾ respond with a PHYSICAL LINK REJECTED (TYPE 3) C_PDU.
- b) After a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU is sent, the called channel access sublayer **shall** ⁽¹⁴⁾ declare the physical link made and transition to a state in which it executes the protocol for data exchange using C_PDUs.
- c) If further PHYSICAL LINK REQUEST (TYPE 1) C_PDUs are received from the same address after the link is made, the Channel Access sublayer **shall** ⁽¹⁵⁾ again reply with a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU.
- d) If at least one DATA (TYPE 0) C_PDU is not received on a newly activated Physical Link after waiting for a specified maximum period of time, the Called Node **shall** ⁽¹⁶⁾ abort the Physical Link and declare it inactive.
- e) The maximum period of time to wait for the first DATA (TYPE 0) C_PDU before aborting a newly activated Physical Link **shall** ⁽¹⁷⁾ be a configurable parameter in the implementation.

Step 3. Caller

There are two possible outcomes to the protocol for making a physical link: success or failure:

- a) Upon receiving a PHYSICAL LINK ACCEPTED (TYPE 2) C_PDU, the calling Channel Access Sublayer **shall**⁽¹⁸⁾ proceed as follows:
 - (1) if the request made an Exclusive Physical Link, the Calling node **shall**⁽¹⁹⁾ break any existing Nonexclusive Physical Links.
 - (2) the Calling node **shall**⁽²⁰⁾ declare the Physical Link as *successfully Made*, otherwise,
- b) upon receiving a PHYSICAL LINK REJECTED (TYPE 3) C_PDU, the Channel Access Sublayer **shall**⁽²¹⁾ declare the Physical Link as *Failed*, otherwise,
- c) upon expiration of its timer without any response having been received from the remote node, the Channel Access Sublayer **shall**⁽²²⁾ repeat Step 1 (i.e. send a PHYSICAL LINK REQUEST (TYPE 1) C_PDU and set a response time) and await again a response from the remote node.

The maximum number of times the Caller sends the PHYSICAL LINK REQUEST (TYPE 1) C_PDU without a response from the called node of either kind **shall**⁽²³⁾ be a configurable parameter in the implementation of the protocol.

After having repeated Step 1 the configurable maximum number of times without any response having been received from the remote node, the Caller's Channel Access Sublayer **shall**⁽²⁴⁾ declare the protocol to make the physical link as *Failed*.

| Figure B-7 (a) and (b) show the nominal procedures followed by the Caller and Called Channel Access Sublayers for Making a Physical Link. This STANAG acknowledges that other implementations may meet the stated requirements.

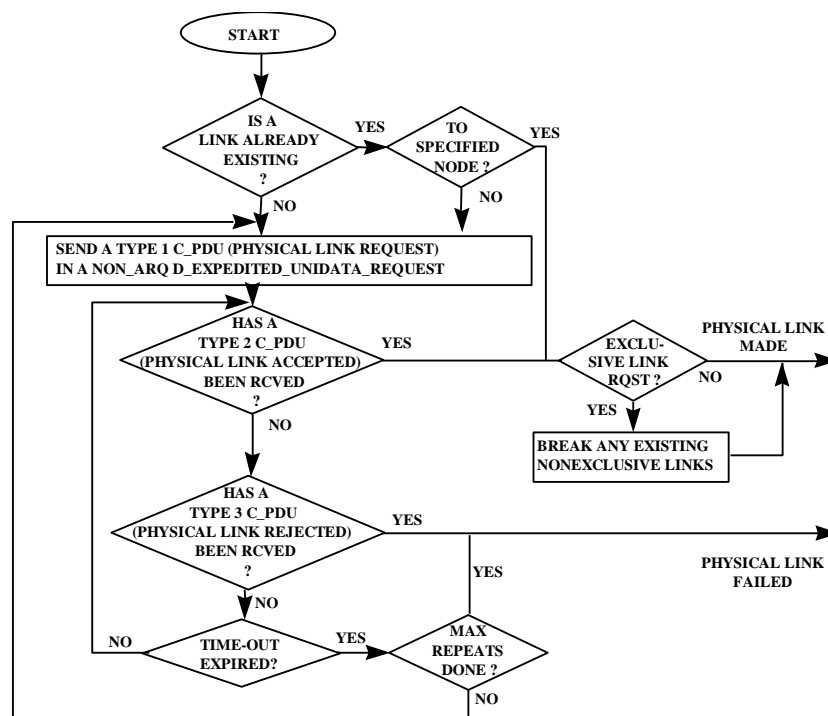


Figure B-7 (a). Type 1 Channel Access Sublayer protocol for Making a Physical Link (Caller Peer)

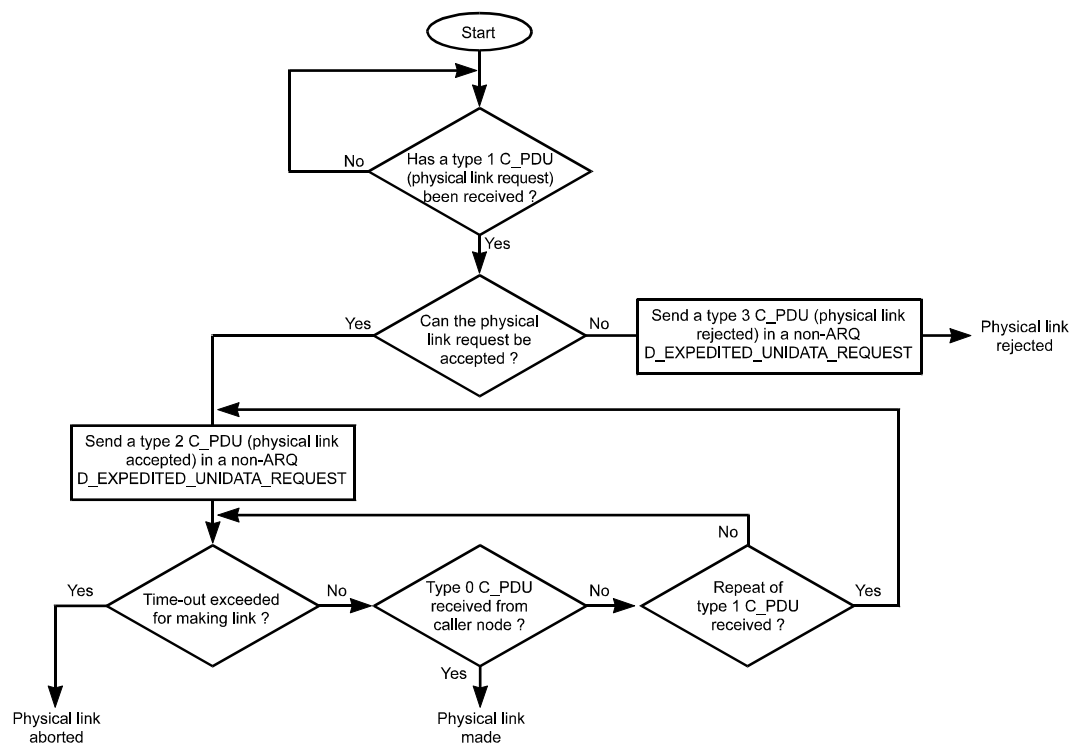


Figure B-7 (b). Type 1 Channel Access Sublayer Protocol for Making a Physical Link (Called Peer)

After having declared the Physical Link as successfully made, the peer Channel Access Sublayers do not carry out any further handshake in order to confirm the successful operation of the Link.

B.3.2.2 Protocol for Breaking a Physical Link

In the specification that follows of the protocol for breaking a physical link, the node which requests the breaking of the Physical Link will be referred to as the Initiator or initiating node, while the node which receives the request will be referred to as the Responder or responding node.

The protocol for breaking the Physical Link **shall** ⁽¹⁾ consist of the following steps:

Step 1: Initiator

- a) To start the protocol, the Initiator's Channel Access Sublayer **shall** ⁽²⁾ send a type 4 C_PDU (PHYSICAL LINK BREAK).
- b) Upon sending this C_PDU the Channel Access Sublayer **shall** ⁽³⁾ start a timer which is set to a value greater than or equal to the maximum time required by the Called Node to send its response (this time depends on the modem parameters, number of re-transmissions, etc.).

Step 2: Responder

- c) Upon receiving the type 4 C_PDU the Responder's Channel Access Sublayer **shall** ⁽⁴⁾ declare the Physical link as *broken* and send a PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU.

Step 3: Initiator

- d) Upon receiving a PHYSICAL LINK BREAK CONFIRM (TYPE 5) C_PDU, the Initiator's Channel Access Sublayer **shall** ⁽⁵⁾ declares the Physical Link as *broken*.
- e) Upon expiration of its timer without any response having been received from the remote node, the Initiator's Channel Access Sublayer **shall** ⁽⁶⁾ repeat step 1 and wait again for a response from the remote node.
- f) After having repeated Step 1 a maximum number of times (left as a configuration parameter) without any response having been received from the remote node, the Initiator's Channel Access Sublayer **shall** ⁽⁷⁾ declare the Physical Link as *broken*.

Figures B-8 (a) and (b) show the procedures followed by the Initiator and Responding Channel Access Sublayers for Breaking a Physical Link.

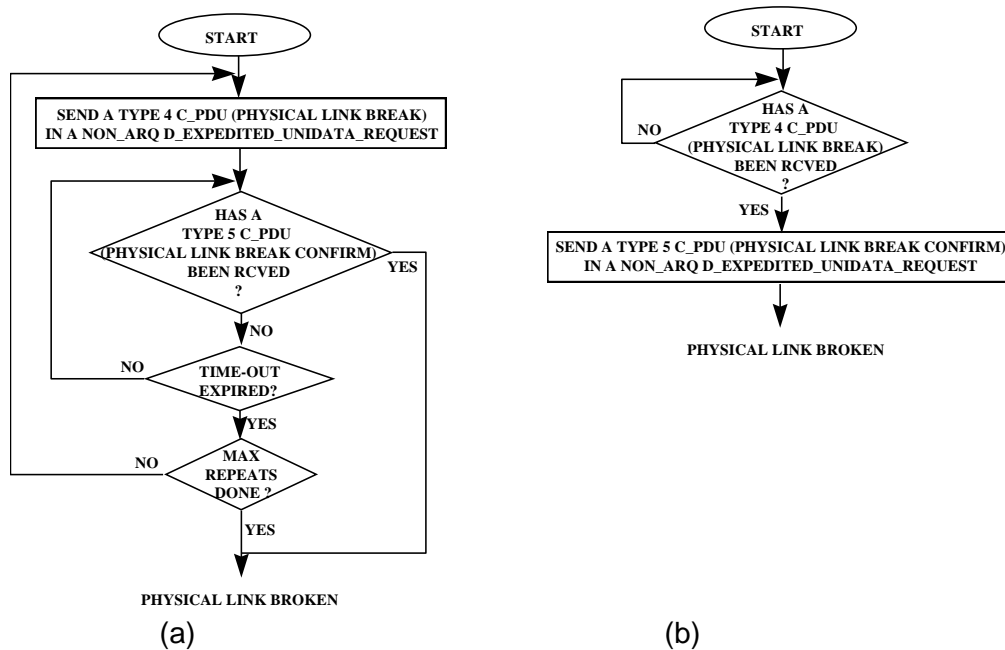


Figure B-8. Type 1 Channel Access Sublayer protocol for Breaking a Physical Link
(a) Initiator Peer. (b) Responding Peer.

B.3 Protocol for Exchanging Data C_PDU.

The protocol for exchanging C_PDUs between peers is simple and straightforward, since the Channel Access Sublayer relies entirely on its supporting (i.e, Data Transfer) sublayer to provide delivery services.

The sending peer **shall** ⁽¹⁾ accept S_PDUs from the Subnetwork Interface Sublayer, envelop them in a "DATA" C_PDU (by adding the C_PCI) and send them to its receiving peer via its interface to the Data Transfer Sublayer.

The receiving peer **shall** ⁽²⁾ receive DATA C_PDUs from the Data Transfer Sublayer interface, check them for validity, strip off the C_PCI, and deliver the enveloped S_PDU to the Subnetwork Interface Sublayer.

Annex C: Data Transfer Sublayer (Mandatory)

The Data Transfer Sublayer is responsible for the efficient transfer across the radio link of protocol data units from the Management Sublayer (i.e., M_PDUs) and Channel Access Sublayer (i.e., C_PDUs) in the STANAG 5066 profile. Usually, but not always, this means the error free delivery of C_PDUs to the Channel Access Sublayer. Efficient transmission over the radio channel of protocol data units of the Data Transfer Sublayer (i.e., D_PDUs, which contain the C_PDUs or M_PDUs) is achieved principally by two mechanisms. The first is segmentation of the large C_PDUs into smaller D_PDUs if necessary and the second is the selective repetition (selective ARQ) by the sending node of D_PDUs which were received in error. The other mode (non ARQ) of transmitting involves the segmentation of the large C_PDUs into smaller D_PDUs and combining the received D_PDUs such that the segmented C_PDU can be reconstructed in a “best-effort” attempt.

C.1 Data Transfer Sublayer Service Definition

Depending on the application and service-type requested by higher sublayers, the user service provided by the Data Transfer Sublayer **shall** ⁽¹⁾ be either a simple **non ARQ service** - commonly known as broadcast mode - or a reliable **selective ARQ service**, as specified herein.

The Data Transfer Sublayer **shall** ⁽²⁾ provide “sub-modes” for **non ARQ** and reliable **selective ARQ** delivery services, which influence the characteristics of the particular service, as specified below

In addition to the Selective ARQ and Non-ARQ services provided to the upper sublayers, the Data Transfer Sublayer shall provide an Idle Repeat Request service for peer-to-peer communication with the Data Transfer Sublayer of other nodes.

C.1.1 Non-ARQ Service

In the **non ARQ service** error-check bits (i.e., cyclic-redundancy-check or CRC bits) applied to the D_DPU **shall** ⁽¹⁾ be used to detect errors, and any D_PDUs that are found to contain transmission errors **shall** ⁽²⁾ be discarded by the data transfer sublayer protocol entity.

A special mode of the non-ARQ service **shall** ⁽³⁾ be available to reconstruct C_PDUs from D_PDUs in error and deliver them to the Channel Access Sublayer.

In the **non ARQ** mode, the following submodes may be specified:

- regular data service.
- expedited data service.
- “in order” delivery of C_PDUs is not guaranteed.
- delivery of complete or error free C_PDUs is not guaranteed.

C.1.2 Selective ARQ Service

The reliable **Selective ARQ service** **shall** ⁽¹⁾ use CRC check bits and flow control procedures, such as requests for retransmission of D_PDUs in which errors have been detected, to provide a reliable data transfer service.

In the **Selective ARQ** service, the following submodes may be specified:

- regular data service.
- expedited data service.
- node and client level delivery confirmation.
- “in-order” or “out-of-order” deliver of C_PDUs to the remote peer network layer.
- Delivery of complete and error free C_PDUs is guaranteed.

C.2 Interface Primitives Exchanged with the Channel Access Sublayer

The implementation of the interface between the Data Transfer Sublayer and the Channel Access Sublayer is not mandated or specified by this STANAG. Since the interface is internal to the subnetwork architecture and may be implemented in a number of ways it is considered beyond the scope of STANAG 5066. A model of the interface has been assumed, however, for the purposes of discussion and specification of other sublayer functions. The STANAG’s presentation of the interface is modelled after that used in Reference 1 [1] to this STANAG, however, the interface defined in the reference is not mandated for use herein.

Despite the advisory nature of the conceptual model of the internal interface between the Data Transfer Sublayer and the Channel Access Sublayer, there are some mandatory requirements that are placed on any interface implementation.

The interface must support the service-definition for the Data Transfer Sublayer, i.e.:

1. The interface **shall** ⁽¹⁾ allow the Channel Access Sublayer to submit protocol data units (i.e., C_PDUs) for transmission using the regular and expedited delivery services provided by the Data Transfer Sublayer.
2. The interface **shall** ⁽²⁾ allow the Data Transfer Sublayer to deliver C_PDUs to the Channel Access Sublayer.
3. The interface **shall** ⁽³⁾ permit the Channel Access Sublayer to specify the delivery services, priority, and time-to-die required by the C_PDUs when it submits them to the Data Transfer Sublayer.
4. The interface **shall** ⁽⁴⁾ permit the Data Transfer Sublayer to specify the delivery services that were used by received C_PDUs when it submits them to the Channel Access Sublayer.
5. The interface **shall** ⁽⁵⁾ permit the Channel Access Sublayer to specify the destination address to which C_PDUs are to be sent.
6. The interface **shall** ⁽⁶⁾ permit the Data Transfer Sublayer to specify the source address from which C_PDUs are received, and the destination address to which they had been sent.

¹ Clark, D., and N. Karavassillis, “Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio”, TM-937, May 1998, pp. 109-115

7. The interface **shall** ⁽⁷⁾ permit the Data Transfer Sublayer to notify the Channel Access sublayer when a warning indication (i.e., a WARNING D_PDU) has been received from a remote peer, the source and destination address associated with the warning, the reason for the warning, and the event (i.e., message type) that triggered the warning message.
8. The interface **shall** ⁽⁸⁾ permit the Data Transfer Sublayer to notify the Channel Access sublayer that a warning indication (i.e., a WARNING D_PDU) has been sent to a remote peer, the destination address associated with the warning, the reason for the warning, and the event (i.e., message type) that triggered the warning message.
9. The interface **shall** ⁽⁹⁾ permit the Channel Access sublayer to notify the Data Transfer Sublayer that a Connection (i.e., either an Exclusive or Nonexclusive Link) has been established with a given node.
10. The interface **shall** ⁽¹⁰⁾ permit the Channel Access sublayer to notify the Data Transfer Sublayer that a Connection (i.e., either an Exclusive or Nonexclusive Link) has been terminated with a given node.
11. The interface **shall** ⁽¹¹⁾ permit the Data Transfer Sublayer to notify the Channel Access sublayer that a Connection (i.e., either an Exclusive or Nonexclusive Link) has been lost with a given node.

Additionally, the protocol-control information from the Channel Access sublayer that is required for the management of the Data Transfer Sublayer **shall** ⁽¹²⁾ not be derived from knowledge of the contents or format of any client data or U_PDUs encapsulated within the C_PDUs exchanged over the interface.

[Note: user's that encrypt their traffic prior to submittal may use the subnetwork. Subnetwork operation must be possible with client data in arbitrary formats that are unknown to the subnetwork, therefore any service requirements or indications must be provided by interface control information provided explicitly with the user data.]

The interface may use knowledge of the contents of C_PDUs (excluding the contents of any encapsulated U_PDUs) to derive protocol control information for the Data Transfer sublayer. This approach is highly discouraged, however. The recommended approach for implementation is that information required for protocol control within the Data Transfer sublayer should be provided explicitly in appropriate interface primitives.

In keeping with accepted practice in the definition of layered protocols, and as a means for specifying the operations of the sublayers that are mandated by this STANAG, the communication between the Data Transfer Sublayer and the Channel Access Sublayer is described herein with respect to a set of Primitives. The interface Primitives are a set of messages for communication and control of the interface and service requests made between the two layers.

By analogy to the design of the client-subnetwork interface specified in Annex A, the technical specification of the Data Transfer Sublayer assumes communication with the Channel Access Sublayer using primitives prefixed with a "D_". A minimal set of D_Primitives has been assumed that meet the requirements stated above and the general function for each D_Primitive is given in Table C-1. These D_Primitives are given without benefit of a list of arguments or detailed description of their use.

**Table C-1 – Nominal Definition of D_Primitives for the Interface between the Data Transfer Sublayer and the Channel Access sublayer
(non-mandatory, for information-only)**

| NAME OF PRIMITIVE | DIRECTION (see Note) | COMMENTS |
|--------------------------------------|----------------------|---|
| D_UNIDATA_REQUEST | CAS →DTS | Request to send an encapsulated C_PDU using the regular delivery service to a specified destination node address with given priority, time-to-die, and delivery mode. |
| D_UNIDATA_REQUEST_CONFIRM | DTS →CAS | Confirmation that a given C_PDU has been sent using the regular delivery service |
| D_UNIDATA_REQUEST_REJECTED | DTS →CAS | Notification that a given C_PDU could not be sent using the regular delivery service and the reason why it was rejected by Data transfer Sublayer. |
| D_UNIDATA_INDICATION | DTS →CAS | Delivers a C_PDU that has been received using the regular delivery service from a remote node, with additional indications of the transmission service given to the C_PDU, the source node address, and the destination node address (e.g., if a group address is the destination). |
| D_EXPEDITED_UNIDATA_REQUEST | CAS →DTS | Request to send an encapsulated C_PDU using the expedited delivery service to a specified destination node address with a specified delivery mode. |
| D_EXPEDITED_UNIDATA_REQUEST_CONFIRM | DTS →CAS | Confirmation that a given C_PDU has been sent using the expedited delivery service |
| D_EXPEDITED_UNIDATA_REQUEST_REJECTED | DTS →CAS | Notification that a given C_PDU could not be sent using the expedited delivery service and the reason why it was rejected by Data transfer Sublayer. |
| D_EXPEDITED_UNIDATA_INDICATION | DTS →CAS | Delivers a C_PDU that has been received using the expedited delivery service from a remote node, with additional indications of the transmission service given to the C_PDU, the source node address, and the destination node address (e.g., if a group address is the destination). |
| D_WARNING_RECEIVED | DTS →CAS | Notifies the Channel Access sublayer that a WARNING D_PDU has been received from a remote node by the Data Transfer Sublayer, with the reason for sending the warning message |
| D_WARNING_TRANSMITTED | DTS →CAS | Notifies the Channel Access sublayer that a WARNING D_PDU has been sent to a remote node by the Data Transfer Sublayer, with the reason for sending the warning message |
| D_CONNECTION_MADE | CAS →DTS | Notifies the Data Transfer Sublayer that a connection has been made with a given remote node. |
| D_CONNECTION_TERMINATED | CAS →DTS | Notifies the Data Transfer Sublayer that a connection has been terminated with a given remote node. |
| D_CONNECTION_LOST | DTS →CAS | Notifies the Channel Access Sublayer that a connection has been lost with a given remote node. |

Note: DTS = Data Transfer Sublayer; CAS = Channel Access Sublayer; <from sublayer> → <to sublayer>

C.3 Structure of Data Transfer Sublayer Protocol Data Units (D_PDUs)

In order to provide the data transfer services specified herein, the Data Transfer Sublayer **shall** ⁽¹⁾ exchange protocol data units (D_PDUs) with its peer(s).

The Data Transfer Sublayer **shall** ⁽²⁾ use the D_PDU types displayed in Table C-2 to support the **Selective ARQ service** and **Non ARQ** service, including the several data transfer submodes defined herein.

Table C-2. D_PDU Types

| <i>D_PDU Frame Types</i> | <i>D_PDU Type #.</i> | <i>Function</i> | <i>Protocol Type</i> | <i>Frame Type</i> |
|--------------------------|----------------------|---|----------------------|-------------------|
| DATA-ONLY | 0 | Simplex data transfer | SRQ | I |
| ACK-ONLY | 1 | Acknowledgement of type 0, 2 data transfer | - | C |
| DATA-ACK | 2 | Duplex data transfer | SRQ | I + C |
| RESET/WIN-RESYNC | 3 | Reset/Re-synchronise peer protocol entities | IRQ | C |
| EXP-DATA-ONLY | 4 | Expedited simplex data transfer | SRQ | I |
| EXP-ACK-ONLY | 5 | Acknowledgement of type 4 data transfer | - | C |
| MANAGEMENT | 6 | Management message transfer | IRQ | C |
| NON-ARQ DATA | 7 | Non-ARQ data transfer | NRQ | I |
| EXP NON-ARQ DATA | 8 | Expedited non-ARQ data transfer | NRQ | I |
| - | 9-14 | Reserved for future extensions | - | - |
| WARNING | 15 | Unexpected or unrecognised D_PDU type | - | C |

C-Frame = Control Frame. I-Frame = Information Frame. I+C-Frame = Information + Control Frame.

All D_PDU types may be used to support half duplex and full duplex transmission modes, used by single and split-frequency operation. There are basically three different types of D_PDUs, or frames, noted by the *Frame-Type* field in Table C-2:

1. C (Control) Frames,
2. I (Information) Frames,
3. and a combined I+C Frame.

The *Protocol Type* field in Table C -2 indicates the type of data-transfer-service protocol with which the D_PDU frame **shall** ⁽³⁾ be used, as follows:

4. NRQ No Repeat-Request.(i.e., Non-ARQ) Protocol
5. SRQ Selective Repeat-Request Protocol
6. IRQ Idle Repeat-Request Protocol

The NRQ protocol **shall** ⁽⁴⁾ only operate in a simplex mode since the local node, after sending I-frames, does not wait for an indication from the remote node as to whether or not the I-frames were correctly received. Multiple repetitions of I-frames can be transmitted in order to increase the likelihood of reception under poor channel conditions, in accordance with the requested service characteristics.

The Selective RQ protocol **shall** ⁽⁵⁾ operate in a half or full duplex mode since the local node, after sending I-frames, waits for an indication in the form of a selective acknowledgement from the remote node as to whether the I-frames were correctly received or not. The local node then either sends the next I-frame, if all the previous I-frames were correctly received, or retransmits copies of the previous I-frame that were not, in accordance with the requirements of Section C.6. The local node will retransmit copies of the previous I-frames if no indication is received after a predetermined time interval. A local node sending I-frames in full duplex mode also sends the indication in the form of a selective acknowledgement embedded in the I-frames as to whether the I-frames were correctly received or not.

The Idle RQ protocol, also known as a stop and wait protocol, **shall** ⁽⁶⁾ operate in a half-duplex mode; the local node, after sending an I-frame, must wait until it receives an acknowledgement from the remote node as to whether or not the I-frame was correctly received. The local node then either sends the next I-frame, if the previous I-frame was correctly received, or retransmits a copy of the previous I-frame if it was not. The local node will retransmit a copy of the previous I-frame if no indication is received after a predetermined time interval.

Different D_PDU frame types may be combined in a transmission, subject to limitations imposed by the state of the Data Transfer Sublayer protocol. These states of the Data Transfer Sublayer protocol and their associated requirements are given in Section C.6.1

C.3.1 Generic simplified D_PDU structure

All D_PDU types that cannot carry segmented C_PDUs **shall** ⁽¹⁾ be of the structure shown in Figure C-1 (a).

D_PDU types that can carry segmented C_PDUs **shall** ⁽²⁾ be structured according to Figure C-1 (b).

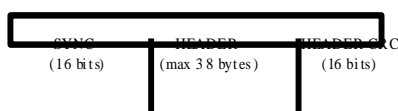


Figure C-1 (a). Format for D_PDU C-Frame types (1, 3, 5, 6 and 15)

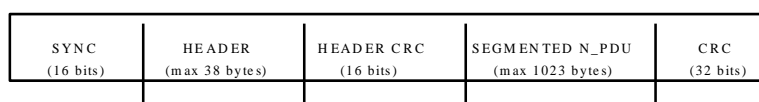


Figure C-1 (b). Format for D_PDU I and I+C Frame types (0, 2, 4, 7 and 8)

C.3.2 Generic detailed D_PDU structure

The detailed structure of the generic D_PDU C-Frame **shall** ⁽¹⁾ be as shown in Figure C-2 (a) or Figure C-2(b)

The D_PDU types 1, 3, 5, 6 and 15 **shall** ⁽²⁾ use only the C-Frame structure defined in Figure C-2 (a).

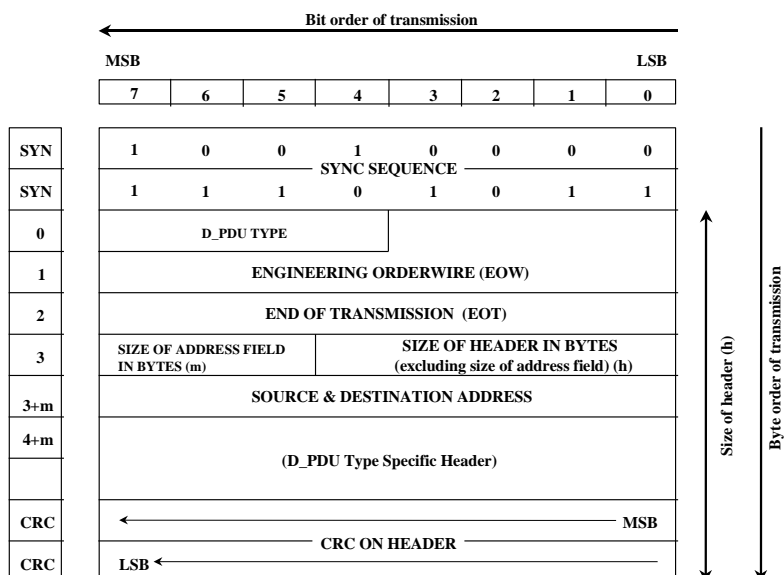


Figure C-2 (a). Generic D_PDU C-Frame Structure.

The D_PDU types 0, 2, 4, 7 and 8 shall ⁽³⁾ use the generic D_PDU I and I+C Frame structure defined in Figure C-2 (b).

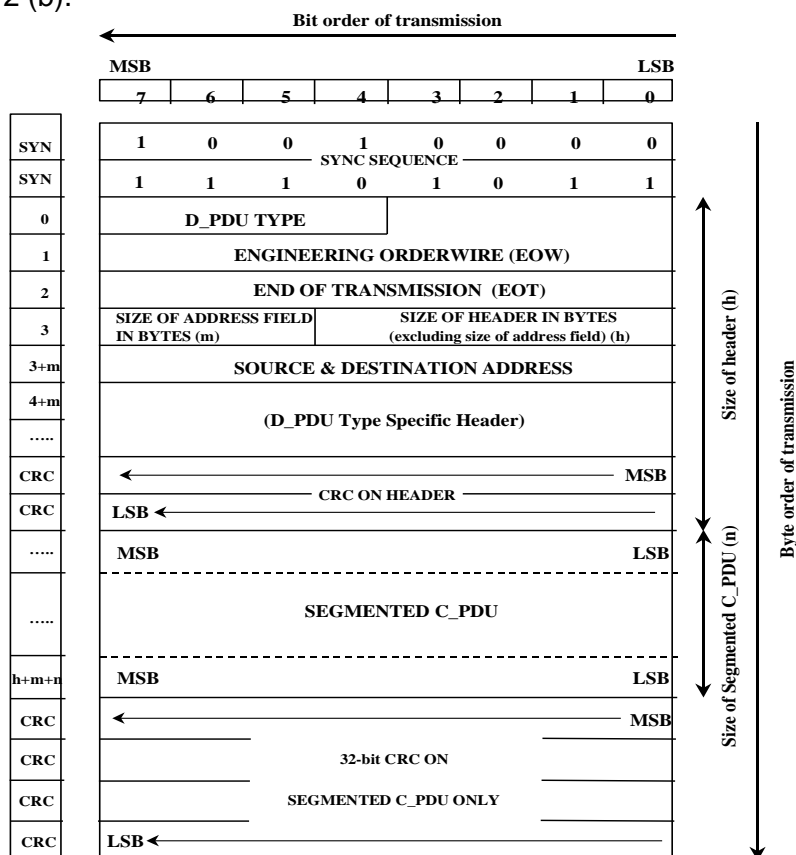


Figure C-2 (b). Generic D_PDU I and I+C Frame Structure.

All D_PDUs, regardless of type, **shall** ⁽⁴⁾ begin with the same 16-bit synchronisation (SYNC) sequence.

The 16-bit sequence **shall** ⁽⁵⁾ be the 16-bit Maury-Styles (0xEB90) sequence shown below, with the least significant bit (LSB) transmitted first:

(MSB) 1 1 1 0 1 0 1 1 1 0 0 1 0 0 0 0 (LSB)

16-bit Maury-Styles synchronisation sequence

The first 4 bytes of all D_PDU headers **shall** ⁽⁶⁾ contain the same fields:

1. a 4 bit *D_PDU Type* field that **shall** ⁽⁷⁾ identify the type of D_PDU;
2. a 12-bit field that **shall** ⁽⁸⁾ contain an engineering order wire (EOW) message;
3. an 8-bit field that **shall** ⁽⁹⁾ contain the end of transmission (EOT) information; and
4. a one byte field that **shall** ⁽¹⁰⁾ contain both a *Size-of-the-Address* field (3 bits) and a *Size-of-the-Header* (5 bits) field.

The next 1 to 7 bytes of every header, as specified in the *Size-of-the-Address* field, **shall** ⁽¹¹⁾ contain source and destination address information for the D_PDU.

The D_PDU *Type-Specific-Header-Part* field **shall** ⁽¹²⁾ be as specified below in this STANAG, for each of the D_PDU types.

The last two bytes of every header **shall** ⁽¹³⁾ contain the Cyclic Redundancy Check (CRC) calculated in accordance with Section C.3.2.8.

The bits in any field in a D_PDU that is specified as NOT USED **shall** ⁽¹⁴⁾ contain the value zero (0).

C.3.2.1D_PDU type

The D_PDU types **shall** ⁽¹⁾ be as defined in Table C-2 and the D_PDU figures below.

The value of the D_PDU type number **shall** ⁽²⁾ be used to indicate the D_PDU type. The four bits available allow for 16 D_PDU types. Ten are here defined, the rest **shall** ⁽³⁾ remain reserved for future extensions to the STANAG.

C.3.2.2Engineering Orderwire (EOW)

The 12 bit EOW field **shall** ⁽¹⁾ carry Management messages for the Engineering Orderwire (EOW). EOW messages may not be explicitly acknowledged although the D_PDU of which they are a part may be. EOW messages can be explicitly acknowledged when they are contained in the MANAGEMENT Type 6 D_PDU through which Management-level acknowledgement services are provided in the Data Transfer Sublayer.

Figure C-3 (a) shows the generic 12-bit EOW structure. The first 4 bits of the EOW **shall** ⁽²⁾ contain the EOW-type field, which identifies the type of EOW message. The remaining 8-bits **shall** ⁽³⁾ contain the EOW-type-specific EOW data.

The various EOW messages including their definitions and extensions are specified further in Section C.5.

Figure C-3 (b) shows how the EOW message is mapped into the generic D_PDU header.

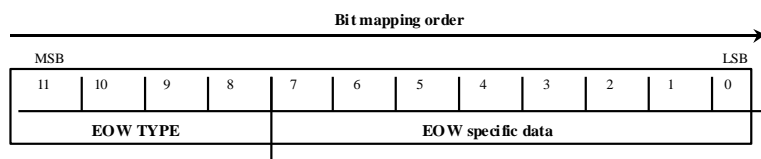


Figure C-3 (a). Generic EOW message.

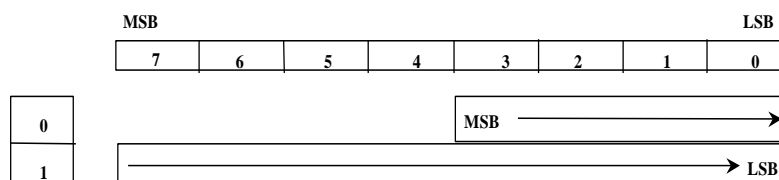


Figure C-3 (b). EOW mapping convention in D_PDU header.

C.3.2.3 End Of Transmission (EOT)

The 8-bit EOT field **shall** ⁽¹⁾ provide an approximation of the time remaining in the current transmission interval specified by the transmitting node. This information is provided for the timing of exchanges between peer nodes to minimize collisions and the link turnaround time (time between the end of a transmission by one node and the start of a transmission by another node).

The number in this field **shall** ⁽²⁾ be a binary number expressing the number of half (1/2) second intervals remaining in the current transmission from the beginning of the current D_PDU including sync bytes.

When operating in half duplex mode, a node **shall** ⁽³⁾ not make a transmission during a transmission by another node (i.e., before the EOT expires).

Once an EOT is sent, the EOT in each subsequent D_PDU in that transmission **shall** ⁽⁴⁾ contain a consistent calculation of the EOT, that is, monotonically decreasing in half-second (0.5 second) intervals.

Calculations of the EOT by a transmitting node **shall** ⁽⁵⁾ be rounded up to the nearest half-second interval. [Note: rounding-up the calculation of the EOT, rather than truncating it, ensures that the transmission will be completed by the time declared in the EOT field of the D_PDU.]

A half-duplex node **shall** ⁽⁶⁾ stop transmitting and shift to receive mode when the EOT becomes zero.

When a node is in broadcast mode, the EOT field **shall** ⁽⁷⁾ be filled with all zeros, unless the remaining broadcast transmission interval is within the maximum EOT value of 127.5 seconds. If a node is in broadcast mode and either the remaining broadcast transmission interval or the total broadcast transmission interval is within the maximum EOT value of 127.5 seconds, the EOT value **shall** ⁽⁸⁾ be computed and advertised as specified herein.

When a node is configured for and operating in full duplex mode, the EOT field **shall** ⁽⁹⁾ be filled with all zeros.

When D_PDU types are combined in a single transmission by the sublayer, any previously advertised values of EOT **shall** ⁽¹⁰⁾ not be violated or contradicted.

When D_PDU types are combined in a single transmission by the sublayer, advertised values for EOT **shall** ⁽¹¹⁾ refer to the end of the combined transmission and not to the end of transmission of the D_PDU which contains the EOT field.

C.3.2.4 Size of Address Field

The Size-of-Address Field **shall** ⁽¹⁾ specify the number of bytes in which the source and destination address are encoded (Note: this value is denoted by the integer value “m” in Figure C-2(a) and Figure C-2(b)). The address field may be from one (1) to seven (7) bytes in length, with the source and destination address of equal length.

Since the D_PDU header must be made up of an integer number of bytes, addresses **shall** ⁽²⁾ be available in 4-bit increments of size: 4 bits (or 0.5 bytes), 1 byte, 1.5 bytes, 2 bytes, 2.5 bytes, 3 bytes, and 3.5 bytes.

C.3.2.5 Size of Header Field

The Size-of-Header field **shall** ⁽¹⁾ specify the number of bytes in which the D_PDU is encoded. (Note: this value is denoted by the integer value “h”, Figure C-2(a) and Figure C-2(b)), and its value includes the sizes of the following fields and elements:

- D_PDU Type
- EOW
- EOT
- Size of Address field
- Size of Header field
- D_PDU-Type-specific header
- CRC field

The value of the Size-of-Header field **shall** ⁽²⁾ not include the size of the source and destination address field.

C.3.2.6 Source & Destination Address

Each D_PDU transmitted by a node **shall** ⁽¹⁾ contain the source and destination address. Half of the bits are assigned to the source and the other half to the destination.

The first half **shall** ⁽²⁾ be the destination address and the second half **shall** ⁽³⁾ be the source address as displayed nominally in Figure C-4(a) (which assumes an odd-number as the address-field size) or Figure C-4(b) (which assumes an even-number as the address-field size).

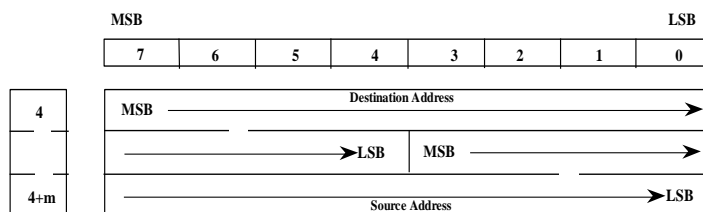


Figure C-4(a). Address mapping convention in D_PDU header, assuming address-field size is odd.

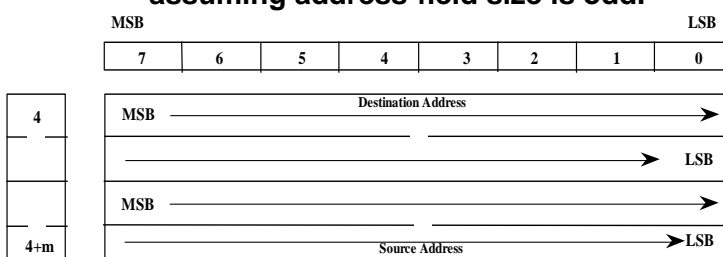


Figure C-4(a). Address mapping convention in D_PDU header, assuming address-field size is even.

Addresses **shall** ⁽⁴⁾ be in the form of a binary number. With 7 bytes available for each of the user and the destination, the smallest possible address field is 4 bits, the largest possible is 3.5 bytes, or 28 bits.

A decimal number **shall** ⁽⁵⁾ represent each byte or fractional byte of an address, and the binary equivalent **shall** ⁽⁶⁾ be mapped into the corresponding byte. Consequently, the decimal representation for node addresses (i.e., unicast addresses) lies in the range [0.0.0.0 ... 15.255.255.255] (this **may** be referred to as 'dotted-decimal address format').

For DPDU's that support group addressing (e.g; THE Type 7/8 Non-ARQ DPDU's), the group-address flag **should** be considered the 29th-bit in the binary representation of the address. Consequently, the decimal representation for group addresses (i.e., multicast addresses) lies in the range [16.0.0.0 ... 31.255.255.255].

The broadcast (all-stations) address in "dotted-decimal" address format is the group address 31.255.255.255.

Any fractional-byte elements in the address **shall** ⁽⁷⁾ be mapped into the first (leftmost) non-zero number in the decimal representation of the address. The remaining numbers in the decimal representation of the address **shall** ⁽⁸⁾ refer to byte-sized elements in the address field. [Note: As an example, if 3.5 bytes are used, the address would be expressed as w.x.y.z, where w can be any value from 0 to 15, and x, y and z can be any value from 0 to 255. The value w will represent the most significant bits and z the least significant bits of the address.]

The address bits **shall** ⁽⁹⁾ be mapped into the address field by placing the MSB of the address into the MSB of the first byte of the address field, and the LSB into the LSB of the last byte of the field, in accordance with Figure C-4(a), for addresses with length of 0.5, 1.5, 2.5, or 3.5 bytes, and Figure C-4(b), for addresses with length of 1, 2, or 3 bytes.

When a field spans more than one octet, the order of the bit values within each octet **shall** ⁽¹⁰⁾ decrease progressively as the octet number increases.

The lowest bit number associated with the field represents the lowest-order value. Leading address bytes which are zero may be dropped from the address, consistent that the requirement that the source and destination address subfields must be of equal length. Trailing address bytes that are zero **shall** ⁽¹¹⁾ be sent.

C.3.2.7 D_PDU Type-Specific Header

The bytes immediately following the address field **shall** ⁽¹⁾ encode the D_PDU Type-Specific header, as specified in the corresponding section below from Sections C.3.3 through C.3.12.

C.3.2.8 Cyclic Redundancy Check (CRC)

The two-bytes following the D_PDU Type-Specific header **shall** ⁽¹⁾ contain a 16-bit Cyclic Redundancy Check (CRC) field.

The header CRC error-check field **shall** ⁽²⁾ be calculated using the following polynomial: $x^{16} + x^{15} + x^{12} + x^{11} + x^8 + x^6 + x^3 + 1$, or in hexadecimal format 0x19949, using the shift-register method shown by the figures in Appendix I of CCITT Recommendation V.41 (or equivalent method in software; an example is given below).

[Note: This polynomial is not the same as that shown in the figure in V.41, Appendix I; the polynomial was chosen to provide a lower probability of undetected error (i.e., better performance) than that given by the polynomial specified in V.41. The polynomial specified here was analyzed and reported by Wolf in a paper in the 1988 IEEE Conference on Military Communications (MILCOM- paper 15.2). Note too that the taps in the figure are shown reversed from the order in which they occur in the polynomial defined in V.41. This reversal is accommodated by the remaining requirements specified below.]

When calculating the header CRC field, the shift registers **shall** ⁽³⁾ be initially set to all (0) zeros.

The header CRC **shall** ⁽⁴⁾ be calculated over all bits in the header, excluding the Maury-Styles synchronisation sequence, and including the following fields and elements:

- D_PDU Type
- EOW
- EOT
- Size of Address field
- Size of Header field
- Source and Destination Address
- D_PDU-Type-Specific header

A node **shall** ⁽⁵⁾ process the information contained in a header with a valid CRC, regardless of the result of the CRC error check over any segmented C_PDU that may be a part of the D_PDU.

The CRC bits **shall** ⁽⁶⁾ be mapped (see Figure C-5) into the CRC octets by placing the MSB of the CRC into the LSB of the first byte of the CRC field, and the LSB of the CRC into the MSB of the last byte of the CRC field. This will result in the MSB of the most significant byte of the CRC being sent first, followed by the remaining bits in descending order, which is consistent with the order of transmission for CRC bits, as specified in Recommendation V.42, section 8.1.2.3.

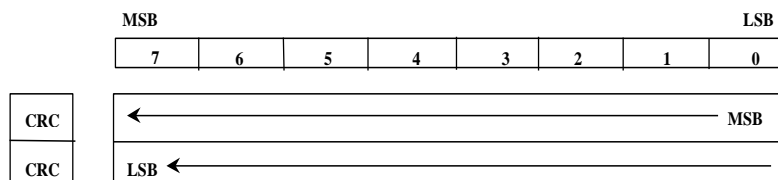


Figure C-5. CRC mapping convention in D_PDU header.

The following C code can be used to calculate the CRC value using the specified polynomial.

```

/* Polynomial  $x^{16} + x^{15} + x^{12} + x^{11} + x^8 + x^6 + x^3 + 1$  */
unsigned short CRC_16_S5066(unsigned char DATA, unsigned short CRC)
{
    unsigned char i, bit;
    for (i=0x01; i; i<=1)
    {
        bit = (((CRC & 0x0001) ? 1:0)^(DATA&i ? 1:0));
        CRC>>=1;
        if (bit) CRC^=0x9299;          /* polynomial representation, bit */
    }                                /* -reversed (read right-to-left), and */
                                    /* with the initial  $x^{16}$  term implied. */

    return (CRC);
}

/* example of usage: */
#define NUM_OCTETS; /* number of octets in the message data */
unsigned char    message[NUM_OCTETS];
unsigned short    CRC_result;
unsigned int      j;

CRC_result = 0x0000;
for (j=0; j < NUM_OCTETS; j++){
    CRC_result = CRC_16_S5066(message[j], CRC_result);
}
/* CRC_result contains final CRC value when loop is completed */

```

NOTE: *This code calculates the CRC bytes in the proper order for transmission as defined above; no bit reversal is required with this code.*

Code Example C-1. Calculation of the CRC-16-S5066 for D_PDU Headers.

The function CRC_16_5066 in Code Example C-1 may be used to calculate the CRC for a data sequence of octets, and is called for each successive octet in the sequence. The function accepts as its first argument an octet in the data sequence, and as its second argument, the value of the CRC calculated for the previous octet in the sequence. The function returns the value of the CRC. When called for the first octet in the data sequence, the value of the CRC must be initialized to zero as required.

The result of this code for the D_PDU described below is 0x1E5F. The least significant byte of the computed CRC should be transmitted before the most significant byte; as noted, the algorithm already performs the requisite bit-level reversal. The receive processing should extract the CRC bytes and re-assemble into the correct order.

D_PDU used in above CRC calculation:
D_PDU Type: Warning
EOW: Type 0 (i.e., all zeros)
EOT: all zeros
Address size: 2
Destination address: 0.0.0.5 (leading zeros not encoded)
Source address: 0.0.0.100 (leading zeros not encoded)
Received D_PDU type: 0
Reason warning sent: 2
CRC: 0x1E5F

The full D_PDU is then (including sync bytes and CRC-field, properly bit-reversed and in the order they should be transmitted):

0x90, 0xEB, 0xF0, 0x00, 0x00, 0x47, 0x05, 0x64, 0x02, 0x5F, 0x1E

C.3.2.9 Segmented C_PDU

For the I and I+C D_PDUs types, the octets of the segmented C_PDUs **shall** ⁽¹⁾ be transmitted in ascending numerical order, following the two-byte CRC on the D_PDU header.

Within an octet, the LSB **shall** ⁽²⁾ be the first bit to be transmitted as shown in Figure C-6.

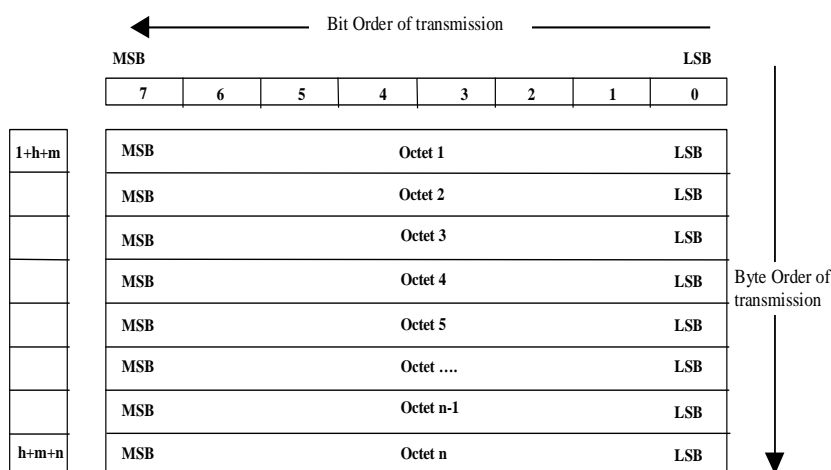


Figure C-6. Segmented C_PDU mapping convention in D_PDU structure.

C.3.2.10 Size of Segmented C_PDU

The SIZE OF SEGMENTED C_PDU field **shall** ⁽¹⁾ be used only with D_PDUs that are I or I+C frame types, i.e., that have a Segmented C_PDU field as shown in Figure C-2(b). (Note: The value of the SIZE OF SEGMENTED C_PDU field is denoted by the integer value “n” in the Figure). This field actually is contained within the D_PDU Type-Specific Header part, but since it is common to all I and I+C D_PDUs the format is defined here.

The bit-value of the SIZE OF SEGMENTED C_PDU **shall** ⁽²⁾ be encoded as a ten-bit field as indicated by Figure C-7.

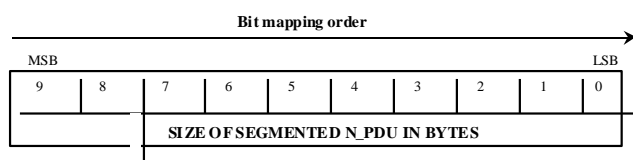


Figure C-7. SIZE OF SEGMENTED C_PDU Field.

The value in the SIZE OF SEGMENTED C_PDU field **shall** ⁽³⁾ not include the two-bytes for the CRC following the Segmented C_PDU. The Segmented C_PDU field can hold a maximum of 1023 bytes from the segmented C_PDU.

The SIZE OF SEGMENTED C_PDU **shall** ⁽⁴⁾ be mapped into consecutive bytes of the D_PDU as indicated in Figure C-8, in the byte locations specified for the applicable D_PDU.

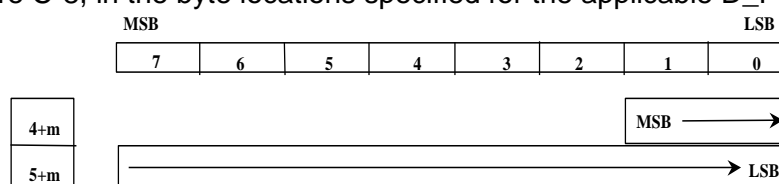


Figure C-8. Segmented C_PDU Size mapping convention in all applicable D_PDU header structures.

C.3.2.11 CRC-ON-SEGMENTED-C_PDU Field

The last four bytes of any I or I+C D_PDU **shall** ⁽¹⁾ contain a 32-bit Cyclic Redundancy Check (CRC) field.

The CRC **shall** ⁽²⁾ be applied and computed on the contents of the Segmented C_PDU using the following polynomial [see footnote ² below]:

$$x^{32} + x^{27} + x^{25} + x^{23} + x^{21} + x^{18} + x^{17} + x^{16} + x^{13} + x^{10} + x^8 + x^7 + x^6 + x^3 + x^2 + x + 1$$

or, in hexadecimal notation: 0x10AA725CF,

using the shift-register method similar to that shown by the figures in Appendix I of CCITT Recommendation V.41, but using a longer shift register and appropriate changes to the tap configuration corresponding to the polynomial specified. Equivalent implementations in software may be used to compute the 32-bit CRC (an example is offered below).

When calculating the header CRC field, the shift registers **shall** ⁽⁴⁾ be initially set to all (0) zeros.

² This polynomial is constructed with a methodology similar to that used to construct the 16-bit polynomial on the header, and for the same reason. Using this methodology, the polynomial for the 32-bit CRC on the segmented C_PDU is the generator polynomial for a two-error-correcting BCH code (65535,32), with a maximum information block length of 65503 bits. Only the error-detection properties of the code are used.

The following C code can be used to calculate the CRC value using the specified polynomial.

/ Polynomial:*

```

x32 + x27 + x25 + x23 + x21 + x18 + x17 + x16 + x13 + x10 + x8 + x7 + x6 + x3 + x2 + x + 1 */
unsigned int CRC_32_S5066(unsigned char DATA, unsigned int CRC)
{
    unsigned char i, bit;

    for (i=0x01; i; i<=1)
    {
        bit = (((CRC & 0x0001) ? 1:0)^(DATA&i ? 1:0));
        CRC>>=1;
        if (bit) CRC^=0xF3A4E550;          /* polynomial representation, bit */
    }                                     /* -reversed (read right-to-left), and */
                                         /* with the initial x32 term implied. */

    return (CRC);
}

/* example of usage: */
#define NUM_OCTETS;      /* number of octets in the message data */
unsigned char          message[NUM_OCTETS];
unsigned int  CRC_result;
unsigned int  j;

CRC_result = 0x00000000;
for (j=0; j < NUM_OCTETS; j++){
    CRC_result = CRC_32_S5066(message[j], CRC_result);
}
/* CRC_result contains final CRC value when loop is completed */

```

NOTE: *This code calculates the CRC bytes in the proper order for transmission as defined above; no bit reversal is required with this code.*

Code Example C-2. Calculation of the CRC-32-S5066 on Segmented C_PDUs.

The function CRC_32_S5066 in Code Example C-2 may be used to calculate the CRC for a data sequence of octets, and is called for each successive octet in the sequence. It is initialized and used in the same manner as the example for the STANAG 5066 16-bit CRC.

When applied to this short C_PDU message data sequence:

message[] = {0xF0, 0x00, 0x00, 0x47, 0x05, 0x64, 0x02},

the result is the 32-bit value (in hexadecimal format): 0xF4178F95

Just as for the 16-bit CRC computation, the algorithm that calculates the 32-bit CRC performs the bit-level reversal in place, and the resultant value must be transmitted least-significant byte first, in order to most-significant byte last. The full message data, with 32-bit CRC appended in proper position, is the following sequence:

0xF0, 0x00, 0x00, 0x47, 0x05, 0x64, 0x02, 0x95, 0x8F, 0x17, 0xF4

C.3.3 DATA-ONLY (TYPE 0) D_PDU (Simplex data transfer)

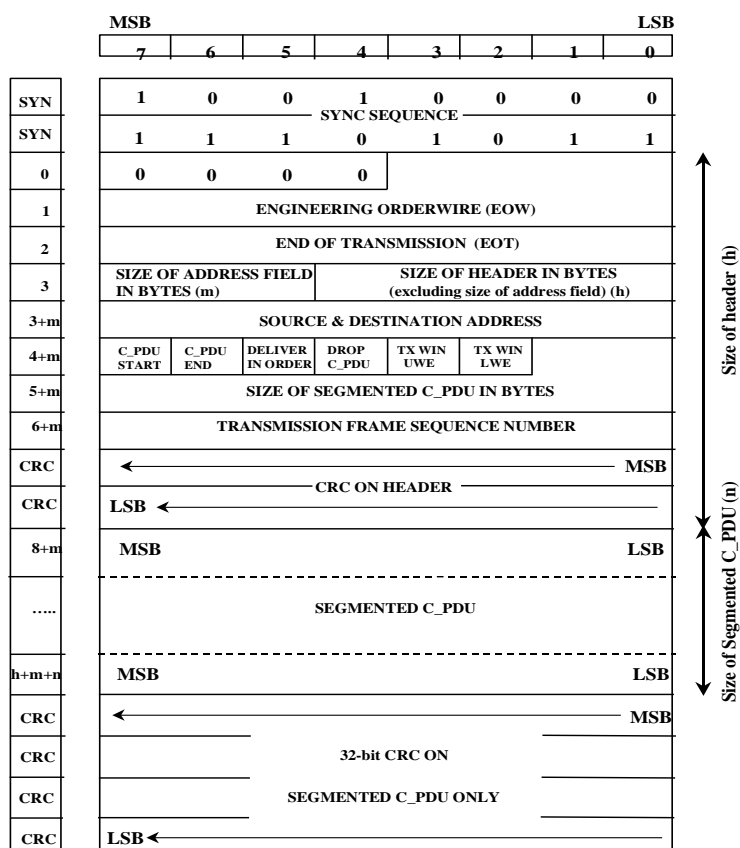


Figure C-9. Frame format for DATA-ONLY D_PDU Type 0

The DATA-ONLY D_PDU **shall** ⁽¹⁾ be used to send segmented C_PDUs when the transmitting node needs an explicit confirmation the data was received.

The DATA-ONLY D_PDU **shall** ⁽²⁾ be used in conjunction with a basic selective automatic repeat request type of protocol.

A Data Transfer Sublayer entity that receives a DATA-ONLY D_PDU **shall** ⁽³⁾ transmit an ACK-ONLY (TYPE 1) D_PDU or a DATA-ACK (TYPE 2) D_PDU as acknowledgement, where the type of D_PDU sent depends on whether or not it has C_PDUs of its own to send to the source of the DATA-ONLY D_PDU.

The DATA-ONLY D_PDU **shall** ⁽⁴⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-9 and the paragraphs below:

- C_PDU START
- C_PDU END
- DELIVER IN ORDER
- DROP C_PDU
- TX WIN UWE
- TX WIN LWE
- SIZE OF SEGMENTED C_PDU
- TRANSMIT SEQUENCE NUMBER

The C_PDU START flag **shall** ⁽⁵⁾ be set to indicate the start of a newly segmented C_PDU; the C_PDU segment contained within this D_PDU is the first segment of the C_PDU, in accordance with the C_PDU-segmentation process described in section C.4

The C_PDU END flag **shall** ⁽⁶⁾ be set to indicate the end of a segmented C_PDU; when a D_PDU is received with the C_PDU END flag set it indicates the last D_PDU that was segmented from the C_PDU. Depending on the status of the DELIVER IN ORDER flag, the link layer will assemble and deliver the C_PDU, if all D_PDUs between and including the C_PDU START and C_PDU END are received completely error free. The re-assembly process of D_PDUs into C_PDUs is described in section C.4

If the DELIVER IN ORDER flag is set on the D_PDUs composing a C_PDU, the C_PDU **shall** ⁽⁷⁾ be delivered to the upper layer when both the following conditions are met:

- 1) The C_PDU is complete.
- 2) All C_PDUs received previously that also had the DELIVER IN ORDER flag set have been delivered.

If the DELIVER IN ORDER flag is cleared on the D_PDUs composing a C_PDU, the C_PDU **shall** ⁽⁸⁾ be delivered to the upper layer when the following condition is met:

- 3) The C_PDU is complete and error free.

When the DROP C_PDU flag is set by the D_PDU source, the receiving Data Transfer Sublayer **shall** ⁽⁹⁾ discard the contents of the segmented C_PDU field of the current D_PDU and all other previously received segments of the C_PDU of which the current D_PDU is a part.

No segmented C_PDU data needs to be sent if the DROP C_PDU flag is set and the SIZE OF SEGMENTED C_PDU field **shall** ⁽¹⁰⁾ be zero in this case.

The TX WIN UWE flag **shall** ⁽¹¹⁾ be set when the TRANSMIT FRAME SEQUENCE NUMBER for the current D_PDU is equal to the Transmit Window Upper Edge (TX UWE) of the transmit-flow-control window.

Similarly, the TX WIN LWE flag **shall** ⁽¹²⁾ be set when the TRANSMIT FRAME SEQUENCE NUMBER for the current D_PDU is equal to the Transmit Lower Window Edge (LWE) of the transmit flow control window.

The SIZE OF SEGMENTED C_PDU field **shall** ⁽¹³⁾ be encoded as specified in Section C.3.2.10.

The TRANSMIT FRAME SEQUENCE NUMBER field **shall** ⁽¹⁴⁾ contain the sequence number of the current D_PDU.

The value of the TRANSMIT FRAME SEQUENCE NUMBER field **shall** ⁽¹⁵⁾ be a unique integer (modulo 256) assigned to the D_PDU during the segmentation of the C_PDU, and will not be released for reuse with another D_PDU until the receiving node has acknowledged the D_PDU.

Values for the TRANSMIT FRAME SEQUENCE NUMBER field **shall** ⁽¹⁶⁾ be assigned in an ascending modulo 256 order during the segmentation of the C_PDU.

The SEGMENTED C_PDU field **shall** ⁽¹⁷⁾ immediately follow the D_PDU header as depicted in Figure C-7. Segmented C_PDUs **shall** ⁽¹⁸⁾ be mapped according to the specification of Section C.3.2.9.

C.3.4 ACK_ONLY (TYPE 1) D_PDU (Acknowledgement of type 0, 2 data transfer)

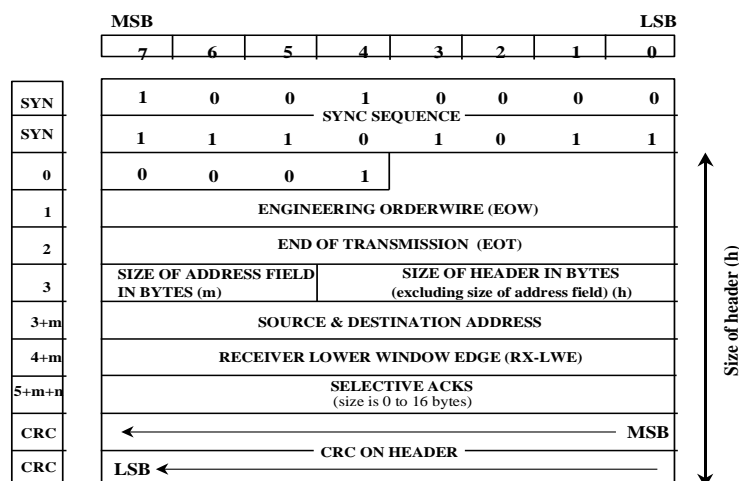


Figure C-10. Frame format for ACK-ONLY D_PDU Type 1

The ACK-ONLY D_PDU **shall** ⁽¹⁾ be used to selectively acknowledge received DATA-ONLY or DATA-ACK D_PDUs when the receiving Data Transfer Sublayer has no segmented C_PDUs of its own to send.

The ACK-ONLY D_PDU **shall** ⁽²⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-10 and the paragraphs below:

- RECEIVE LOWER WINDOW EDGE (LWE)
- SELECTIVE ACKS

The value of the RECEIVE LOWER WINDOW EDGE (RX LWE) field **shall** ⁽³⁾ equal the D_PDU sequence number (modulo 256) of the RX LWE pointer associated with the node's receive ARQ flow-control window.

The SELECTIVE ACKS field can have a dynamic length of 0 to 16 bytes, and **shall** ⁽⁴⁾ contain a bit-mapped representation of the status of all received D_PDUs with sequence numbers from the LWE to and including the UWE pointers of the receive flow-control window. The size of the SELECTIVE ACK field can be determined from knowledge of the structure of the ACK-ONLY D_PDU and the value of the Size of Header field, and is not provided explicitly in the ACK-ONLY D_PDU.

A set (1) bit within the SELECTIVE ACKS field **shall** ⁽⁵⁾ indicate a positive acknowledgement (ACK), i.e., that the D_PDU with the corresponding Frame Sequence Number was received correctly.

Only D_PDU frames with a correct segmented C_PDU CRC **shall** ⁽⁶⁾ be acknowledged positively even if the header CRC is correct.

Frames with the DROP C_PDU flag set **shall** ⁽⁷⁾ be acknowledged positively regardless of the results of the CRC check on the segmented C_PDU.

A cleared (0) bit within the SELECTIVE ACKS field **shall** ⁽⁸⁾ indicate a negative acknowledgement (NACK), i.e., that the D_PDU with the corresponding Frame Sequence Number was received incorrectly, or not at all.

The construction of the SELECTIVE ACK field and the mapping of D_PDU frame-sequence numbers to bits within the SELECTIVE ACK field **shall** ⁽⁹⁾ be in accordance with Figure C-11 and Figure C-12 and the paragraphs below.

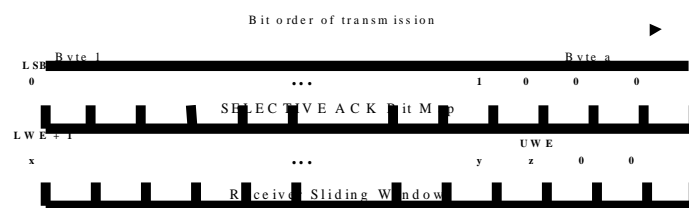


Figure C-11. Constructing the SELECTIVE ACK Field
(Note: the bits that are set (1) and cleared (0) are representing example bits.)

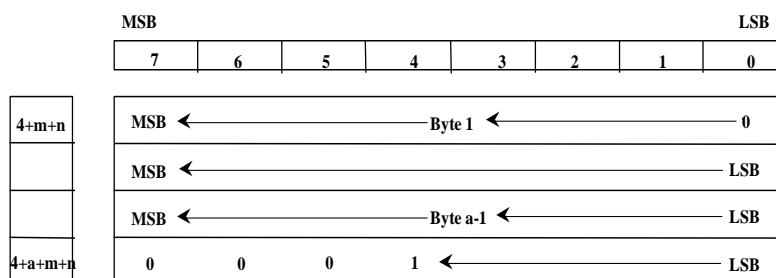


Figure C-12. SELECTIVE ACK mapping convention
(Note: the example bits that were shown in figure L-5 can be seen in this mapping example)

The LSB of the first byte of the SELECTIVE ACK field **shall** ⁽¹⁰⁾ correspond to the D_PDU whose frame sequence number is equal to 1 + the value in the RX LWE field of this ACK-ONLY D_PDU [the bit corresponding to the RX LWE is always zero, by definition, and is therefore not sent].

Each subsequent bit in the SELECTIVE ACK field **shall** ⁽¹¹⁾ represent the frame sequence number of a subsequent D_PDU in the receive flow-control sliding window, in ascending order of the respective frame-sequence numbers without omission of any values.

The bit corresponding to the Receive Upper Window Edge (RX UWE) **shall** ⁽¹²⁾ be in the last byte of the SELECTIVE ACK field.

If the bit representing the RX UWE is not the MSB of the last byte, the remaining bits in the byte (until and including the MSB) **shall** ⁽¹³⁾ be set to 0 as padding. No further SELECTIVE ACK bytes **shall** ⁽¹⁴⁾ be transmitted after such bits are required.

C.3.5 DATA-ACK (TYPE 2) D_PDU (Duplex data transfer)

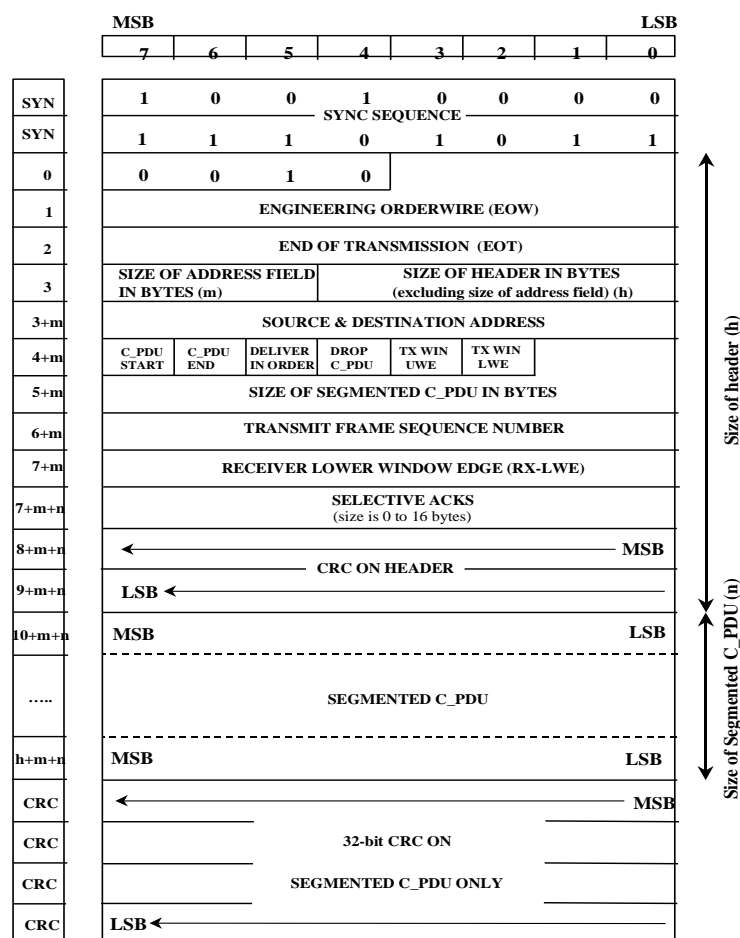


Figure C-13. Frame format for DATA-ACK D_PDU Type 2

The DATA-ACK (TYPE 2) D_PDU is a part of a basic selective automatic repeat request type of protocol. The DATA-ACK D_PDU **shall** ⁽¹⁾ be used to send segmented C_PDUs when the transmitting node needs an explicit confirmation the data was received and has received D_PDUs to selectively acknowledge.

A Data Transfer Sublayer entity that receives a DATA-ACK D_PDU **shall** ⁽²⁾ transmit an ACK-ONLY (TYPE 1) D_PDU or a DATA-ACK (TYPE 2) D_PDU as acknowledgement, where the type of D_PDU sent depends on whether or not it has C_PDUs of its own to send to the source of the DATA-ACK D_PDU.

The DATA-ACK D_PDU is a combination of the DATA-ONLY and ACK-ONLY D_PDU. All of the field specifications from the DATA-ONLY and ACK-ONLY D_PDUs apply to this D_PDU. The DATA-ACK D_PDU **shall** ⁽⁴⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-13 and the referenced paragraphs:

- C_PDU START – **shall** ⁽⁵⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPDU;
- C_PDU END – **shall** ⁽⁶⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPDU;
- DELIVER IN ORDER – **shall** ⁽⁷⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPDU;

- DROP C_PDU– **shall** ⁽⁸⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- TX WIN UWE– **shall** ⁽⁹⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- TX WIN LWE– **shall** ⁽¹⁰⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- SIZE OF SEGMENTED C_PDU– **shall** ⁽¹¹⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- TRANSMIT SEQUENCE NUMBER– **shall** ⁽¹²⁾ be as specified in Section 3.3 for the DATA-ONLY D_DPU;
- RECEIVE LOWER WINDOW EDGE (RX LWE) – **shall** ⁽¹³⁾ be as specified in Section 3.4 for the ACK-ONLY D_DPU;
- SELECTIVE ACKS– **shall** ⁽¹⁴⁾ be as specified in Section 3.4 for the ACK-ONLY D_DPU;

C.3.6 RESET/WIN_RESYNC (TYPE 3) D_PDU (Reset/Re-synchronise peer protocol entities)

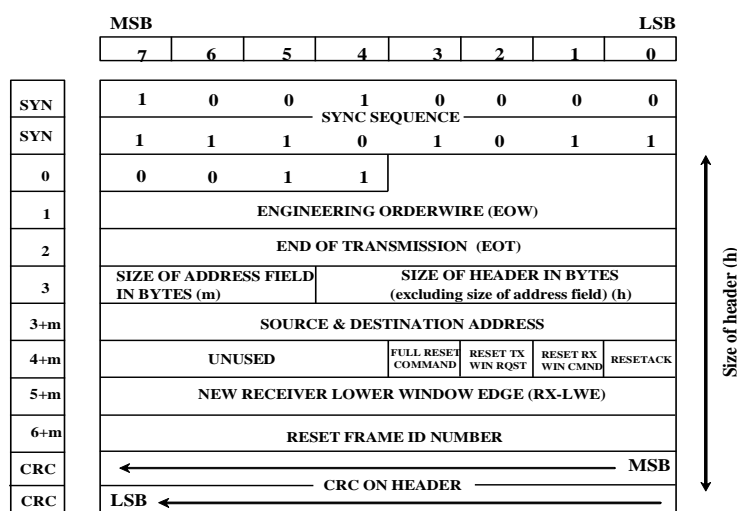


Figure C-14. Frame format for RESET/WIN RESYNC D_PDU Type 3

The RESET/WIN RESYNC D_PDU **shall** ⁽¹⁾ be used to control the re-synchronisation or re-initialisation of the selective-repeat ARQ protocol operating on the link between the source and destination nodes.

Reset and resynchronization operations **shall** ⁽²⁾ be performed with respect to the transmit lower-window edge (TX LWE) and the receive lower-window edge (RX LWE) for the flow-control sliding windows at the sending node and receiving node, as specified in Section C.6.2.

The RESET/WIN RESYNC D_PDU **shall** ⁽³⁾ use a basic stop and wait type of protocol (denoted as the IRQ protocol elsewhere in this STANAG) in which the reception of this D_PDU **shall** ⁽⁴⁾ result in the transmission of an acknowledgement D_PDU by the receiving node. For the IRQ protocol used with the RESET/WIN RESYNC D_PDU, the RESET/WIN RESYNC D_PDU is used for both data and acknowledgement, as specified below.

Transmission of D_PDUs supporting the regular-data service, i.e., of DATA (Type 0), ACK (Type 1), and DATA-ACK (Type 2) D_PDUs, **shall**⁽⁵⁾ be suspended pending completion of any stop-and-wait protocol using the RESET/WIN RESYNC D_PDUs. [Note: The ARQ windowing variables will be reset to zero or changed as a result of the use of the RESET/WIN RESYNC D_PDUs, and suspension of normal data transmission until these variables have been reset is required, lest the variables controlling the selective-repeat ARQ protocol become even further corrupted than the error-state that presumably triggered the reset or resynchronization protocol.]

The RESET/WIN RESYNC D_PDU **shall**⁽⁶⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-14 and the paragraphs below:

- FULL RESET COMMAND
- RESET TX WIN RQST
- RESET RX WIN CMND
- RESET ACK
- NEW RECEIVE LOWER WINDOW EDGE (LWE)
- RESET FRAME ID NUMBER

The FULL RESET COMMAND flag **shall**⁽⁷⁾ be set equal to one (1) to force a full reset of the ARQ machines at the transmitter and receiver to their initial values as specified in Sections C.6.2 and C.6.3.

A Type 3 D_PDU with the RESET TX WIN RQST flag set equal to one (1) **shall**⁽⁸⁾ be used to request a resynchronisation of the TX-LWE and RX-LWE pointers used for DATA in the transmit and receive nodes.

A node that receives a Type 3 D_PDU with the RESET TX WIN RQST flag set equal to one **shall**⁽⁹⁾ respond by forcing resynchronization of the windows using a RESET/WIN RESYNC D_PDU and the RESET RX WIN CMND flag, as specified below.

A RESET/WIN RESYNC TYPE 3 D_PDU with the RESET RX WIN CMND flag set equal to one (1) **shall**⁽¹⁰⁾ be used to force a resynchronisation of the TX-LWE pointer at the sending node and the RX-LWE pointer at the receiving node.

A node that sends a Type 3 D_PDU with the RESET RX WIN CMND flag set equal to one **shall**⁽¹¹⁾ proceed as follows:

- The NEW RECEIVE LWE field **shall**⁽¹²⁾ be set equal to the value of the sending node's TX-LWE.
- The sending node **shall**⁽¹³⁾ wait for a RESET/WIN RESYNC Type 3 D_PDU with the RESET ACK flag set equal to one as an acknowledgement that the resynchronization has been performed.

A node that receives a Type 3 D_PDU with the RESET RX WIN CMND flag set equal to one **shall**⁽¹⁴⁾ proceed as follows:

- The value of the node's TX LWE **shall**⁽¹⁵⁾ be set equal to the value of the NEW RECEIVE LWE field in the RESET/WIN RESYNC D_PDU that was received;
- The node **shall**⁽¹⁶⁾ send a RESET/WIN RESYNC Type 3 D_PDU with the RESET ACK flag set equal to one as an acknowledgement that the resynchronization has been performed.

The RESET ACK flag **shall**⁽¹⁷⁾ be set equal to one (1) to indicate an acknowledgement of the most recently received RESET/WIN RESYNC TYPE 3 D_PDU.

A node may use a RESET/WIN RESYNC acknowledgement transmission, i.e., a Type 3 D_PDU with the RESET ACK flag set equal to one (1), to request/force a reset or resynchronization of its own ARQ flow control variables, e.g., if the link is supporting reliable duplex communication and the ARQ machines for both directions of data-flow must be reset or resynchronized.

The NEW RECEIVE LWE field specifies the value of the new receiver ARQ RX-LWE, as noted above, and **shall** ⁽¹⁸⁾ be valid only when the value of the RESET RX WIN CMND flag equals one (1). The value of the NEW RECEIVE LWE field **shall** ⁽¹⁹⁾ be ignored in any other situation.

The Data Transfer Sublayer **shall** ⁽²⁰⁾ use the RESET FRAME ID NUMBER field to determine if a given RESET/WIN RESYNC D_PDU received is a copy of one already received.

The value of the RESET FRAME ID NUMBER field **shall** ⁽²¹⁾ be a unique integer (modulo 256) assigned in ascending order to RESET/WIN RESYNC D_PDUs, and will not be released for reuse with another D_PDU until the D_PDU to which it was assigned has been acknowledged.

[Implementation Note: Due to the fact that C_PDUs have a certain time-to-live (TTL) assigned to them it is possible that this time passes when the C_PDU is partially transmitted and acknowledged by the transmitting node. Deleting such partially transferred C_PDU can be performed by two methods:

- a) The first method makes use of a D_PDU frame's DROP C_PDU flag, setting this flag will indicate to the receiving node that the current D_PDU and all other previously received portions of the C_PDU have to be discarded by the link layer. This method still requires the transmission and acknowledgement of all D_PDU headers only until the dropped C_PDU has been completely received and acknowledged.
- b) The second method makes use of the RESET/WIN RESYNC D_PDU, sending this D_PDU will enable the transmitter to skip the discarded C_PDU or multiple C_PDUs without the need of sending all D_PDU headers. It is also possible to RESET the Selective ARQ machine by sending a FULL RESET COMMAND. This will put the ARQ machine back into its starting position upon the creation of a physical link. The FULL RESET COMMAND will not delete one or more selected C_PDUs but will delete all partially received and acknowledged C_PDUs still waiting to be completed.]

C.3.7 EXPEDITED DATA ONLY (TYPE 4) D_PDU (Simplex expedited data transfer)

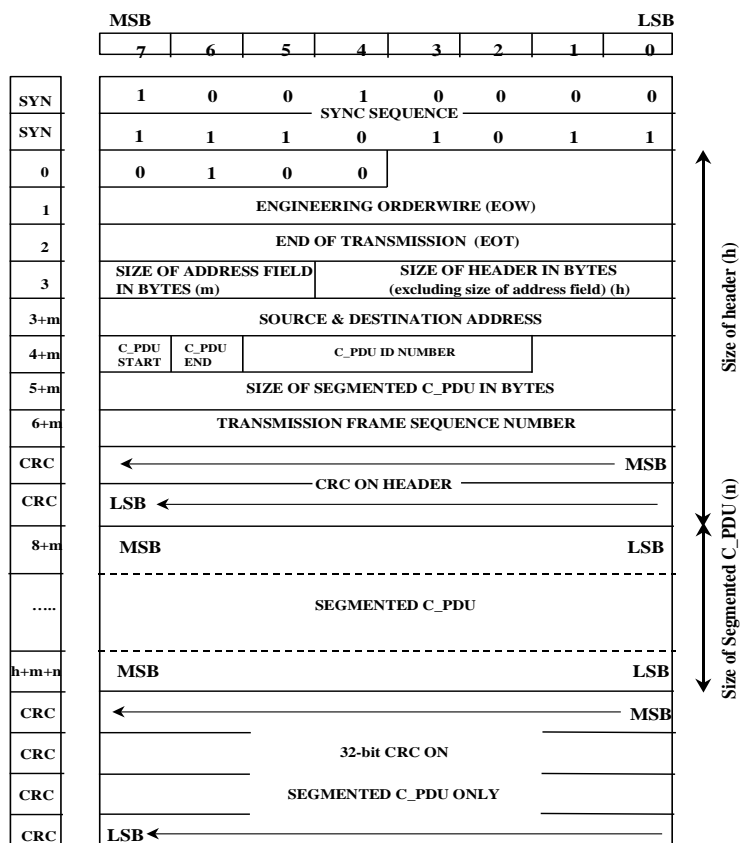


Figure C-15. Frame format for EXPEDITED DATA ONLY D_PDU Type 4

The EXPEDITED DATA-ONLY (TYPE 4) D_PDU **shall** ⁽¹⁾ be used to send segmented C_PDUs that require Expedited Delivery Service when the transmitting node needs an explicit confirmation the data was received.

A Data Transfer Sublayer entity that receives EXPEDITED DATA-ONLY (TYPE 4) D_PDU **shall** ⁽²⁾ send an EXPEDITED DATA-ONLY (TYPE 5) D_PDU as a selective acknowledgement of all EXPEDITED DATA-ONLY (TYPE 4) D_PDUs received from the source node.

The EXPEDITED DATA-ONLY D_PDU is similar in structure to the DATA-ONLY D_PDU. The EXPEDITED DATA-ONLY D_PDU **shall** ⁽³⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-15 and the paragraphs noted:

- C_PDU START – **shall** ⁽⁴⁾ be as specified for the DATA-ONLY D_PDU in Section C.3.3
- C_PDU END – **shall** ⁽⁵⁾ be as specified for the DATA-ONLY D_PDU in Section C.3.3
- C_PDU ID NUMBER – **shall** ⁽⁶⁾ be as specified in the paragraphs below.
- SIZE OF SEGMENTED C_PDU – **shall** ⁽⁷⁾ be as specified in Section C.3.2.10 for all D_PDUs that have a Segmented C_PDU field.
- TRANSMIT SEQUENCE NUMBER – **shall** ⁽⁸⁾ be as specified for the DATA-ONLY D_PDU in Section C.3.3, with additional requirements as noted below.

The C_PDU ID NUMBER field **shall** ⁽⁶⁾ specify the C_PDU of which this Expedited D_PDU is a part. The value of the C_PDU ID NUMBER field **shall** ⁽⁷⁾ be an integer (modulo 16) assigned in an ascending modulo 16 order to the C_PDU, and **shall** ⁽⁸⁾ not be released for reuse with another C_PDU until the entire C_PDU has been acknowledged.

As noted above, the TRANSMIT SEQUENCE NUMBER field in the EXPEDITED DATA-ONLY D_PDU is defined and used in the same manner as that specified for the DATA-ONLY D_PDU. However, the EXPEDITED DATA-ONLY D_PDUs **shall** ⁽⁹⁾ be assigned frame numbers from a frame sequence number pool (0, 1 ...255) that is reserved exclusively for the transmission of EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs. The FRAME SEQUENCE NUMBER is used, in this D_PDU, to sequence the D_PDUs that make up a C_PDU receiving expedited delivery service.

(Note: the further implication of this requirement is that there are independent state machines and flow-control windows [or different states and sets of variables within a single state-machine] for the Expedited and Regular delivery services in the Data Transfer Sublayer).

The SEGMENTED C_PDU field is a field that is attached to the header structure defined in Figure C-15. The segmented PDU **shall** ⁽¹⁰⁾ immediately follow the D_PDU header. Segmented C_PDUs **shall** ⁽¹¹⁾ be mapped according to the convention described in C.3.2.9.

The processing of EXPEDITED D_PDUs in the EXPEDITED DATA state **shall** ⁽¹¹⁾ differ from the processing of DATA-ONLY or DATA-ACK D_PDUs in the DATA state in the following ways:

- Data (i.e., C_PDUs) using the Expedited Delivery Service **shall** ⁽¹²⁾ be transferred using EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs. If duplex communication is required, EXPEDITED DATA-ONLY and EXPEDITED ACK-ONLY D_PDUs may be placed together in a transmission interval.
- C_PDUs requiring Expedited Delivery Service and the associated EXPEDITED D_PDUs **shall** ⁽¹³⁾ not be queued for processing within the Data Transfer Sublayer behind D_PDUs containing non-expedited data (i.e., DATA-ONLY or DATA-ACK D_PDUs).

C.3.8 EXPEDITED ACK ONLY (TYPE 5) D_PDU (Acknowledgement of simplex expedited data transfer)

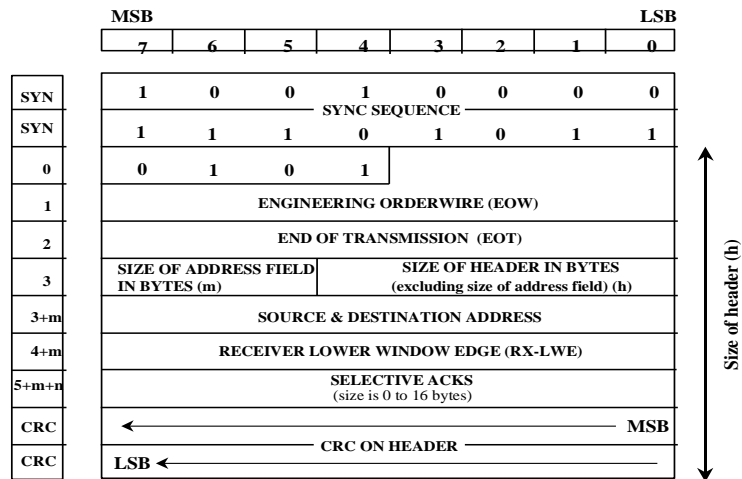


Figure C-16. Frame format for EXPEDITED ACK ONLY D_PDU Type 5

The EXPEDITED ACK-ONLY (TYPE 5) D_PDU **shall** ⁽¹⁾ be used to selectively acknowledge received EXPEDITED DATA-ONLY D_PDUs.

The EXPEDITED ACK-ONLY (TYPE 5) D_PDU type **shall** ⁽²⁾ have the same format as the ACK-ONLY (TYPE 1) D_PDU, differing only in the value of the D_PDU Type field in byte 0, as specified in Figure C-16.

C.3.9 MANAGEMENT (TYPE 6) D_PDU (Management message transfer)

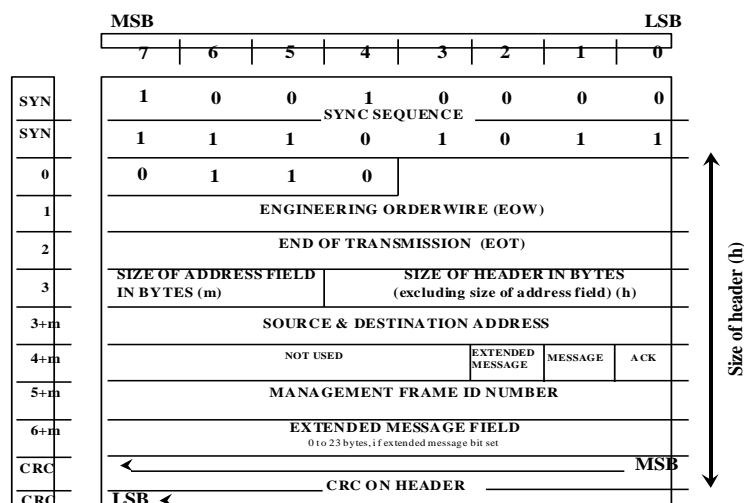


Figure C-17. Header format for MANAGEMENT D_PDU TYPE 6

The MANAGEMENT (TYPE 6) D_PDU **shall** ⁽¹⁾ be used to send EOW Messages or Management Protocol Data Units (M_PDUs) when the transmitting node needs an explicit Acknowledgement that they were received.

A Data Transfer Sublayer entity **shall** ⁽²⁾ acknowledge receipt of a MANAGEMENT (TYPE 6) D_PDU by sending a MANAGEMENT (TYPE 6) D_PDU with the ACK flag set to the value one (1).

The processing and transmission of MANAGEMENT (TYPE 6) D_PDUs **shall** ⁽³⁾ take precedence over and bypass all other pending D_PDU types in the Data Transfer Sublayer.

The exchange of MANAGEMENT D_PDUs is regulated by a stop-and-wait protocol, i.e., there **shall** ⁽⁴⁾ be only one unacknowledged MANAGEMENT D_PDU at any time.

The MANAGEMENT D_PDU **shall** ⁽⁵⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-17 and the paragraphs below:

- EXTENDED MESSAGE Flag
- VALID MESSAGE
- ACK
- MANAGEMENT FRAME ID NUMBER
- EXTENDED MANAGEMENT MESSAGE

The VALID MESSAGE field **shall** ⁽⁶⁾ be set to the value one (1) if the EOW field of the D_PDU contains a valid Management message or the initial segment of a valid Management message that is continued in the EXTENDED MANAGEMENT MESSAGE field.

The VALID MESSAGE field **shall** ⁽⁷⁾ be set to the value zero (0) if the EOW field contains an Engineering Orderwire Message for which an acknowledgement message is not required. If the VALID MESSAGE field is set to zero, the MANAGEMENT D_PDU **shall** ⁽⁸⁾ be used only to acknowledge receipt of another MANAGEMENT D_PDU.

The EXTENDED MESSAGE Flag **shall** ⁽⁹⁾ be set to the value one (1) if the D_PDU contains a non-zero, non-null EXTENDED MANAGEMENT MESSAGE field.

If the EXTENDED MESSAGE Flag is set to the value zero (0), the EXTENDED MANAGEMENT MESSAGE field **shall** ⁽¹⁰⁾ not be present in the MANAGEMENT D_PDU.

The MANAGEMENT FRAME ID NUMBER field **shall** ⁽¹¹⁾ contain an integer in the range [0,255] with which MANAGEMENT D_PDUs **shall** ⁽¹²⁾ be identified.

The Data Transfer Sublayer **shall** ⁽¹³⁾ maintain variables to manage the frame ID numbers associated with this D_PDU:

- the TX MANAGEMENT FRAME ID NUMBER **shall** ⁽¹⁴⁾ maintain the value of the Frame ID Number for MANAGEMENT D_PDUs that are transmitted;
- the RX MANAGEMENT FRAME ID NUMBER **shall** ⁽¹⁵⁾ maintain the value of the Frame ID Number for the most recently received MANAGEMENT D_PDUs.

On initialisation (such as a new connection), a node's Data Transfer Sublayer **shall** ⁽¹⁶⁾ set its current TX MANAGEMENT FRAME ID NUMBER to zero and **shall** ⁽¹⁷⁾ set its current RX MANAGEMENT FRAME ID NUMBER to an out-of-range value (i.e., a value greater than 255).

The current value of the TX MANAGEMENT FRAME ID NUMBER **shall** ⁽¹⁸⁾ be placed in the appropriate field of each unique MANAGEMENT D_PDU transmitted. The current value of the TX MANAGEMENT FRAME ID NUMBER **shall** ⁽¹⁹⁾ be incremented by one, modulo 256, after each use, unless transmission of repeated copies of the MANAGEMENT D_PDU are specified for its use, e.g., as in Section C.6.4.2.

Management D_PDUs that have been repeated (e.g., in accordance Section C.6.4.2) **shall** ⁽²⁰⁾ have the same MANAGEMENT FRAME ID NUMBER.

The Data Transfer Sublayer **shall** ⁽²¹⁾ compare the MANAGEMENT FRAME ID NUMBER of received MANAGEMENT D_PDUs to the current RX MANAGEMENT FRAME ID NUMBER, and process them as follows:

- if the MANAGEMENT FRAME ID NUMBER in the received D_PDU differs from the current RX MANAGEMENT FRAME ID NUMBER value, the D_PDU **shall** ⁽²²⁾ be treated as a new D_PDU, and the Data Transfer Sublayer **shall** ⁽²³⁾ set the current RX MANAGEMENT FRAME ID NUMBER value equal to the value of the received MANAGEMENT FRAME ID NUMBER.
- if the value in the received D_PDU is equal to the current RX MANAGEMENT FRAME ID NUMBER value, the node **shall** ⁽²⁴⁾ assume that the frame is a repetition of a MANAGEMENT D_PDU that has already been received, and the value of the current RX MANAGEMENT FRAME ID NUMBER **shall** ⁽²⁵⁾ be left unchanged.

There **shall** ⁽²⁶⁾ be a one-to-one correspondence between MANAGEMENT messages and MANAGEMENT D_PDUs; that is, each message is placed into a separate D_PDU (which may be repeated a number of times as specified in Section C.6.4).

The 12-bit EOW section of the D_PDU **shall** ⁽²⁷⁾ carry the EOW (non-extended) MANAGEMENT message, as specified in Section C.5.

The EXTENDED MANAGEMENT MESSAGE field may be used to transmit other implementation-specific messages that are beyond the scope of this STANAG. When the EXTENDED MESSAGE field is present and in use, the EXTENDED MESSAGE Flag **shall** ⁽²⁸⁾ be set to the value one (1).

C.3.10 NON-ARQ (TYPE 7) DATA D_PDU (Non-ARQ data transfer)

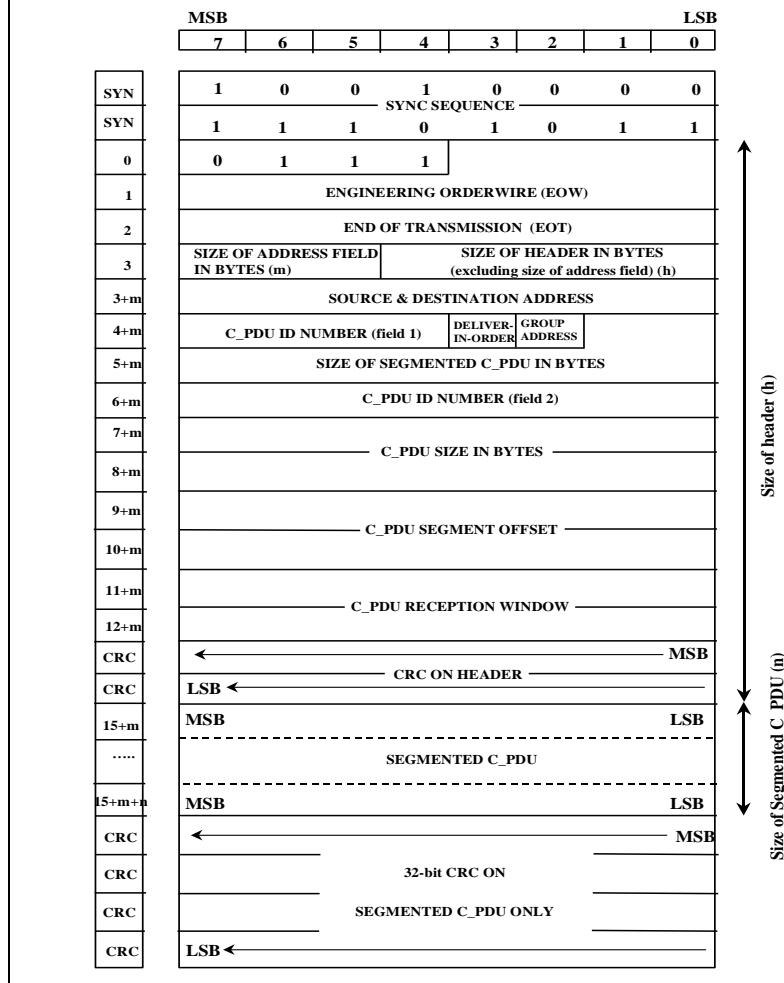


Figure C-18. Frame format for NON-ARQ DATA D_PDU TYPE 7

The NON-ARQ DATA (TYPE 7) D_PDU **shall** ⁽¹⁾ be used to send segmented C_PDUs when the transmitting node needs no explicit confirmation the data was received.

The NON-ARQ DATA D_PDU **shall** ⁽²⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-18 and the paragraphs below:

- C_PDU ID NUMBER (field 1)
- DELIVER IN ORDER
- GROUP ADDRESS
- SIZE OF SEGMENTED C_PDU
- C_PDU ID NUMBER (field 2)
- C_PDU SIZE
- C_PDU SEGMENT OFFSET
- C_PDU RECEPTION WINDOW

The C_PDU ID NUMBER field **shall** ⁽³⁾ identify the C_PDU to which the C_PDU segment encapsulated by the NON-ARQ DATA D_PDU belongs.

The value encoded in the C_PDU ID NUMBER field **shall** ⁽⁴⁾ be a unique integer (modulo 4096) identifier assigned in an ascending order (also modulo 4096) to the C_PDU during its segmentation and encapsulation into D_PDUs.

The value encoded in the C_PDU ID NUMBER field **shall** ⁽⁵⁾ not be released for reuse and assignment to another C_PDU until the time specified in the C_PDU RECEPTION WINDOW expires, as noted below.

The C_PDU ID NUMBER space (i.e, the set of ID numbers in the range [0..4095]) for NON-ARQ DATA (TYPE 7) D_PDUs **shall** ⁽⁶⁾ be different than the similarly-defined number space for EXPEDITED NON-ARQ DATA (TYPE 8) D_PDUs.

The value of the C_PDU ID NUMBER **shall** ⁽⁷⁾ be encoded in a 12 bit field as specified in Figure C-19.

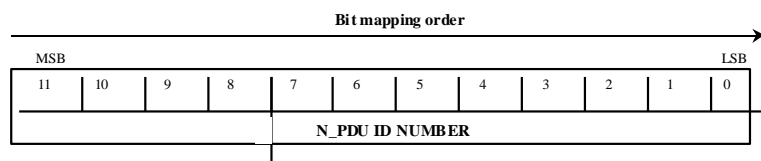


Figure C-19. C_PDU ID NUMBER Field

The value of the C_PDU ID NUMBER **shall** ⁽⁸⁾ be mapped into the NON-ARQ DATA D_PDU into two split fields as follows, and as depicted in Figure C-20:

- The four-most-significant bits of the value of the C_PDU ID NUMBER **shall** ⁽⁹⁾ be mapped into C_PDU ID NUMBER (field 1);
- The eight least-significant bits of the value of the C_PDU ID NUMBER **shall** ⁽¹⁰⁾ be mapped into C_PDU ID NUMBER (field 2).

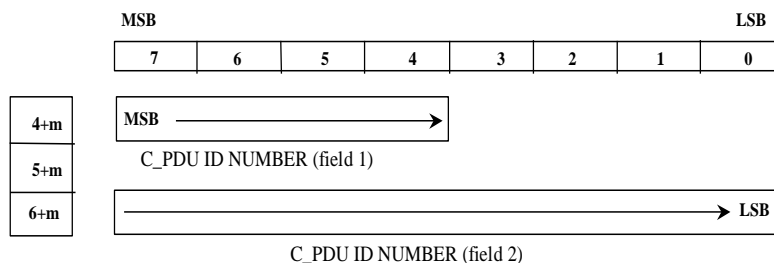


Figure C-20. C_PDU ID NUMBER mapping convention in D_PDU header

If the DELIVER IN ORDER flag is set (i.e., its value equals 1) on the D_PDUs composing a C_PDU, the C_PDU **shall** ⁽¹¹⁾ be delivered to the network layer when both the following conditions are met:

- 1) C_PDU is complete and error free or the C_PDU RECEPTION WINDOW expires.

- 2) Previous C_PDUs that also had the DELIVER IN ORDER flag set have been delivered.

If the DELIVER IN ORDER flag is cleared (0) on the D_PDUs composing a C_PDU, the C_PDU **shall** ⁽¹²⁾ be delivered to the network layer when the following condition is met.

- 1) C_PDU is complete and error free or the C_PDU RECEPTION WINDOW expires.

The GROUP ADDRESS flag **shall** ⁽¹³⁾ indicate that the destination address should be interpreted as a group address rather than an individual address, as follows:

- The destination address **shall** ⁽¹⁴⁾ be interpreted as a group address when the GROUP ADDRESS flag is set (1).
- However when the GROUP ADDRESS flag is cleared (0) the destination address **shall** ⁽¹⁵⁾ be interpreted as an individual node address.

Note: If a specific PDU is intended for only one node, the node's normal address may also be used with the NON-ARQ DATA D_PDU. Group addresses are intended to allow PDUs to be addressed to specific groups of nodes when using NON-ARQ DATA D_PDUs. The use of a bit to designate a "group address" allows the same number (~268 million!) and structure of group addresses to be designated as for normal addresses, rather than requiring some portion of the total address space for group addresses. The management of group addresses is outside of the scope of this STANAG.

The SIZE OF SEGMENTED C_PDU field **shall** ⁽¹⁶⁾ specify the number of bytes contained in the SEGMENTED C_PDU file in accordance with the requirements of Section C.2.10.

The C_PDU SIZE field **shall** ⁽¹⁷⁾ indicate the size in bytes of the C_PDU of which the C_PDU segment encapsulated in this D_PDU is a part.

The value of the C_PDU SIZE field **shall** ⁽¹⁸⁾ be encoded in a 16 bit field, with the bits mapped as specified by Figure C-21. The number **shall** ⁽¹⁹⁾ be mapped into the D_PDU by placing the MSB of the field into the MSB of the first byte in the D_PDU as can be seen in Figure C-22.

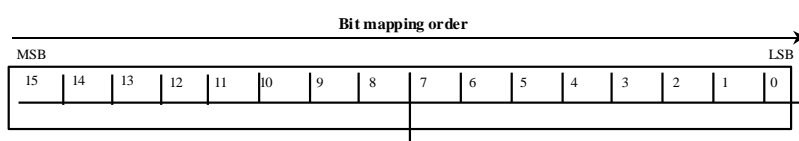


Figure C-21. C_PDU SIZE Field

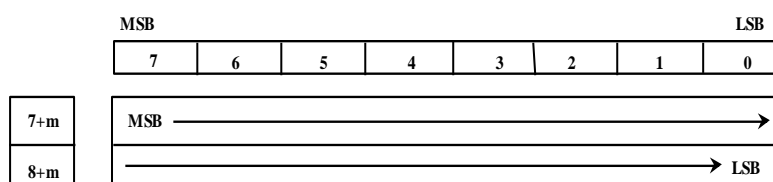


Figure C-22. C_PDU SIZE mapping convention in D_PDU header

The C_PDU SEGMENT OFFSET field **shall** ⁽²⁰⁾ indicate the location of the first byte of the SEGMENTED C_PDU with respect to the start of the C_PDU. For the purposes of this field, the

bytes of the C_PDU **shall** ⁽²¹⁾ be numbered consecutively starting with 0.

The C_PDU SEGMENT OFFSET field is a 16 bit field, the bits **shall** ⁽²²⁾ be mapped as specified by Figure C-23. The number **shall** ⁽²³⁾ be mapped into the D_PDU by placing the MSB of the field into the MSB of the first byte in the D_PDU as specified in Figure C-24.

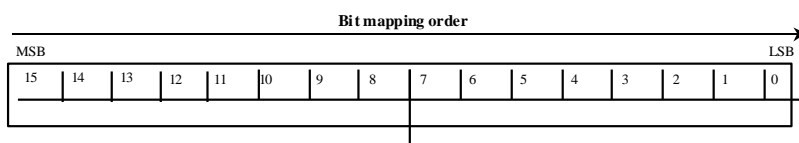


Figure C-23. C_PDU SEGMENT OFFSET Field

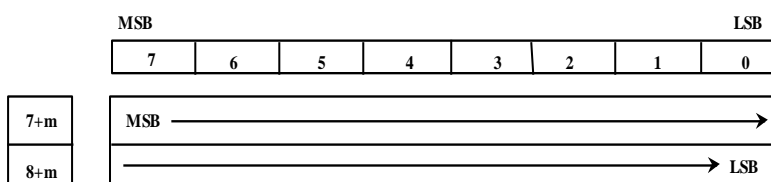


Figure C-24. C_PDU SEGMENT OFFSET mapping convention in D_PDU header

The C_PDU RECEPTION WINDOW field **shall** ⁽²⁴⁾ indicate the maximum remaining time in units of half (1/2) seconds relative to the start of the D_PDU during which portions of the associated C_PDU may be received.

As in the case of the EOT field, the C_PDU RECEPTION WINDOW **shall** ⁽²⁵⁾ be updated just prior to transmitting each D_PDU. The receiving node can use this information to determine when to release a partially received C_PDU. (The transmitter is not allowed to transmit D_PDU's that are a portion of a C_PDU when the C_PDU RECEPTION WINDOW has expired).

The value of the C_PDU RECEPTION WINDOW field **shall** ⁽²⁶⁾ be encoded in a 16 bit field with the bits be mapped as specified by Figure C-25. The value **shall** ⁽²⁷⁾ be mapped into the D_PDU by placing the MSB of the field into the MSB of the first byte in the D_PDU as specified in Figure C-26.

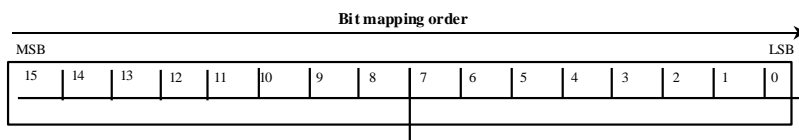


Figure C-25. C_PDU RECEPTION WINDOW Field

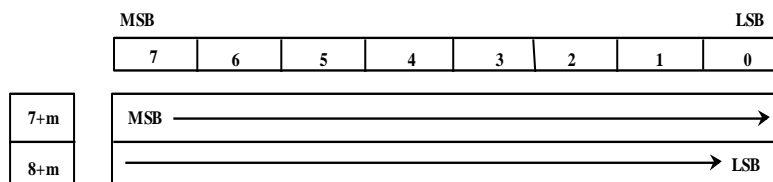


Figure C-26. C_PDU RECEPTION WINDOW mapping convention in D_PDU header

C.3.11 EXPEDITED NON-ARQ DATA (TYPE 8) D_PDU (Expedited non-ARQ data transfer)

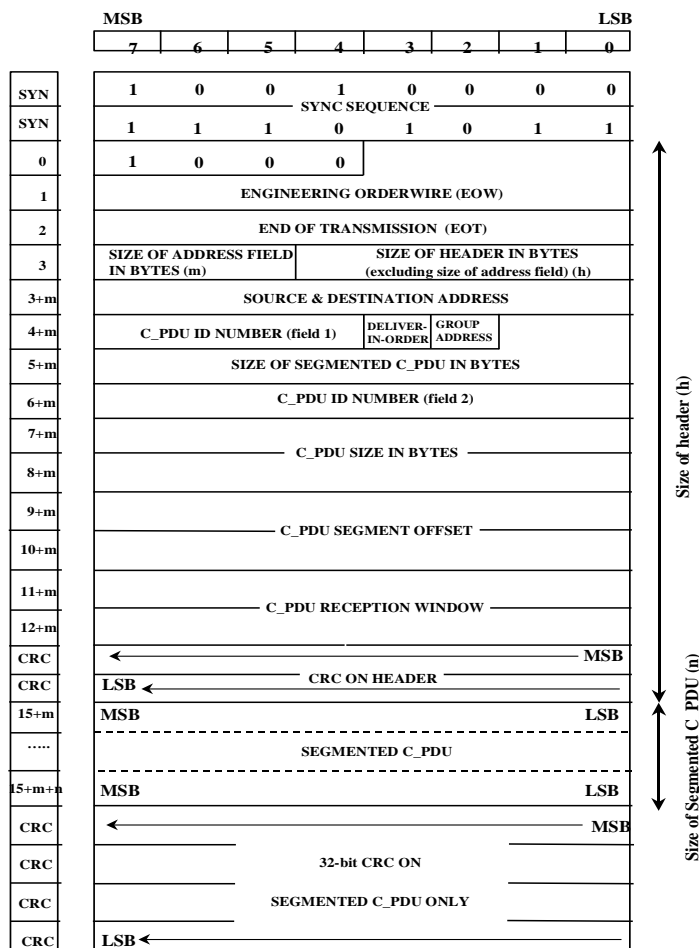


Figure C-27. Frame format for EXPEDITED NON-ARQ DATA D_PDU Type 8

The frame format for EXPEDITED NON-ARQ DATA (TYPE 8) D_PDUs **shall** ⁽¹⁾ be identical to the NON_ARQ DATA D_PDU with the exception that the TYPE field has a value of 8, as specified in Figure C-27.

The C_PDU ID NUMBER space (i.e, the set of ID numbers in the range [0..4095]) for EXPEDITED NON-ARQ DATA (TYPE 8) D_PDUs **shall** ⁽²⁾ be different than the similarly-defined number space for NON-ARQ DATA (TYPE 7) D_PDUs.

C.3.12 WARNING (TYPE 15) D_PDU (in response to unexpected or unrecognised D_PDU type)

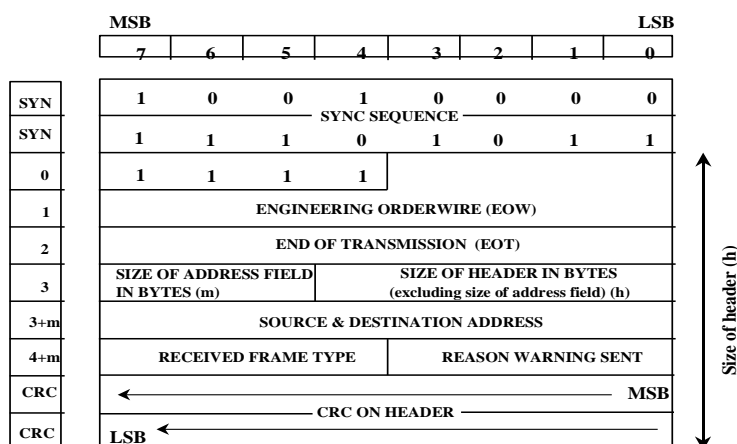


Figure C-28. Frame format for WARNING D_PDU Type 15

A Data Transfer Sublayer **shall** ⁽¹⁾ a WARNING D_PDU to any remote node from which an unexpected or unknown D_PDU type has been received.

The WARNING D_PDU **shall** ⁽²⁾ contain the following fields within its D_PDU Type-Specific part, mapped and encoded in accordance with Figure C-28 and the paragraphs below:

- RECEIVED FRAME TYPE
- REASON WARNING SENT

The RECEIVED FRAME TYPE field **shall** ⁽³⁾ indicate the frame type that caused the warning to be sent.

The value of the RECEIVED FRAME TYPE field **shall** ⁽⁴⁾ be encoded in four bits, and located within the D_PDU as specified in Figure C-29 and Figure C-30.

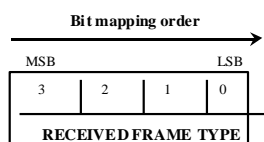


Figure C-29. RECEIVED FRAME TYPE Field

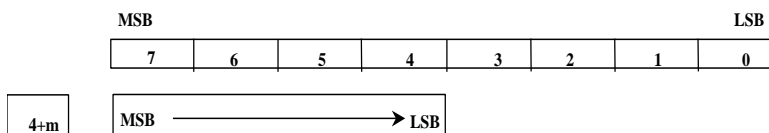


Figure C-30. RECEIVED FRAME TYPE mapping convention in D_PDU header

The REASON WARNING SENT field **shall** ⁽⁵⁾ indicate the reason the frame type caused a warning, with values as Specified in Table C-3:

Table C-3: Encoding of WARNING D_PDU Reason Field

| Reason | Field Value |
|---|-------------|
| Unrecognised D_PDU type Received | 0 |
| Connection-related D_PDU Received While Not Currently Connected | 1 |
| Invalid D_PDU Received | 2 |
| Invalid D_PDU Received for Current State | 3 |
| <i>Unspecified/reserved</i> | 4-15 |

The value of the REASON WARNING SENT field **shall** ⁽⁶⁾ be encoded in four bits, and located within the D_PDU as specified in Figure C-31 and Figure C-32.

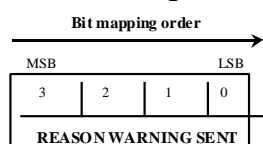


Figure C-31. REASON WARNING SENT Field

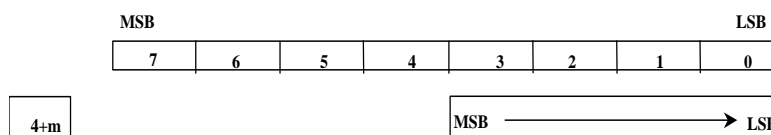


Figure C-32. REASON WARNING SENT mapping convention in D_PDU header

The transmission of WARNING type D_PDUs **shall** ⁽⁷⁾ be initiated independently by the Data Transfer Sublayer in response to certain D_PDUs and **shall** ⁽⁸⁾ not be acknowledged explicitly.

WARNING type D_PDUs may be inserted into an ongoing transmission interval provided that the transmission interval period is not exceeded.

A WARNING D_PDU **shall** ⁽⁹⁾ be sent in the following conditions:

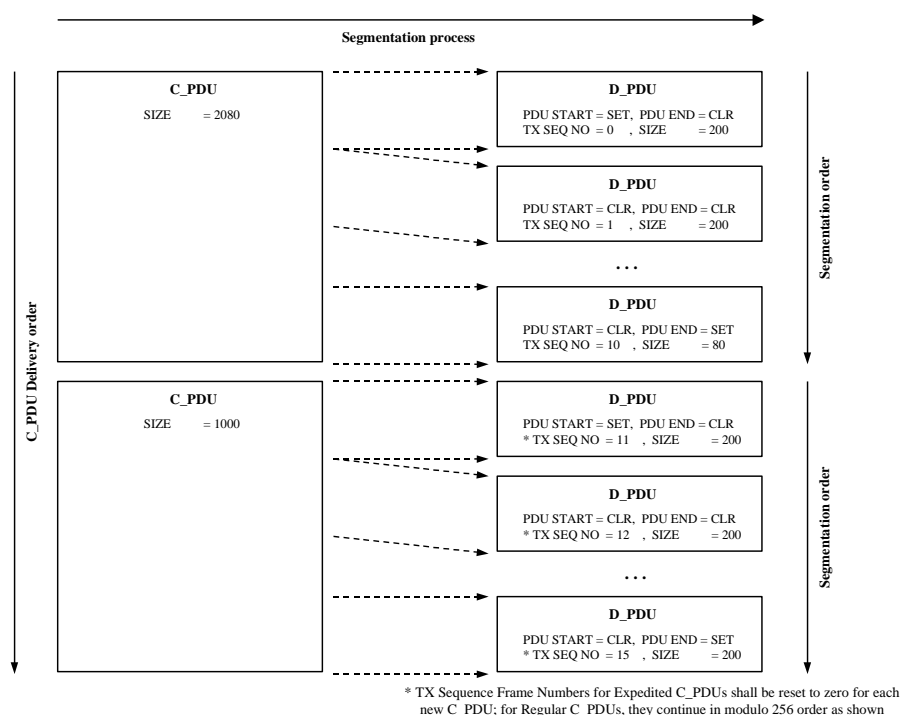
1. A node receives a D_PDU header addressed to itself with a valid CRC and an unrecognised D_PDU type (value 0000)
2. A node is not in the IDLE/BROADCAST state and it receives a D_PDU header addressed to itself, from a node with which it is not currently connected. (value 0001)
3. A node is in IDLE/BROADCAST state and it receives a D_PDU header addressed to itself which is other than type 7 or type 8 D_PDU (value 0010)
4. A node receives any D_PDU which is recognized but is not of the allowed type for the state which the receiving node is in (value 0011; this is the general case of the preceding)

A WARNING D_PDU **shall** ⁽¹⁰⁾ not be sent in response to receipt of a WARNING D_PDU.

C.4 C_PDU Segmentation and Re-assembly Processes

The process of C_PDU segmentation and re-assembly **shall** ⁽¹⁾ be as defined in the subsections that follow for ARQ and non-ARQ delivery services provided to regular and expedited C_PDUs.

C.4.1 ARQ Mode Segmentation and Re-assembly

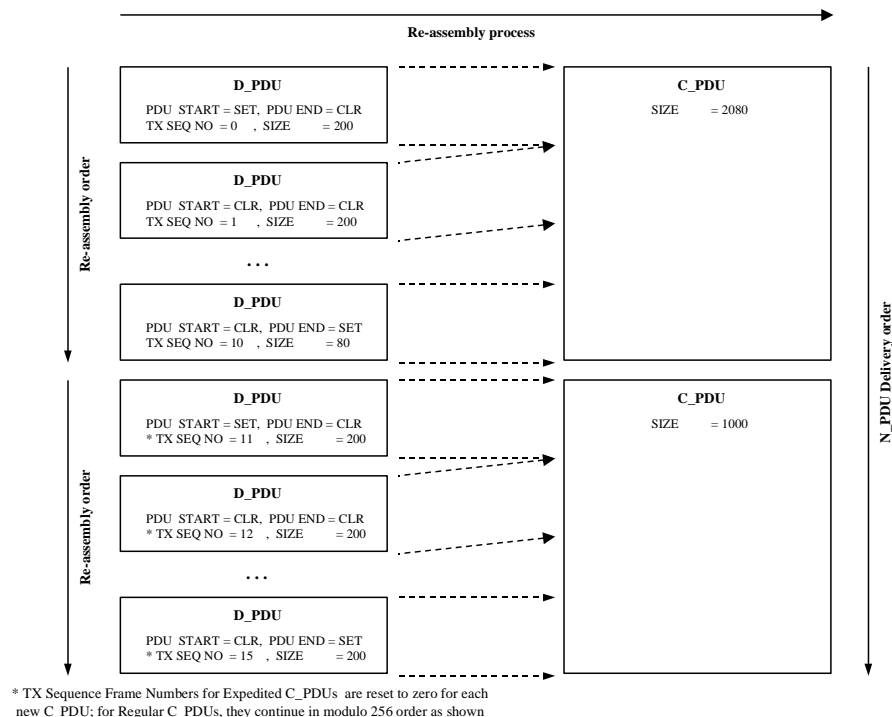


**Figure C-33. C_PDU Segmentation for ARQ Delivery Services
(Regular and Expedited Service)**

Segmentation of a C_PDU into segments small enough to fit within a D_PDU for ARQ-delivery (i.e., a DATA, DATA-ACK, or EXPEDITED-DATA D_PDU) **shall** ⁽¹⁾ be performed in accordance with the example shown in Figure C-33 and as follows:

- The Maximum C_PDU-Segment Size within a D_PDU for ARQ-Delivery services **shall** ⁽²⁾ be a configurable parameter no greater than 1023 bytes in any implementation compliant with this STANAG. An implementation may configure the Maximum C_PDU-Segment Size to match the interleaver size for optimum channel efficiency or other reasons.
- An entire C_PDU that is smaller than the Maximum C_PDU-Segment Size **shall** ⁽³⁾ be placed in the C_PDU segment of a single D_PDU.
- A DATA or DATA-ACK, or EXPEDITED D_PDU that contains an entire C_DPU **shall** ⁽⁴⁾ be marked with the both C_PDU START field and the C_PDU END field set equal to the value "1". [Note: An 'only' C_PDU segment is both the "first" and "last" segment of a sequence of one.]

- d. The Data Transfer Sublayer **shall** ⁽⁵⁾ divide C_PDU's larger than the Maximum C_PDU-Segment Size into segments that are no larger than the Maximum C_PDU Segment Size.
- e. Only the last segment or the only segment taken from a C_PDU may be smaller than the Maximum C_PDU-Segment size. A C_PDU smaller than the Maximum C_PDU-Segment size **shall** ⁽⁶⁾ be placed only in the D_PDU that contains the last segment of the C_PDU, i.e., only in a D_PDU for which the C_PDU END field is set equal to one.
- f. The bytes within a C_PDU segment **shall** ⁽⁷⁾ be taken from the source as a contiguous sequence of bytes in the same order in which they occurred in the source C_PDU.
- g. D_PDU's containing C_PDU segments taken in sequence from the source C_PDU **shall** ⁽⁸⁾ have sequential Frame Sequence number fields, modulo 256. [Note: With the first C_PDU segment placed in a D_PDU with Frame Sequence = P, the second would have Frame Sequence = P+1, the third P+2, and so on, with Frame-Sequence operations performed modulo-256.]



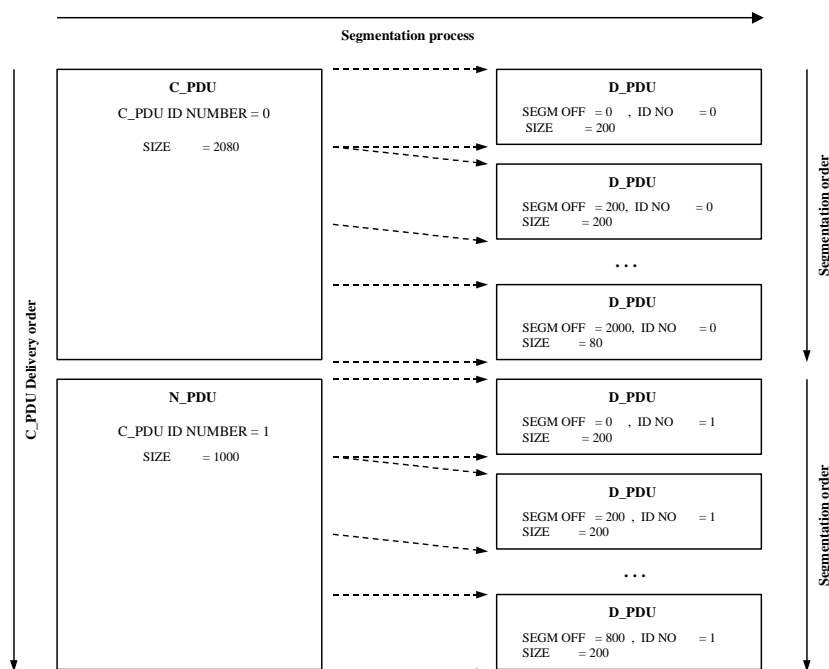
**Figure C-34. C_PDU Re-assembly for ARQ Delivery Services
(Regular and Expedited Service)**

Re-assembly of a C_PDU from its segments **shall** ⁽⁹⁾ be performed in accordance with the example shown in Figure C-34 and as follows (unless noted otherwise, C_PDU segments that are reassembled are expected to have passed the CRC error-check and have no detectable errors):

- (a) The re-assembly process for C_PDU's receiving ARQ-service **shall** ⁽⁹⁾ use the Frame-Sequence-Number field, C_PDU START flag, and C_PDU END flag to determine when all segments of a C_PDU have been received.

- (b) A C_PDU segment taken from a D_PDU whose C_PDU START and C_PDU END flags are both set to the value “one” (1) **shall** ⁽¹⁰⁾ be taken as a single C_PDU and processed as follows:
- 1) If the D_PDU is for a regular (unexpedited) data type, the C_PDU **shall** ⁽¹¹⁾ be delivered to the Channel Access Sublayer using a D_UNIDATA_INDICATION primitive;
 - 2) If the D_PDU is for an unexpedited data type, the C_PDU **shall** ⁽¹²⁾ be delivered to the Channel Access Sublayer using a D_EXPEDITED_UNIDATA_INDICATION primitive;
- (c) A segment from a C_PDU larger than the Maximum C_PDU Segment Size **shall** ⁽¹³⁾ be combined in modulo 256 order with other segments whose D_PDU Frame Sequence Numbers lie in the range defined by the Frame Sequence Numbers of the C_PDU START and C_PDU END segments;
- (d) A completely reassembled C_PDU **shall** ⁽¹⁴⁾ be delivered to the Channel Access Sublayer using the appropriate D_Primitive.

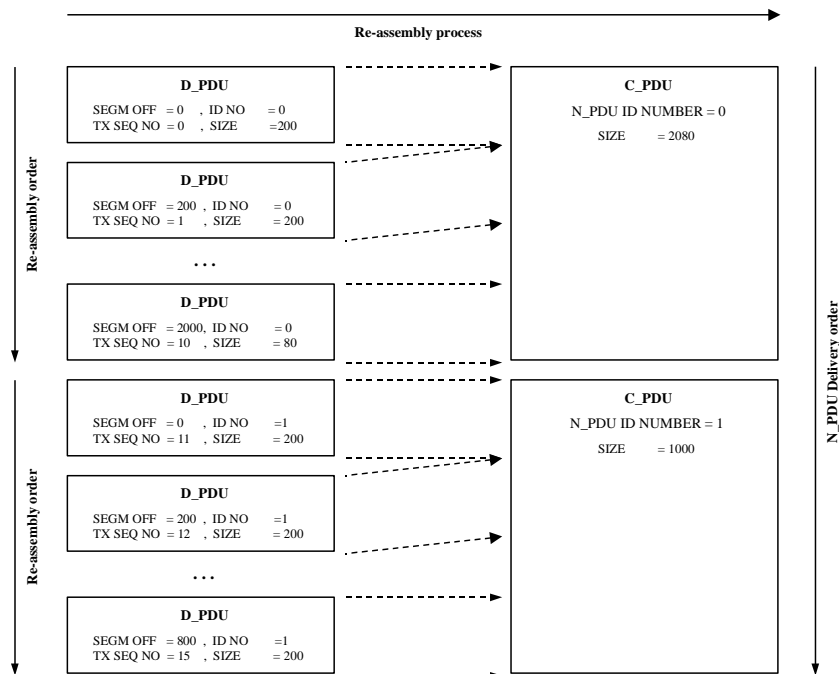
C.4.2 NON-ARQ Mode Segmentation and Re-assembly



**Figure C-35. C_PDU Segmentation for Non-ARQ Delivery Services
(Regular and Expedited Service)**

Segmentation of a C_PDU into segments small enough to fit within a D_PDU for non-ARQ-delivery (i.e, a Non-ARQ DATA, or EXPEDITED-Non-ARQ-DATA D_PDU) **shall** ⁽¹⁾ be performed in accordance with the example shown in Figure C-35, and as follows:

- a. The Maximum C_PDU-Segment Size within a D_PDU for non-ARQ-Delivery services **shall** ⁽²⁾ be a configurable parameter no greater than 1023 bytes in any implementation compliant with this STANAG. An implementation may configure the Maximum C_PDU-Segment Size to match the interleaver size for optimum channel efficiency or other reasons;
- b. An entire C_PDU for non-ARQ delivery that is smaller than the Maximum C_PDU-Segment Size **shall** ⁽³⁾ be placed in the C_PDU segment of a single D_PDU;
- c. A unique C_PDU ID number **shall** ⁽⁴⁾ be assigned to the non-ARQ C_PDU in accordance with the requirements of Section C.3.10;
- d. all D_PDUs containing segments from the same C_PDU **shall** ⁽⁵⁾ have the same C_PDU ID number;
- e. The Segment Offset field of the D_PDU containing the first segment from a C_PDU **shall** ⁽⁶⁾ be equal to zero;
- f. The Segment Offset field of the D_PDU containing any subsequent segment from a C_PDU **shall** ⁽⁷⁾ be set equal to the number of bytes from the original C_PDU that precede the first byte of the segment.



**Figure C-36. C_PDU Re-assembly for Non-ARQ Delivery Services
(Regular and Expedited Service)**

For Non-ARQ services, re-assembly of a C_PDU from its segments **shall** ⁽⁸⁾ be performed in accordance with the example shown in Figure C-36 and as follows (unless noted otherwise, C_PDU segments that are reassembled are expected to have passed the CRC error-check and have no detectable errors):

- a) The re-assembly process for Non-ARQ C_PDUs **shall** ⁽⁹⁾ use the C_PDU ID Number, Segment-Offset field, C_PDU-Segment-Size field, and C_PDU-Size field to determine when all segments of a C_PDU have been received.
- b) If the Error-Free Delivery Mode has been specified, a reassembled C_PDU **shall** ⁽¹⁰⁾ be delivered if and only if all segments of the C_PDU have been received without errors;
- c) If the Deliver-w/-Errors Mode has been specified, the re-assembly process **shall** ⁽¹¹⁾ proceed as follows:
 - 1) C_PDU segments received without detected errors **shall** ⁽¹²⁾ be collected as received in their D_PDUs and placed in order within the reassembled C_PDU;
 - 2) C_PDU segments received with detected errors **shall** ⁽¹³⁾ be placed within the reassembled C_PDU just as they are received in their D_PDUs (i.e, with errors), with the size in bytes and the position of the first byte of the segment noted in the D_Primitive used to deliver the C_PDU to the Channel Access Sublayer;
 - 3) At the end of a specified and configurable timeout-interval the size in bytes and the position of the first byte of any C_PDU segments that have been lost or still not received **shall** ⁽¹⁴⁾ be noted in the D_Primitive that delivers the C_PDU to the Channel Access Sublayer
- d) If the Deliver-In-Order Mode has been specified (with or without the Deliver-with-Errors Mode specified), C_PDUs **shall** ⁽¹⁵⁾ be delivered to the Channel Access Sublayer only if C_PDUs with lower-numbered C_PDU ID Numbers have already been delivered.
- e) If the Deliver-out-of-Order Mode has been specified, C_PDUs **shall** ⁽¹⁶⁾ be delivered to the Channel Access Sublayer as soon as all segments have been received (in Error-Free Mode) or received and accounted for (in Deliver-with-Errors Mode).
- f) Delivery of the reassembled D_PDU **shall** ⁽¹⁷⁾ be performed with the D_Primitive appropriate for the type of data (i.e., regular or expedited) received.

C.5 EOW and Management Message Types

The basic set of EOW Messages may be placed in the 12-bit EOW section of any D_PDU for which an explicit acknowledgement of the EOW Message is not required, and in the MANAGEMENT D_PDU whenever an explicit acknowledgement of the EOW Message is required. The internal structure and use of these messages is specified in this section.

In addition, Extended Management messages may be defined as implementation-specific extensions to the STANAG in accordance with the format and use of the MANAGEMENT D_PDU. The specification and use of Extended Management messages is beyond the scope of this STANAG. Implementation-specific Extended Management messages **shall** ⁽¹⁾ be identified and encoded through the use of the Extended Message Flag as specified in Section C.3.9 for the Management D_PDU.

The types of EOW messages **shall** ⁽²⁾ be as defined in the table below:

Table C-4. EOW Message Types

| Message Type | Function | Contents |
|--------------|--|---|
| 0 | RESERVED | all bits reset to 0 |
| 1 | Data Rate Change Request (DRC_Req) | New HF modem transmit data rate and interleaving setting for DRC master |
| 2 | Data Rate Change Response (DRC_Resp) | Positive or negative response (including reason if negative) |
| 3 | Unrecognized Type Error: user defined message type | message type field which is not recognized |
| 4 | Capability Advertisement | bit map describing capabilities of node |
| 5 | Frequency Change/ ALM Request | As defined in Annex I – Messages and Procedures for Frequency Change |
| 6 | Frequency Change/ ALM Response | As defined in Annex I - Messages and Procedures for Frequency Change |
| 7 | High Data Rate (HDR) Change Request (HDR_DRC_Req) | Management information for High-Data-Rate change: specifies waveform, number of channels, data-rate, and interleaver settings |
| 8-15 | Unspecified/user-defined | |

The format of the EOW message types **shall** ⁽³⁾ be as shown in Figure C-37.

| MSB | | | LSB | | MSB | | | | | LSB | |
|------|----|---|-----|----------|-----|---|---|---|---|-----|---|
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TYPE | | | | contents | | | | | | | |

Figure C-37. Generic Format of EOW Messages

The TYPE field of the EOW message **shall** ⁽⁴⁾ be filled with the hexadecimal value of the appropriate message type (units only), with the LSB of the TYPE value placed in the LSB of the TYPE field.

The Contents field **shall** ⁽⁵⁾ be EOW type-specific, in accordance with the subsections below.

C.5.1 Data Rate Change Request (TYPE 1) EOW Message

| | | | | | | | | | | | | |
|------|----|---|-----|-----------|---|---|-----|--------------|---|------------------|-----|-----|
| MSB | | | LSB | MSB | | | LSB | MSB | | LSB | MSB | LSB |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| 0 | 0 | 0 | 1 | Data Rate | | | | Interleaving | | Other parameters | | |
| TYPE | | | | | | | | | | | | |

Figure C-38. Format of Management Message Type 1.

The Data Rate Change Request (TYPE 1) EOW Message **shall** ⁽¹⁾ be used in conjunction with the Data Rate Change protocol, as specified in Section C.6.4.

The Data Rate Change Request (TYPE 1) EOW Message may be used by a receiving node as an advisory message indicating the modem parameters to which the link may be changed to improve performance. Alternatively, a transmitter may use the Data Rate Change Request (TYPE 1) EOW Message to request a receiver to change to the indicated parameters.

The Data Rate Change Request (TYPE 1) EOW Message **shall** ⁽²⁾ be formatted and encoded as specified in Figure C-38 and the paragraphs that follow, and includes the following type-specific subfields:

- Data Rate
- Interleaving
- Other Parameters

The Data Rate parameter **shall** ⁽³⁾ be the rate at which the node originating the message (i.e, either the DRC Master or Advisee, as noted in Section C.6.4 specifying Data Rate Change Procedures) wishes to transmit data, in accordance with the encoding defined in the following table:

Table C-5. Data Rate Parameter Message Type 1

| MSB - LSB | Interpretation |
|-----------|----------------|
| 0 0 0 0 | 75 bps |
| 0 0 0 1 | 150 bps |
| 0 0 1 0 | 300 bps |
| 0 0 1 1 | 600 bps |
| 0 1 0 0 | 1200 bps |
| 0 1 0 1 | 2400 bps |
| 0 1 1 0 | 3200 bps |
| 0 1 1 1 | 3600 bps |
| 1 0 0 0 | 4800 bps |
| 1 0 0 1 | 6400 bps |
| 1 0 1 0 | 8000 bps |
| 1 0 1 1 | 9600 bps |
| others | reserved |

The Interleaver Parameter field **shall** ⁽⁴⁾ specify the interleaver requested for use by the node producing the message (for that link, if multiple links/modems are in use) with respect to transmit and receive operation, in accordance with the following table:

Table C-6. Interleaver Parameter: Message Type 1

| MSB - LSB | Interpretation |
|-----------|--------------------|
| 0 0 | no interleaving |
| 0 1 | short interleaving |
| 1 0 | long interleaving |
| 1 1 | reserved |

The Other Parameters field **shall** ⁽⁵⁾ specify the capabilities of the modem in use by the node producing the message (for that link, if multiple links/modems are in use) with respect to transmit and receive data rates, and whether the message is an advisory message or request message, in accordance with the following table:

Table C-7. Contents for Type 1 Message (Other Parameters)

| MSB - LSB | Interpretation |
|-----------|---|
| 0 0 | DRC Request: master has independent data rate (change applies to Tx data rate only) |
| 0 1 | DRC Request: Tx and Rx data rate at master must be equal (change will apply to both Tx and Rx data rates) |
| 1 0 | DRC Advisory: Advising node has independent data rate for Tx and Rx (change applies to Rx data rate only) |
| 1 1 | DRC Advisory: Tx and Rx data rate at advising node must be equal (change will apply to both Tx and Rx data rates) |

C.5.2 Data Rate Change Response (TYPE 2) EOW Message

| MSB | | | LSB | MSB | | | LSB | MSB | | | | LSB |
|------|----|---|-----|----------|---|---|-----|--------|---|---|---|-----|
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| TYPE | | | | Response | | | | Reason | | | | |

Figure C-39. Format of Type 2 Message

The Data Rate Change Response (TYPE 2) EOW Message **shall** ⁽¹⁾ be used in conjunction with the Data Rate Change protocol, as specified in Section C.6.4.

The Data Rate Change Response (TYPE 2) EOW Message **shall** ⁽²⁾ be encoded as shown in Figure C-39, and include the following fields:

- Response
- Reason

The Response field **shall** ⁽³⁾ indicate the originator's response to the last DRC-related message it received, with possible responses and their encoding as defined in the following table:

Table C-7. Contents for Type 2 Message (Response)

| MSB - LSB | Interpretation |
|-----------|----------------|
| 0 0 0 | accept |
| 0 0 1 | refuse |
| 0 1 0 | cancel |
| 0 1 1 | confirm |

The Reason field **shall** ⁽⁴⁾ indicate the originator's reason for its response, with possible reasons and their encoding as defined in the following table:

Table C-8. Contents for Type 2 Message (Reason) provided in Response to a Type 1 Data-Rate Change message

| MSB - LSB | Interpretation |
|-----------|--|
| 0 0 0 0 | no reason (used to indicate unconditional acceptance of DRC_Request) |
| 0 0 0 1 | Tx and Rx parameters must be the same (conditionally accept) |
| 0 0 1 0 | Not possible to change modem data rate |
| 0 0 1 1 | Not possible to change modem interleaving |
| 0 0 1 0 0 | Not possible to change modem data rate or interleaving |
| 0 0 1 0 1 | Not consistent with local conditions |

[N.B.: Reason-Field encoding differs when provided in Response to a Type 7 High-Data-Rate Change message, as specified in para. C.5.5]

C.5.3 Unrecognised Type Error (TYPE 3) EOW Message

| MSB | | | LSB | MSB | | | LSB | MSB | | | | LSB |
|------|----|---|-----|----------|---|---|-----|--------|---|---|---|-----|
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| TYPE | | | | Response | | | | Reason | | | | |

Figure C-40. Format of Type 3 Message

The UNRECOGNIZED-TYPE ERROR (Type 3) EOW Message **shall** ⁽¹⁾ be used to declare an error related to receipt of an EOW message.

[N.B.: A node **should** send an UNRECOGNIZED-TYPE ERROR (Type 3) EOW Message to the originator of an unrecognised EOW message whenever the node receives one.]

The UNRECOGNIZED-TYPE ERROR (Type 3) EOW Message **shall** ⁽²⁾ be encoded as shown in Figure C.40 and include the following fields:

- Response
- Reason

The type of the unrecognised EOW message or the message that triggered the error **shall** ⁽³⁾ be mapped into the four least-significant bits of the “reason” field.

The unused MSB of the “reason” field, and all bits in the “response” field, **shall** ⁽⁴⁾ be reset to the value zero (0).

C.5.4 Capability Advertisement (TYPE 4) EOW Message

| | | | | | | | | | | | | | | | |
|------|----|---|---|-----|---|---|---|----------|---|---|---|-----|--|--|--|
| MSB | | | | LSB | | | | MSB | | | | LSB | | | |
| 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | |
| TYPE | | | | | | | | contents | | | | | | | |

Figure C-41. Format of Capabilities (Type 4) MANAGEMENT Messages

The EOW Message Type 4 allows a node to inform another node of its capabilities related to modem parameters, waveform, and control.

The EOW Message Type 4 **shall** ⁽¹⁾ be encoded as shown in Figure C-41 and contains a single field, Content.

The Content field **shall** ⁽²⁾ be encoded as the bit-mapped specification of capabilities defined in the following table:

Table C-9. Contents of Message field for MANAGEMENT Message Type 4

| bit | meaning |
|---------|---|
| 7 (MSB) | Adaptive modem parameters (DRC) capable ^{note 1} (0 = no, 1 = yes) |
| 6 | STANAG 4529 available ^{note 2} (0 = no, 1 = yes) |
| 5 | MIL-STD-188-110 75-2400 bps available ^{note 2} (0 = no, 1 = yes) |
| 4 | extended data rate capable ^{note 3} (0 = no, 1 = yes) |
| 3 | full duplex supported ^{note 4} (0 = no, 1 = yes) |
| 2 | split frequency supported ^{note 4} (0 = no, 1 = yes) |
| 1 | non-ARCS ALE capable ^{note 5} (0 = no, 1 = yes) |
| 0 (LSB) | ARCS capable ^{note 6} (0 = no, 1 = yes) |

Notes

1. If a node is DRC capable, it must implement at minimum 75 bps through 2400 bps (1200 bps for STANAG 4529) with short and long interleaving in accordance with the relevant document.
2. All nodes shall have, at minimum, the STANAG 4285 waveform available.
3. Annex G describes waveforms for data rates above 2400 bps.
4. A full duplex node must have split frequency operation available.
5. non-ARCS ALE capability may imply MIL-STD-188-141 Appendix A or proprietary capabilities and any ambiguity must be resolved outside of STANAG 5066.
6. ARCS –capable systems are those equipped with NATO's Automatic Radio Control System for HF Links, STANAG 4538 (FLSU/RLSU) or MIL-STD-188-141B (RLSU); any ambiguity must be resolved outside of STANAG 5066.

C.5.5 High Data Rate Change Request (TYPE 7) EOW Message

The Type 7 Extended EOW Message **may** ⁽¹⁾ be used to specify a High Data Rate (HDR) Change Request.

The message allows specification of a wider range of waveform types than is supported by the Type 1 Data

Rate Change Request EOW Message alone. The High Data Rate (HDR) Change Request Message is an Extended EOW format and **must** be embedded within a Type 6 Management DPDU messages as shown in the Figure below.

| Byte/ Bit Num. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|----------------|--|------------|---|--|--------------------|-------------------|---------------------|-----|---|
| | The two-byte message preamble is not shown; | | | | | | | | Other remarks: DPDU_TYPE = 6 EOW_TYPE = 7 ⁽¹⁾ EOW_DATA, waveform encoded as per Table C-9-2 |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | |
| 1 | MODEM WAVEFORM | | | | NUMBER OF CHANNELS | | | | |
| 2 | END_OF_TRANSMISSION (EOT) | | | | | | | | m, k in bytes |
| 3 | SIZE_OF_ADDRESS (m ∈ {1 ... 7}) | | | SIZE_OF_HEADER ⁽³⁾ (k = 14) | | | | | |
| 3 + m | SOURCE_AND_DESTINATION_ADDRESS | | | | | | | | Field-length = m bytes |
| 4+m | | NOT_USED_1 | | | | EXT MSG = 1 | VALID MSG = 1 | ACK | This is the extended form of the EOW message. This is a required field for the Type 6 DPDU |
| 5+m | MSB - -- MANAGEMENT FRAME ID NUMBER -- - LSB | | | | | | | | |
| 6+m | MSB - - - - - | | | | | | | | Requested Data Rate for each channel |
| 7+m | Channel | | | | | | | | |
| 8+m | Data-Rate (in bps) | | | | | | | | |
| 9+m | LSB | | | | | | | | interleaver length for each channel |
| 10+m | MSB | | | Interleaver | | | | - | |
| 11+m | | | | Length | | | | LSB | |
| CRC_H_1 | CRC_ON_HEADER | | | | | | | MSB | |
| CRC_H_2 | LSB | | | | | | | | |

Figure C-41.1 - Type 7 Extended EOW High-Data-Rate Change Request Message Format (all fields of the encapsulating Type 6 DPDU message are also shown)

The format of the embedded Type 7 EOW message field **shall** ⁽²⁾ be as follows.

Table C-9-1. Contents of the Type 7 EOW HDR Change Request Message

| MSB | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | LSB |
|-----|------|----|---|---|----------------|---|---|---|--------------------|---|---|---|-----|
| | 0 | 1 | 1 | 1 | | | | | | | | | |
| | TYPE | | | | MODEM WAVEFORM | | | | NUMBER OF CHANNELS | | | | |

The MODEM WAVEFORM field **shall** ⁽³⁾ contain an enumeration of the modem waveform requested as defined in the following table.

Table C-9-2. Contents of the MODEM WAVEFORM field

| WAVEFORM | ENUMERATION |
|-----------------------------|-------------|
| MS110A | 0 |
| MS110B | 1 |
| STANAG_4285 | 2 |
| STANAG_4539 | 3 |
| STANAG_4529 | 4 |
| STANAG_4415 | 5 |
| STANAG_4481_FSK | 6 |
| USER_CONFIGURATION_OPTION_1 | 7 |
| USER_CONFIGURATION_OPTION_2 | 8 |
| USER_CONFIGURATION_OPTION_3 | 9 |
| Unspecified | 10 .. 31 |

The NUMBER OF CHANNELS field **shall** ⁽⁴⁾ specify the number of channels being transmitted. The range **shall** ⁽⁵⁾ be from 1 8, with binary 0 specifying 8 channels.

The response to the Type 7 Extended EOW message **shall** ⁽⁶⁾ be an EOW Type 2 DRC Response, or an EOW Type 3 Unrecognised Type Error (or no response, as noted in the paragraph below).

Both EOW types will be encoded as specified in paragraph C.5. If the DRC Response indicates “Refuse”, the Reason field **shall** ⁽⁷⁾ be encoded as specified in the following table.

Table C-9-3. Contents for Type 2 Message (Reason Field) provided in Response to a Type 7 High Data-Rate Change message

| MSB-LSB | Interpretation |
|---------|--|
| 00000 | As defined in paragraph C.5, Table C-8 |
| 00001 | Not used |
| 00010 | Not possible to change to specified waveform |
| 00011 | Not possible to change to specified number of channels |
| 00100 | Not possible to change to specified interleaver length |
| 00101 | As defined in paragraph C.5, Table C-8 |

By definition, the HDR Change Request is a request and not an advisory.

The procedure for using the Type 7 Extended EOW HDR Change Request message **shall** ⁽⁷⁾ be as

Described in paragraph C.6.4. Specifically; the sender of the Type 7 message **shall** ⁽⁸⁾ be the HDRC master and the receiver will be the HDRC slave. The slave will respond with accept or refuse (EOW Type 2), unrecognised (EOW Type 3), or no response:

- If the slave accepts the request (using the EOW Type 2), then the master will send an acknowledgement and both will change to the new mode. After the change, the slave will send a confirm and the master will acknowledge.
- If the slave explicitly refuses the request using the EOW Type 2, the master may cease the request or send a new request with different parameters, depending on the reason given.
- A slave that explicitly refuses the request by responding with an Unrecognised EOW (Type 3) message signals that it does not participate (or understand) the HDR Change protocol and the master **shall** ⁽⁹⁾ cease immediately.
- If no response is received from the slave, the master may cease the request or may send the request again.

The initial data rate and interleaver length **shall** ⁽¹⁰⁾ be provided by using the Extended Message field of the Type 6 D_PDU, as shown in Figure C-41.1. This information **shall** ⁽¹¹⁾ be included in the Type 7 EOW HDR Change Request and **may** be included in the Type 2 EOW Data Rate Change Response message. If contained in the Type 2 EOW Data Rate Change Response message, the initial data rate and interleaver length are recommended values. The data rate and interleaver length shall be encoded in the Extended Message field as follows.

Table C-9-4. Contents of the Extended EOW Message field in a Type 7 EOW High Data-Rate Change message

| Byte OFFSET ⁽¹⁾ | Description |
|----------------------------|---|
| 0 .. 3 | Data rate (MSB at offset 0) |
| 4 .. 5 | Interleaver length in hundredths of seconds (MSB at offset 4) |

(1) Note offset is with respect to the first byte of the extended message field.

C.6 Peer-to-peer Communication Protocols

This section discusses the interactions between the Data Transfer Sublayer entities at different nodes in terms of states, state-transition diagrams, and state tables for a hypothetical state machine. This STANAG does not mandate a state machine implementation. The requirement for interoperability is that the system act consistently with the state-transition and actions rules for message exchange and format presented in this STANAG.

C.6.1 Data Transfer Sublayer States and Transitions

The expected and allowed interactions of one node with another are described herein with respect to states of the node's Data Transfer sublayer. Receiving certain PDUs (from the modem) or D_Primitives (from the Channel Access Sublayer) will cause a node, depending on its state, to transmit certain PDUs and/or D_Primitives, and/or change to another state.

The Data Transfer Sublayer interactions with peers **shall** ⁽¹⁾ be defined with respect to the states shown in Figure C-42 and as follows:

| | |
|-----------------------------|---------------------------|
| IDLE(UNCONNECTED) | IDLE(CONNECTED) |
| DATA(UNCONNECTED) | DATA(CONNECTED) |
| EXPEDITED-DATA(UNCONNECTED) | EXPEDITED-DATA(CONNECTED) |
| MANAGEMENT(UNCONNECTED) | MANAGEMENT(CONNECTED) |

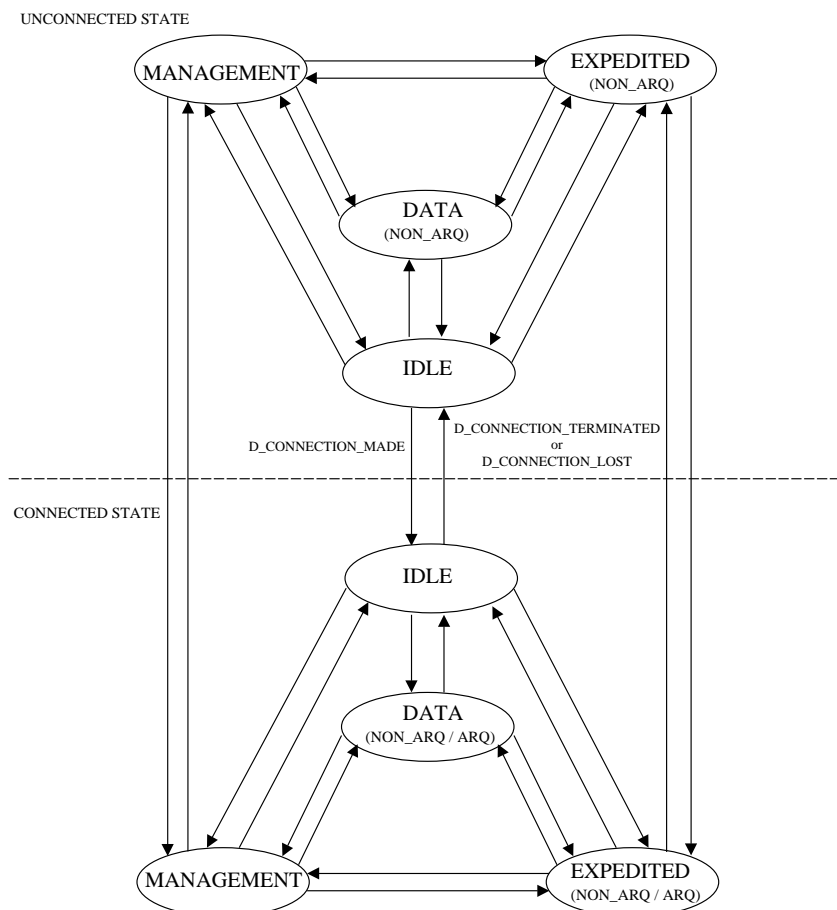


Figure C-42. Nominal State and Transition Diagram (Data Transfer Sublayer)

C.6.1.1 State Transition Rules

The transitions between DTS States and the actions that arise from given events **shall** ⁽¹⁾ be as defined in the tables presented in the subsections that follow.

DTS states, transitions, and actions **shall** ⁽²⁾ be defined and maintained with respect to a specified remote node address.

In all of the tables below, “PDU received” refers to a PDU received that is addressed to the node in question from the specified remote node for which the states are maintained. Action and transitions rules **shall** ⁽³⁾ not occur based on PDUs addressed to other nodes or from a node other than the specified remote node for which the states are maintained.

Likewise, “D_Primitive received” refers to a D_Primitive received from the Channel Access sublayer that references the specified remote node. Action and transitions rules **shall** ⁽⁴⁾ not occur based on D_Primitives that reference nodes other than the specified remote node.

DTS states **shall** ⁽⁵⁾ be maintained for each specified node for which a connection or ARQ protocol must be maintained. [Note: If multiple links, as defined by the Channel Access or Subnetwork Interface sublayers, must be maintained simultaneously, then a set of DTS State Variables must be maintained for each of the links. If links are not maintained simultaneously, DTS State variables may be reused for any node]

C.6.1.1.1 IDLE(UNCONNECTED) State Transition Rules

When in the IDLE(UNCONNECTED) State, the Data Transfer Sublayer **shall** ⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-10. Event/Transition/Action Rules for
IDLE (UNCONNECTED) State: CAS-Related Events**

| Received D_Primitive/ Event | State Transition to: | Action & Comments |
|---------------------------------------|---|--|
| D_CONNECTION_MADE | IDLE(CONNECTED) | INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.] |
| D_CONNECTION_TERMINATED | IDLE(UNCONNECTED) , i.e., No Change. | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.] |
| D_UNIDATA_REQUEST (ARQ) | IDLE(UNCONNECTED) , i.e., No Change. | REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_EXPEDITED_UNIDATA_REQUEST (ARQ) | IDLE(UNCONNECTED) , i.e., No Change. | REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_UNIDATA_REQUEST (NON-ARQ) | DATA(UNCONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs |
| D_EXPEDITED_UNIDATA_REQUEST (NON-ARQ) | EXPEDITED DATA (UNCONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs |

When in the IDLE(UNCONNECTED) State, the Data Transfer Sublayer **shall** ⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-11. Event/Transition/Action Rules for
IDLE (UNCONNECTED) State: Reception-Related Events**

| Received D_PDU Event | State Transition to: | Action & Comments |
|--------------------------------------|---|--|
| DATA Type 0 D_PDU | IDLE(UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| ACK Type 1 D_PDU | IDLE(UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| DATA-ACK Type 2 D_PDU | IDLE(UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| RESET/WIN RESYNC Type 3 D_PDU | IDLE(UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-DATA Type 4 D_PDU | IDLE(UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-ACK Type 5 D_PDU | IDLE(UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (UNCONNECTED) | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | DATA(UNCONNECTED) | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON_ARQ Type 8 D_PDU | EXPEDITED-DATA (UNCONNECTED) | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU |
| WARNING Type 15 D_PDU | IDLE(UNCONNECTED) , i.e., No Change. | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.1.2 DATA(UNCONNECTED) State Transition Rules

When in the DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall** ⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-12. Event/Transition/Action Rules for
DATA (UNCONNECTED) State: CAS-Related Events**

| Received D_Primitive Event | State Transition to: | Action & Comments |
|--|--|--|
| D_CONNECTION_MADE | IDLE(CONNECTED) | INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.] |
| D_CONNECTION_TERMINATED | DATA (UNCONNECTED) , i.e., No Change. | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.] |
| D_UNIDATA_REQUEST (ARQ) | DATA (UNCONNECTED) , i.e., No Change. | REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_EXPEDITED_UNIDATA _REQUEST (ARQ) | DATA(UNCONNECTED) , i.e., No Change. | REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_UNIDATA_REQUEST (NON-ARQ) | DATA(UNCONNECTED) , i.e., No Change. | SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs |
| D_EXPEDITED_UNIDATA _REQUEST (NON_ARQ) | EXPEDITED-DATA (UNCONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs |

When in the DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall** ⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-13. Event/Transition/Action Rules for
DATA (UNCONNECTED) State: Reception-Related Events**

| Received D_PDU Event | State Transition to: | Action & Comments |
|--------------------------------------|---|---|
| DATA Type 0 D_PDU | DATA(UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| ACK Type 1 D_PDU | DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| DATA-ACK Type 2 D_PDU | DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| RESET/WIN RESYNC Type 3 D_PDU | DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-DATA Type 4 D_PDU | DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-ACK Type 5 D_PDU | DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (UNCONNECTED) | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | DATA(UNCONNECTED), i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON_ARQ Type 8 D_PDU | DATA(UNCONNECTED), i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_ INDICATION if the DTS received a complete C_PDU |
| WARNING Type 15 D_PDU | DATA(UNCONNECTED), i.e., No Change. | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.1.3 EXPEDITED-DATA(UNCONNECTED) State Transition Rules

When in the EXPEDITED-DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall**⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-14. Event/Transition/Action Rules for
EXPEDITED-DATA (UNCONNECTED) State: CAS-Related Event**

| Received D_Primitive Event | State Transition to: | Action & Comments |
|---------------------------------------|--|--|
| D_CONNECTION_MADE | IDLE(CONNECTED) | INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.] |
| D_CONNECTION_TERMINATED | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.] |
| D_UNIDATA_REQUEST (ARQ) | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_EXPEDITED_UNIDATA_REQUEST (ARQ) | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_UNIDATA_REQUEST (NON-ARQ) | IF completed transmitting all Expedited-Data, change to DATA(UNCONNECTED), OTHERWISE, EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | IF completed sending all Expedited-Data, SEGMENT C_PDU, then, COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE, QUEUE C_PDU Pending completion of Expedited-Data transmission. |
| D_EXPEDITED_UNIDATA_REQUEST (NON-ARQ) | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs |

When in the EXPEDITED-DATA(UNCONNECTED) State, the Data Transfer Sublayer **shall**⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-15. Event/Transition/Action Rules for
EXPEDITED-DATA (UNCONNECTED) State: Reception-Related Event**

| Received D_PDU/ Event | State Transition to: | Action & Comments |
|--------------------------------------|--|--|
| DATA Type 0 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| ACK Type 1 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| DATA-ACK Type 2 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| RESET/WIN RESYNC Type 3 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-DATA Type 4 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-ACK Type 5 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (UNCONNECTED) | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON_ARQ Type 8 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU |
| WARNING Type 15 D_PDU | EXPEDITED-DATA (UNCONNECTED), i.e., No Change. | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.1.4 MANAGEMENT(UNCONNECTED) State Transition Rules

When in the MANAGEMENT(UNCONNECTED) State, the Data Transfer Sublayer **shall** ⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-16. Event/Transition/Action Rules for
MANAGEMENT (UNCONNECTED) State: CAS-Related Events**

| Received D_Primitive/ Event | State Transition to: | Action & Comments |
|---|---|--|
| D_CONNECTION_MADE | MANAGEMENT (CONNECTED) | INITIALIZE State Variables for ARQ processing, in accordance with Section C.6.2 [Note: CAS has accepted a connection request from the remote node.] |
| D_CONNECTION_TERMINATED | MANAGEMENT (UNCONNECTED) , i.e., No Change. | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to terminate a connection that did not exist.] |
| D_UNIDATA_REQUEST (ARQ) | MANAGEMENT (UNCONNECTED) , i.e., No Change. | REPLY w/ D_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_EXPEDITED_UNIDATA _ REQUEST (ARQ) | MANAGEMENT (UNCONNECTED) , i.e., No Change. | REPLY w/ D_EXPEDITED_UNIDATA_INDICATION (REJECT). [Note: reliable data cannot be sent over a nonexistent connection.] |
| D_UNIDATA_REQUEST (NON-ARQ) | IF Management Protocol completed, DATA(UNCONNECTED), OTHERWISE, MANAGEMENT (UNCONNECTED) , i.e., No Change. | IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol. |
| D_EXPEDITED_UNIDATA _ REQUEST (NON_ARQ) | IF Management Protocol completed, EXPEDITED DATA (UNCONNECTED), OTHERWISE, MANAGEMENT (UNCONNECTED) , i.e., No Change. | IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol. |

When in the MANAGEMENT(UNCONNECTED) State, the Data Transfer Sublayer **shall** ⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-17. Event/Transition/Action Rules for
MANAGEMENT (UNCONNECTED) State: Reception-Related Events**

| Received D_PDU Event | State Transition to: | Action & Comments |
|--------------------------------------|--|--|
| DATA Type 0 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| ACK Type 1 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| DATA-ACK Type 2 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| RESET/WIN RESYNC Type 3 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-DATA Type 4 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| EXPEDITED-ACK Type 5 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =1="Connection-Related D_PDU but Unconnected"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON_ARQ Type 8 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU |
| WARNING Type 15 D_PDU | MANAGEMENT (UNCONNECTED) , i.e., No Change. | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.1.5 IDLE(CONNECTED) State Transition Rules

When in the IDLE(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-18. Event/Transition/Action Rules for
IDLE (CONNECTED) State: CAS-Related Event**

| Received D_Primitive Event | State Transition to: | Action & Comments |
|---------------------------------------|---------------------------------|--|
| D_CONNECTION_MADE | IDLE(CONNECTED), i.e. No Change | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.] |
| D_CONNECTION_TERMINATED | IDLE(UNCONNECTED) | TERMINATE ARQ processing; RESET State Variables. |
| D_UNIDATA_REQUEST (ARQ) | DATA(CONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant ARQ DATA Type 0 D_PDUs; |
| D_EXPEDITED_UNIDATA_REQUEST (ARQ) | EXPEDITED-DATA(CONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant ARQ DATA Type 4 D_PDUs; |
| D_UNIDATA_REQUEST (NON-ARQ) | DATA(CONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs |
| D_EXPEDITED_UNIDATA_REQUEST (NON-ARQ) | EXPEDITED DATA (CONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs |

When in the IDLE(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-19. Event/Transition/Action Rules for
IDLE (CONNECTED) State: Reception-Related Event**

| Received D_PDU/ Event | State Transition to: | Action & Comments |
|--------------------------------------|------------------------------------|---|
| DATA Type 0 D_PDU | DATA(CONNECTED) | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU. |
| ACK Type 1 D_PDU | DATA(CONNECTED) | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| DATA-ACK Type 2 D_PDU | DATA(CONNECTED) | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU as appropriate; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_CONFIRM or D_UNIDATA_REQUEST_REJECTED. |
| RESET/WIN RESYNC Type 3 D_PDU | IDLE(CONNECTED), i.e. No Change | COMPOSE and SEND RESET/WIN RESYNC Type 3 D_PDU as required; EXECUTE FULL RESET/WIN RESYNC Protocol if required. |
| EXPEDITED-DATA Type 4 D_PDU | EXPEDITED-DATA (CONNECTED) | PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND EXPEDITED ACK Type 5 D_PDU |
| EXPEDITED-ACK Type 5 D_PDU | EXPEDITED-DATA (CONNECTED) | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (CONNECTED) | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | DATA(CONNECTED) | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON_ARQ Type 8 D_PDU | EXPEDITED-DATA (CONNECTED) | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if a complete C_PDU is received |
| WARNING Type 15 D_PDU | IDLE(CONNECTED), i.e. No Change | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.1.6 DATA(CONNECTED) State Transition Rules

When in the DATA(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-20. Event/Transition/Action Rules for
DATA (CONNECTED) State: CAS-Related Events**

| Received D_Primitive Event | State Transition to: | Action & Comments |
|--|-------------------------------|--|
| D_CONNECTION_MADE | DATA(CONNECTED) | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.] |
| D_CONNECTION_TERMINATED | IDLE(UNCONNECTED) | TERMINATE ARQ processing; RESET State Variables. |
| D_UNIDATA_REQUEST (ARQ) | DATA(CONNECTED) | SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant DATA Type 0 D_PDUs; |
| D_EXPEDITED_UNIDATA _REQUEST (ARQ) | EXPEDITED- DATA(CONNECTED) | SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant EXPEDITED-DATA Type 4 D_PDUs; |
| D_UNIDATA_REQUEST (NON-ARQ) | DATA(CONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs |
| D_EXPEDITED_UNIDATA _REQUEST (NON_ARQ) | EXPEDITED DATA (CONNECTED) | SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs |

When in the DATA(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-21. Event/Transition/Action Rules for
DATA (CONNECTED) State: Reception-Related Events**

| Received D_PDU Event | State Transition to: | Action & Comments |
|--------------------------------|---|---|
| DATA Type 0 D_PDU | DATA(CONNECTED) , i.e., No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if a complete C_PDU received; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU. |
| ACK Type 1 D_PDU | DATA (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; SEND additional DATA Type 0 D_PDUs, if any and as able; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_CONFIRM or D_UNIDATA_REQUEST_REJECTED. |
| DATA-ACK Type 2 D_PDU | DATA (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU or DATA-ACK Type 2 D_PDU as appropriate; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_CONFIRM or D_UNIDATA_REQUEST_REJECTED. |
| RESET/WIN RESYNC Type 3 D_PDU | If FULL_RESET_CMD, IDLE(CONNECTED), otherwise, DATA(CONNECTED) , i.e. No Change | COMPOSE and SEND RESET/WIN RESYNC Type 3 D_PDU as required; EXECUTE FULL RESET/WIN RESYNC Protocol if required. |
| EXPEDITED-DATA Type 4 D_PDU | EXPEDITED-DATA (CONNECTED) | PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND EXPEDITED ACK Type 5 D_PDU |
| EXPEDITED-ACK Type 5 D_PDU | EXPEDITED-DATA (CONNECTED) | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (CONNECTED) | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | DATA(CONNECTED), i.e. No Change | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON_ARQ Type 8 D_PDU | EXPEDITED-DATA (CONNECTED) | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if the DTS received a complete C_PDU |
| WARNING Type 15 D_PDU | DATA(CONNECTED) , i.e., No Change. | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.1.7 EXPEDITED-DATA (CONNECTED) State Transition Rules

When in the EXPEDITED-DATA(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-22. Event/Transition/Action Rules for
EXPEDITED-DATA (CONNECTED) State: CAS-Related Events**

| Received D_Primitive Event | State Transition to: | Action & Comments |
|---|--|---|
| D_CONNECTION_MADE | EXPEDITED-DATA (CONNECTED), i.e. No Change | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.] |
| D_CONNECTION_ TERMINATED | IDLE(UNCONNECTED) | TERMINATE ARQ processing; RESET State Variables. |
| D_UNIDATA_REQUEST (ARQ) | in Accordance with Section C.6.1.3: IF completed transmitting all Expedited-Data, change to DATA(CONNECTED), OTHERWISE, EXPEDITED-DATA (CONNECTED), i.e., No Change. | in Accordance with Section C.6.1.3: IF completed sending all Expedited-Data, SEGMENT C_PDU, UPDATE State Variables, then, COMPOSE and SEND all resultant DATA Type 0 D_PDUs; OTHERWISE, QUEUE C_PDU Pending completion of Expedited-Data transmission. |
| D_EXPEDITED_UNIDATA _ REQUEST (ARQ) | in Accordance with Section C.6.1.3: EXPEDITED-DATA (CONNECTED), i.e. No Change | in Accordance with Section C.6.1.3: SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant EXPEDITED-DATA Type 4 D_PDUs; |
| D_UNIDATA_REQUEST (NON-ARQ) | in Accordance with Section C.6.1.3: IF completed transmitting all Expedited-Data, change to DATA(CONNECTED), OTHERWISE, EXPEDITED-DATA (CONNECTED), i.e., No Change. | in Accordance with Section C.6.1.3: IF completed sending all Expedited-Data, SEGMENT C_PDU, then, COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE, QUEUE C_PDU Pending completion of Expedited-Data transmission. |
| D_EXPEDITED_UNIDATA _ REQUEST (NON_ARQ) | EXPEDITED-DATA (CONNECTED), i.e. No Change. | SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs |

When in the EXPEDITED-DATA(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-23. Event/Transition/Action Rules for
EXPEDITED-DATA (CONNECTED) State: Reception-Related Events**

| Received D_PDU Event | State Transition to: | Action & Comments |
|--------------------------------|--|--|
| DATA Type 0 D_PDU | EXPEDITED-DATA (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU |
| ACK Type 1 D_PDU | EXPEDITED-DATA (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_CONFIRM or D_UNIDATA_REQUEST_REJECTED. |
| DATA-ACK Type 2 D_PDU | EXPEDITED-DATA (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND ACK Type 1 D_PDU; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_CONFIRM or D_UNIDATA_REQUEST_REJECTED. |
| RESET/WIN RESYNC Type 3 D_PDU | If FULL_RESET_CMD, IDLE(CONNECTED), otherwise, DATA(CONNECTED) | COMPOSE and SEND RESET/WIN RESYNC Type 3 D_PDU as required; EXECUTE FULL RESET/WIN RESYNC Protocol if required. |
| EXPEDITED-DATA Type 4 D_PDU | EXPEDITED-DATA (CONNECTED), i.e. No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if received complete C_PDU; COMPOSE and SEND EXPEDITED ACK Type 5 D_PDU |
| EXPEDITED-ACK Type 5 D_PDU | EXPEDITED-DATA (CONNECTED), i.e. No Change. | PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; SEND additional EXPEDITED-DATA D_PDUs, if any and as able; as appropriate, NOTIFY CAS using D_EXPEDITED_UNIDATA_REQUEST_CONFIRM or D_EXPEDITED_UNIDATA_REQUEST_REJECTED. |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (CONNECTED) | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | EXPEDITED-DATA (CONNECTED), i.e. No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON_ARQ Type 8 D_PDU | EXPEDITED-DATA (CONNECTED), i.e. No Change | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if the DTS received a complete C_PDU |
| WARNING Type 15 D_PDU | EXPEDITED-DATA (CONNECTED) , i.e., No Change. | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.1.8 MANAGEMENT (CONNECTED) State Transition Rules

When in the MANAGEMENT(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽¹⁾ respond to reception of a D_Primitive from the Channel Access Sublayer as specified in the following Table:

**Table C-24. Event/Transition/Action Rules for
MANAGEMENT (CONNECTED) State: CAS-Related Events**

| Received D_Primitive/ Event | State Transition to: | Action & Comments |
|---------------------------------------|--|---|
| D_CONNECTION_MADE | MANAGEMENT (CONNECTED) , i.e., No Change. | IGNORE [Note: this is an anomalous event, the DTS may notify the CAS that an attempt was made to make a connection that already exists.] |
| D_CONNECTION_TERMINATED | IDLE (UNCONNECTED) | TERMINATE ARQ processing; RESET State Variables. |
| D_UNIDATA_REQUEST (ARQ) | IF Management Protocol completed, DATA(CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED) , i.e., No Change. | IF Management Protocol completed, SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant DATA Type 0 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol. |
| D_EXPEDITED_UNIDATA_REQUEST (ARQ) | IF Management Protocol completed, EXPEDITED DATA (CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED), i.e., No Change. | IF Management Protocol completed, SEGMENT C_PDU; UPDATE State ARQ Variables; COMPOSE and SEND all resultant EXPEDITED-DATA Type 4 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol. |
| D_UNIDATA_REQUEST (NON-ARQ) | IF Management Protocol completed, DATA(CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED) , i.e., No Change. | IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant NON-ARQ DATA Type 7 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol. |
| D_EXPEDITED_UNIDATA_REQUEST (NON_ARQ) | IF Management Protocol completed, EXPEDITED DATA (CONNECTED), OTHERWISE, MANAGEMENT (CONNECTED) , i.e., No Change. | IF Management Protocol completed, SEGMENT C_PDU; COMPOSE and SEND all resultant EXPEDITED NON-ARQ DATA Type 8 D_PDUs; OTHERWISE QUEUE C_PDU Pending completion of management protocol. |

When in the MANAGEMENT(CONNECTED) State, the Data Transfer Sublayer **shall** ⁽²⁾ respond to reception of a D_PDU from the lower layers as specified in the following Table:

**Table C-25. Event/Transition/Action Rules for
MANAGEMENT (CONNECTED) State: Reception-Related Events**

| Received D_PDU Event | State Transition to: | Action & Comments |
|--------------------------------|---|--|
| DATA Type 0 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE ACK Type 1 D_PDU for deferred transmission on transition to an appropriate state. |
| ACK Type 1 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; UPDATE ARQ State Variables; ADVANCE Flow-Control WINDOW; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_CONFIRM or D_UNIDATA_REQUEST_REJECTED. |
| DATA-ACK Type 2 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; UPDATE ARQ State Variables; DELIVER received C_PDU to CAS using D_UNIDATA_INDICATION if received complete C_PDU; COMPOSE ACK Type 1 D_PDU for deferred transmission on transition to an appropriate state; as appropriate, NOTIFY CAS using D_UNIDATA_REQUEST_CONFIRM or D_UNIDATA_REQUEST_REJECTED. |
| RESET/WIN RESYNC Type 3 D_PDU | MANAGEMENT (CONNECTED) | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED; CONTINUE w/ MANAGEMENT Protocol |
| EXPEDITED-DATA Type 4 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | PROCESS the D_PDU; REASSEMBLE C_PDU segments; DELIVER to CAS using D_EXPEDITED_UNIDATA_INDICATION if received complete C_PDU; COMPOSE EXPEDITED ACK Type 5 D_PDU for deferred transmission on transition to an appropriate state. |
| EXPEDITED-ACK Type 5 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | COMPOSE and SEND WARNING Type 15 D_PDU, w/ Reason =3="Invalid D_PDU for Current State"; NOTIFY CAS using D_WARNING_TRANSMITTED |
| MANAGEMENT Type 6 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | PROCESS Type 6 D_PDU; RESPOND if required. |
| NON-ARQ DATA Type 7 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if received complete C_PDU |
| EXPEDITED NON-ARQ Type 8 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | PROCESS the D_PDU and reassemble C_PDU segments; DELIVER to CAS using D_UNIDATA_INDICATION if the DTS received a complete C_PDU |
| WARNING Type 15 D_PDU | MANAGEMENT (CONNECTED) , i.e., No Change. | CEASE action indicated in warning; NOTIFY CAS using D_WARNING_RECEIVED |

C.6.1.2 State Rules for Sending and Receiving D_PDUs

A node **shall** ⁽¹⁾ transmit only those D_PDUs which are allowed for its current state as follows:

| | |
|-----------------------------|-------------------------------------|
| IDLE(UNCONNECTED) | type 15 D_PDUs |
| DATA(UNCONNECTED) | type 7, or 15 D_PDUs |
| EXPEDITED DATA(UNCONNECTED) | type 8, or 15 D_PDUs |
| MANAGEMENT(UNCONNECTED) | type 6, or 15 D_PDUs |
| IDLE(CONNECTED) | type 15 D_PDUs |
| DATA(CONNECTED) | type 0, 1, 2, 3, 7, 8, or 15 D_PDUs |
| EXPEDITED DATA(CONNECTED) | type 1, 4, 5, 8, or 15 D_PDUs |
| MANAGEMENT(CONNECTED) | type 6, 8, or 15 D_PDUs |

A node **shall** ⁽²⁾ receive and process all valid D_PDUs regardless of its current state. Transmission of responses to a received D_PDU may be immediate or deferred, as appropriate for the current state and as specified in Section 6.1.1.

C.6.1.3 D_PDU Processing Rules: EXPEDITED-DATA (CONNECTED) State

A separate Frame Sequence Number counter **shall** ⁽¹⁾ be maintained for the transmission of ARQ Expedited Data, distinct from the counter with the same name used for regular-delivery service with D_PDU types 0 and 2.

A separate C_PDU ID counter **shall** ⁽²⁾ be maintained for the transmission of ARQ Expedited Data, distinct from the counter with the same name used for regular non-ARQ and expedited non-ARQ delivery services with D_PDU types 7 and 8.

Upon entering the EXPEDITED-DATA (CONNECTED) state, the EXPEDITED-DATA D_PDU Frame Sequence Number counter **shall** ⁽³⁾ be set to the value zero (0).

Starting or restarting another ARQ machine (i.e. establishing a link with a new node or re-establishing a link with a previously connected node) **shall** ⁽⁴⁾ reset the ARQ machine for the EXPEDITED DATA-state.

The processing of D_PDUs containing Expedited Data **shall** ⁽⁵⁾ proceed according to a *C_PDU-level/stop-and-wait* protocol, as follows:

- No new Expedited-C_PDUs **shall** ⁽⁶⁾ be accepted for service until the last D_PDU of a prior Expedited-C_PDU has been acknowledged.
- Each time a D_EXPEDITED_UNIDATA_REQUEST Primitive is accepted for service, the Expedited Data D_PDU Frame Sequence Number counter **shall** ⁽⁷⁾ be reset to the value zero and the C_PDU ID counter **shall** ⁽⁸⁾ be incremented modulo-16.

Upon exiting the EXPEDITED DATA(CONNECTED) state to another state, all unsent EXPEDITED-DATA C_PDUs (and portions of C_PDUs) **shall** ⁽⁹⁾ be discarded.

Similarly at a receiving node, transition from the EXPEDITED DATA(CONNECTED) state to another state **shall** ⁽¹⁰⁾ result in the deletion of a partially assembled C_PDU. [Note: The decision to delete partially processed C_PDUs when a transition is made to another data transfer state reflects the primary usage of the expedited data transfer service, i.e. the transfer of short, high priority, upper layer peer-to-peer PDUs that require immediate acknowledgement.]

C.6.2 D_PDU Frame-Sequence Numbers and Flow-Control when in the DATA STATE

Because frame numbers are operated upon using modulo arithmetic, it is convenient to represent the ARQ sliding-window that controls D_PDU flow as a segment of a circular buffer as shown in Figure C-43.

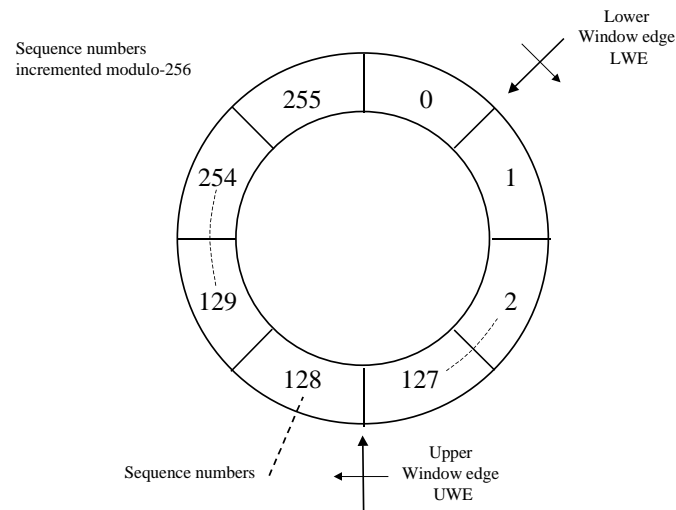


Figure C-43. Sequence-number Space: Integers 0..255

The Data Transfer Sublayer **shall** ⁽¹⁾ satisfy the following requirements for D_PDU flow control:

- a) Each node **shall** ⁽²⁾ maintain a transmit and a receive flow-control window buffer for each connection supported.
- b) The frame sequence numbers **shall** ⁽³⁾ be assigned uniquely and sequentially in an ascending modulo 256 order during the segmentation of the C_PDU into D_PDUs.
- c) The frame sequence numbers **shall** ⁽⁴⁾ not be released for reuse with another D_PDU until the receiving node has acknowledged the D_PDU to which the number is assigned.
- d) The transmit lower window edge (TX LWE) **shall** ⁽⁵⁾ indicate the lowest-numbered outstanding unacknowledged D_PDU (lowest-numbered allowing for the modulo 256 operations).
- e) The transmit upper window edge (TX UWE) **shall** ⁽⁶⁾ be the number of the last new D_PDU that was transmitted (highest D_PDU number, allowing for the modulo 256 operation).
- f) The difference (as a modulo-256 arithmetic operator) between the TX UWE and TX LWE – 1 **shall** ⁽⁷⁾ be equal to the “current transmitter window size”.
- g) The “maximum window size” **shall** ⁽⁸⁾ equal 128.
- h) The “maximum allowable window size” may be a node-configurable parameter (this is recommended) and **shall** ⁽⁹⁾ not exceed the “maximum window size”.
- i) The “current transmitter window size” at any moment is variable as a function of the current TX UWE and TX LWE and **shall** ⁽¹⁰⁾ not exceed the “maximum allowable window size”.

This allows for no more than 128 (“maximum window size”) outstanding unacknowledged D_PDUs in any circumstance.

- j) If the “current transmitter window size” equals the “maximum allowable window size”, no additional new D_PDUs **shall**⁽¹¹⁾ be transmitted until the TX LWE has been advanced and the newly computed difference (modulo 256) between the TX UWE and the TX LWE – 1 is less than the maximum allowable window size.
- k) The receive lower window edge (RX LWE) **shall**⁽¹²⁾ indicate the oldest D_PDU number that has not been received (lowest D_PDU number, allowing for modulo 256 arithmetic operations).
- l) The receive lower window edge (RX LWE) **shall**⁽¹³⁾ not be decreased when retransmitted D_PDUs are received that are copies of D_PDUs received previously.
- m) The receive upper window edge (RX UWE) **shall**⁽¹⁴⁾ be the frame-sequence number of the last new D_PDU received. [Note: More explicitly, the RX UWE is the Frame Sequence Number of the received D_PDU which is the greatest distance (modulo 256) from the RX LWE. The RX UWE increases monotonically as D_PDUs with higher (mod 256) TX Frame Sequence Numbers are received; it does not move back when retransmitted D_PDUs are received.]
- n) D_PDUs with TX FSN falling outside the maximum window size of 128 **shall**⁽¹⁴⁾ not be accepted or acknowledged by the receiver node.
- o) If the value of the RX LWE field in any ACK Type 1 or DATA-ACK Type 2 D_PDU is greater than the TX LWE, the Data Transfer Sublayer **shall**⁽¹⁶⁾ declare all D_PDUs with Frame Sequence Numbers between the TX LWE and the RX LWE value as acknowledged, and **shall**⁽¹⁷⁾ advance the TX LWE by setting it equal to the value of the RX LWE.
- p) The initial condition of the window edge pointers (e.g., on the initialization of a new link) **shall**⁽¹⁸⁾ be as follows:
 - TX LWE = 0
 - TX UWE = 255
 - RX LWE = 0
 - RX UWE = 255

[Note that, although the flow control limitations would allow as many as 128 D_PDUs to be transmitted in one transmission, other protocol parameters will also effect this in an indirect way; the structure of the EOT parameter allows a maximum transmission interval of about 2 minutes. If a D_PDU size of 200 bytes is used at 75 bps, only 5 D_PDUs can be transmitted. However, even if a node receives no acknowledgements, it would be possible to transmit many more D_PDUs before transmission would halt due to flow control restrictions.]

The nominal relationships between the ARQ Flow-Control variables specified above are shown in Figure C-44.

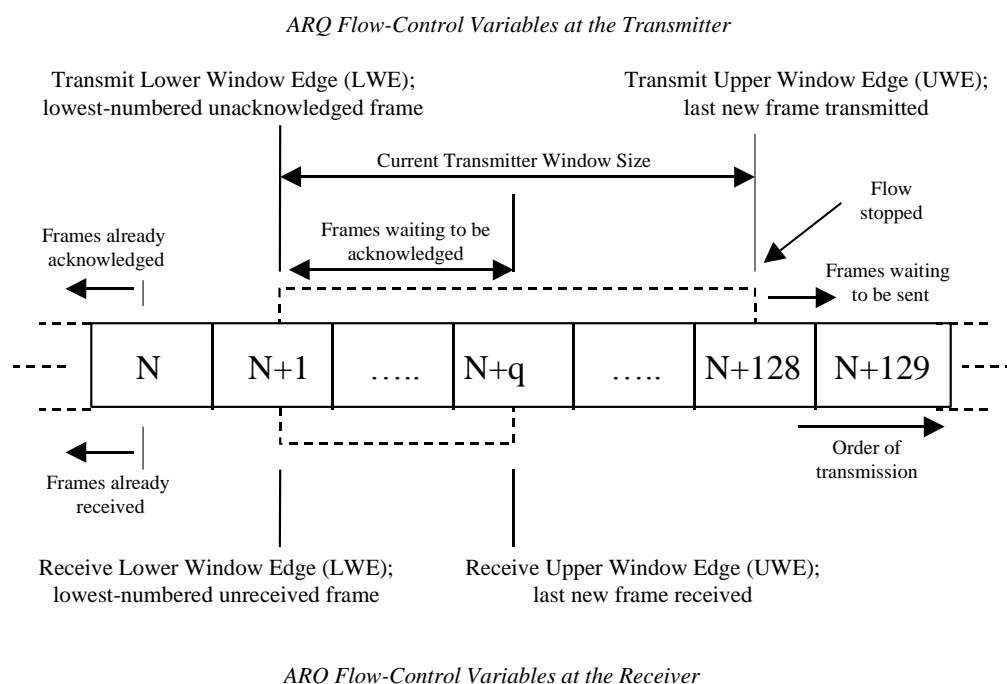


Figure C-44. ARQ Flow Control Window Variables

C.6.3 Synchronisation of the ARQ Machine

Procedures for synchronizing the ARQ machine for a link are specified in the following paragraphs.

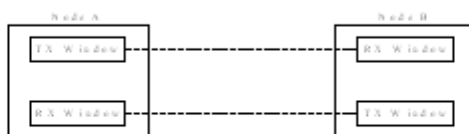
C.6.3.1 Initial Synchronisation

On starting an ARQ machine (i.e. establishing a link with a new node, or establishing a new link with a previously connected node), the transmit and receive ARQ window-edge pointers **shall** ⁽¹⁾ be set to the initial values (as defined in C.6.2).

On an HF circuit, it may be desirable to revive a data state connection that has, for example, timed out partway through a file transmission due to a temporary worsening of the propagation conditions. In this case, the loss of data will be minimised if the ARQ machine associated with the connection is re-activated (i.e. the transmit and receive ARQ window-edge pointers are not re-initialised on re-establishing the link). However, this assumption may, for various reasons, *not* be valid (for example, because one of the nodes has experienced a power failure before re-activation) and so a synchronisation verification procedure **shall** ⁽²⁾ be executed whenever a link is re-established.

C.6.3.2 Verification and Maintenance of Synchronisation

The following synchronisation verification procedure **shall** ⁽¹⁾ be used to verify on an *ongoing basis* if the peer ARQ processes are in synchronisation and, if required, to effect a reset or re-synchronisation of the peer ARQ window pointers.³



This figure illustrates the fact that, at each end of the node, there is one transmit and one receive window. The dotted lines show which pairs of windows need to remain in synchronisation.

C.6.3.2.1 Synchronisation Tests Performed at the Destination Node

Synchronisation tests performed at the destination node make use of the TX lower window edge (LWE) and TX upper window edge (UWE) flags in conjunction with the TX FRAME SEQ # contained in the header of DATA-ONLY and DATA-ACK frames. The appropriate flag is raised (value = 1) when the TX FRAME SEQ # corresponds to the originating node's transmit ARQ LWE or UWE pointers. The following tests **shall** ⁽²⁾ be used to detect *loss of synchronisation*.

Test 1 : Transmit Upper Window Edge

The purpose of this test is to ensure that the TX UWE of the originating node is within the valid range defined by the destination node window edge pointers. This test **shall** ⁽³⁾ be carried out whenever a D_PDU is received with its TX UWE flag set. If the TX UWE passes this test then the two nodes are *in synchronisation*.

Equation 1 :

$$\text{IN SYNC} = (\text{TX UWE} \geq \text{RX UWE}) \text{ AND } (\text{TX UWE} \leq \text{MAX WIN SIZE} - 1 + \text{RX LWE})$$

If the peer ARQ machines are properly synchronised, the TX UWE cannot be less than the RX UWE (as this would indicate that a D_PDU has been received which has not been transmitted). This condition is expressed by the first part of equation 1. It also cannot exceed the limits defined by the maximum window size of 128 and therefore cannot be greater than the {RX LWE + MAX WIN SIZE - 1} (modulo 256) [4]. This condition is expressed by the second part

³ Verification of synchronisation on an ongoing basis and, if required, re-synchronisation is the responsibility of the Data Transfer Sublayer. However, under some circumstances a reset or re-synchronisation may be initiated by the Management Sublayer, e.g. following the (re)establishment of a link and as part of some link maintenance procedures.

⁴ All the synchronisation tests assume a fixed window size equivalent to the maximum window size of 128 frames. Although the STANAG permits the use of a variable transmit window size, this information is not transmitted over the air. The destination node therefore has no knowledge of the window size of the originating node and so cannot take it

of equation 1. Equation 1 can therefore be used to establish whether the TX UWE is *in synchronisation* with the RX window edges and *loss of synchronisation* has occurred if this equation is not satisfied.

The *in synchronisation* and *out of synchronisation* regions of the FSN circle described by equation 1 are illustrated graphically in Figure C-45.

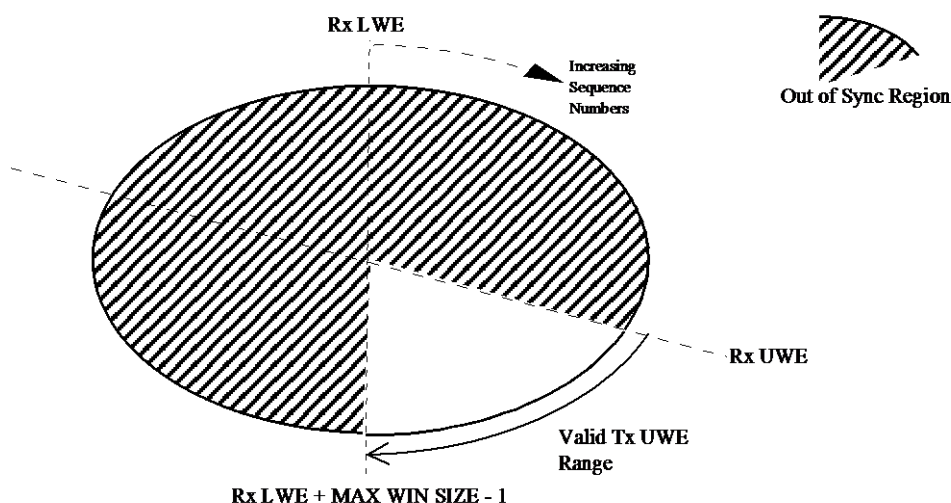


Figure C-45. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 1)

Test 2 : Transmit Lower Window Edge

The purpose of this test is to ensure that the TX LWE of the originating node is within the valid range defined by the destination node window edge pointers. This test **shall** ⁽⁴⁾ be carried out whenever a D_PDU is received with its TX LWE flag set. If the TX LWE passes this test then the two nodes are *in synchronisation*.

Equation 2 :

$$\text{IN SYNC} = (\text{TX LWE} \geq \text{RX UWE} - (\text{MAX WIN SIZE} - 1)) \text{ AND } (\text{TX LWE} \leq \text{RX LWE})$$

If the peer ARQ machines are properly synchronised, the TX LWE should not be outside the bounds defined by the maximum window size as seen from the perspective of the destination node. The TX LWE indicated by the incoming D_PDU therefore cannot be less than the $\{\text{RX UWE} - (\text{MAX WIN SIZE} - 1)\}$ (modulo 256). This condition is expressed by the first part of equation 2. The TX LWE also cannot be greater than the RX LWE (as this would indicate that the originating node has marked as acknowledged a frame that the destination node has not yet received). This condition is expressed by the second part of equation 2. Equation 2 can therefore be used to establish whether the TX LWE is *in synchronisation* with the RX window edges and *loss of synchronisation* has occurred if this equation is not satisfied.

The *in synchronisation* and *out of synchronisation* regions of the FSN circle described by equation 2 are illustrated graphically in Figure C-46.

into account in the synchronisation tests. The tests should still be applied when the window size is varied but in this case some out-of-synchronisation conditions will not be detected.

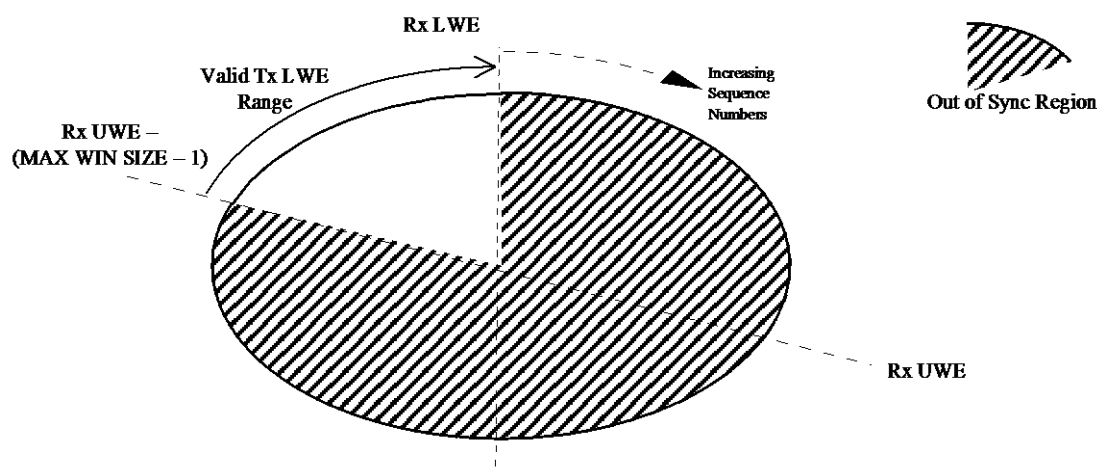


Figure C-46. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 2)

Test 3 : All Received Frames

The purpose of this test is to ensure that the frame sequence number of a frame received from the originating node is within the valid range defined by the destination node window edge pointers. This test **shall** ⁽⁵⁾ be carried out on every DATA and DATA-ACK frame received. If the frame sequence number of the incoming frame passes this test then the two nodes are *in synchronisation*.

Equation 3 :

$$\text{IN SYNC} = (\text{TX FSN} \leq \text{RX LWE} + (\text{MAX WIN SIZE} - 1)) \text{ AND} \\ (\text{TX FSN} \geq \text{RX UWE} - (\text{MAX WIN SIZE} - 1))$$

If either part of equation 3 is not true, then the frame sequence number falls outside the range defined by the maximum window size of 128 frames and *loss of synchronisation* between the two nodes has occurred.

The *in synchronisation* and *out of synchronisation* regions defined by equation 3 are illustrated graphically in Figure C-47.

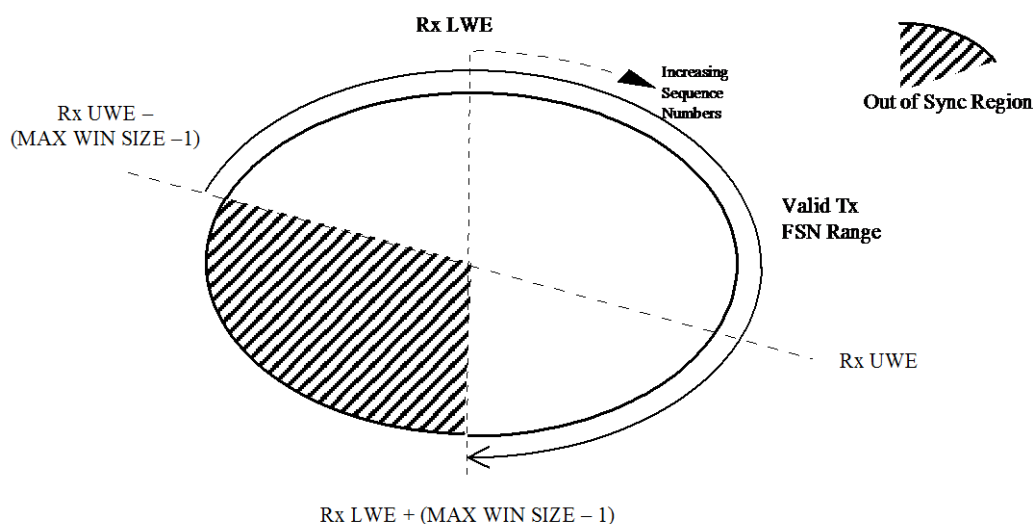


Figure C-47. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 3)

C.6.3.2.2 Synchronisation Tests Performed at the Originating Node

The purpose of tests carried out at the originating node is to ensure that the frame sequence numbers of acknowledged frames are within the range defined by the transmit window edge pointers. These tests **shall** ⁽⁶⁾ be applied whenever a DATA-ACK or ACK-ONLY frame is received.

Test 4 : Receive Lower Window Edge

The value of the RX LWE is included in all DATA-ACK and ACK-ONLY frames. The following test is used to determine whether the RX LWE is within the range defined by the TX window edge pointers. If the RX LWE passes this test then the two nodes are *in synchronisation*.

Equation 4 :

$$\text{IN SYNC} = (\text{RX LWE} \geq \text{TX LWE}) \text{ AND } (\text{RX LWE} \leq \text{TX UWE} + 1)$$

If equation 4 is not satisfied then *loss of synchronisation* has occurred.

The *in synchronisation* and *out of synchronisation* regions defined by equation 4 are illustrated graphically in Figure C-48.

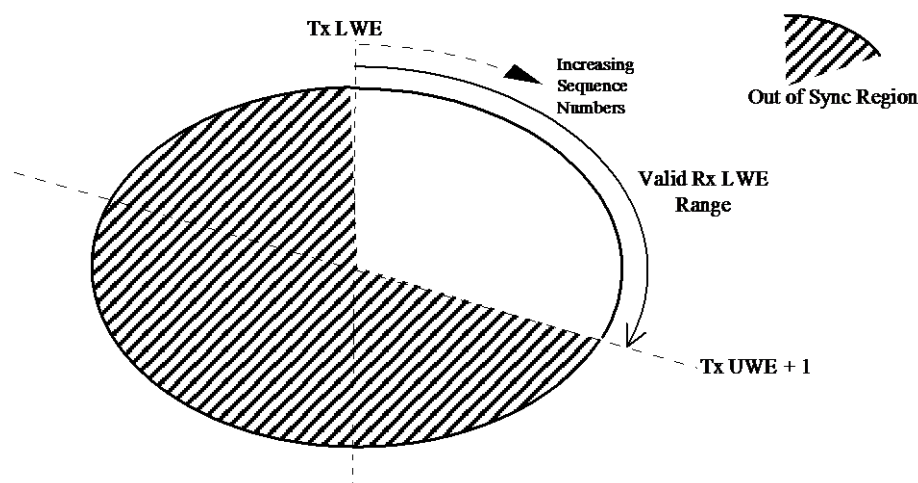


Figure C-48. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 4)

Test 5 : Explicitly Acknowledged Frames

Individual frames may be acknowledged in a DATA-ACK or ACK-ONLY frame by setting bits in the selective ack header field. The frame sequence numbers (FSNs) corresponding to such bits must also fall within the range defined by the transmit window edges if the two nodes are in synchronisation. The following test **shall** ⁽⁷⁾ be used to determine whether acknowledged FSNs fall within the correct range.

Equation 5 :

$$\text{IN SYNC} = (\text{Acknowledged FSN} > \text{TX LWE}) \text{ AND } (\text{Acknowledged FSN} \leq \text{TX UWE})$$

If acknowledged FSNs do not satisfy the condition defined in equation 5 then *loss of synchronisation* has occurred.

The *in synchronisation* and *out of synchronisation* regions of the frame sequence number circle of the originating node are illustrated graphically in Figure C-49.

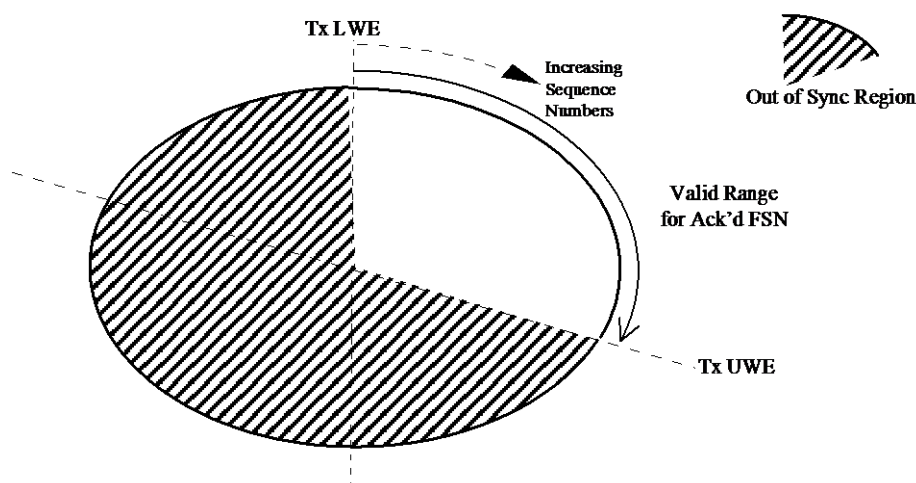


Figure C-49. Showing IN-SYNC and OUT-OF-SYNC Regions of the FSN Circle (Equation 5)

C.6.4 Data Rate Control

Adaptive control of the transmission data-rate by the Data Transfer Sublayer is optional in STANAG 5066. Data Rate Control requires a modem-control interface (virtual or real) between the Data Transfer Sublayer and the attached modem through which changes in the modem data rate and other waveform parameters may be effected by the Data Transfer sublayer. In addition, the Data Transfer Sublayer should be capable of receiving modem-status reports over the modem-control interface. A minimum set of capabilities required and desirable of the modem-control interface are defined in Annex E for information for implementers of this STANAG.

If Data Rate Control is implemented, it **shall** ⁽¹⁾ be implemented as specified in the subsections below.

Nodes that do not implement Data Rate Control **shall** ⁽²⁾ use the appropriate EOW and Management message types specified in Section 3.5 and the protocol defined below to signal this condition to other nodes.

C.6.4.1 Initial (Default) Data Rate

All connections on which the data rate or other modem parameters can be controlled **shall** ⁽¹⁾ be initiated at 300 bps, using short interleaving. The waveform **shall** ⁽²⁾ be selected by the operator during node initialization.⁵

C.6.4.2 Adaptive Data Rate Change Procedure

The receiving node has available to it complete information on the signal it is receiving and is therefore best able to determine the optimum data rate at which data should be sent to it.

The receiving node **may** use the EOW Type 1 message to advise the sending node of the optimum data rate, with the "Other parameters" field of the message set in accordance with Section C.5 to indicate that the message is for advisement, and not a request to change the data

⁵ If the waveform used supports automatic recognition of coding and interleaving parameters, defined initial settings are not necessary.

rate. The action and the final decision remain with the sending node. Mechanisms other than the Type 1 EOW (i.e., Data Rate Change) message **may** be used to arrive at a decision to change data rate; however, these other mechanisms are not defined here and will not be interoperable between vendors. Algorithms to determine when or if the data rate change capability would be exercised are beyond the scope of this STANAG. At a minimum, systems implementing STANAG 5066 **shall** ⁽¹⁾ implement and support data rate changes in accordance with the procedures defined here.

On receiving an EOW Type 1 Data Rate Change message with the "Other-parameters" field indicating that this is a request for a Data Rate Change, a node **shall** ⁽²⁾ comply with the parameters specified in the message unless some specific reason prevents it doing so. Generally this means that the node will initiate a data rate change (DRC) procedure.

(Note: Reasons for failure to comply with the request include lack of remote modem control, or local conditions do not support a change if modem TX and RX data rate must be the same, and are specified for the EOW/Management messages in Section C.5).

Following a decision to change data rate, a node **shall** ⁽³⁾ use TYPE 6 D_PDUs (i.e., MANAGEMENT D_PDUs) containing Type 1 and Type 2 EOW messages to implement and coordinate the change.

The data-rate change procedure **shall** ⁽⁴⁾ be executed in the MANAGEMENT (CONNECTED) State in accordance with Section 6.1.

If the waveform and modems used support automatic recognition of coding and interleaving parameters⁶ by the receiver, the data rate change procedures defined here may not be necessary. Following receipt of an EOW DRC advisory message, a system using such a waveform and modem can simply change the transmit data rate. Annex H provides more details on this subject as information for implementers.

Data rate changes **shall** ⁽⁵⁾ be effective only for a single connection. If a node has a number of connections active with different nodes, the data rate change decisions and procedures **shall** ⁽⁶⁾ be executed independently for each connection.

The node initiating the data rate change is referred to as the data rate change master (DRC master) for this DRC procedure. Figure C-50 shows a scenario of a successful DRC procedure. The numbers in brackets in the figure, i.e, message [1], are included for reference to the explanatory text.

⁶ Examples of such waveforms include the waveforms defined in Annex G to this document, and the MIL-STD-188-110A single tone waveform.

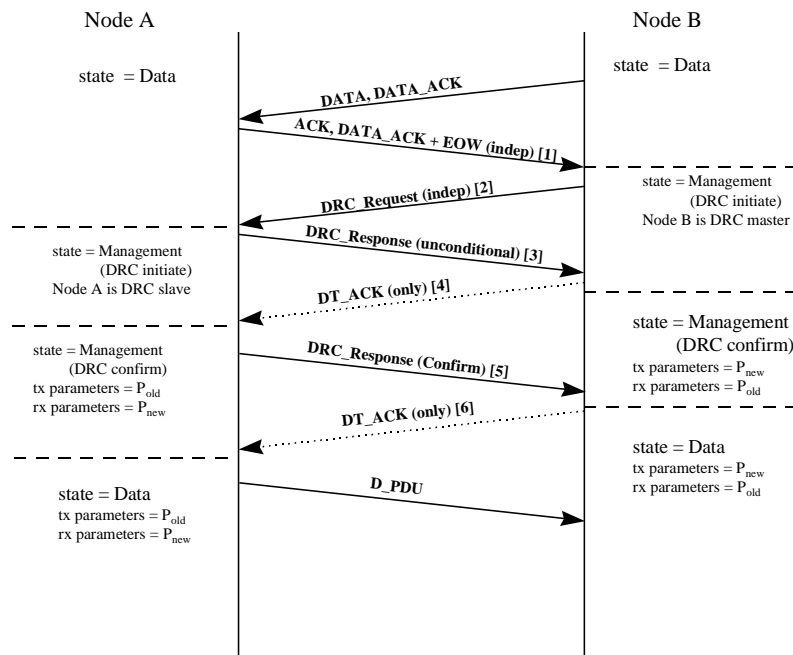


Figure C-50. Data Rate Change Procedure (Scenario 1)

The message numbered [1] in the figure indicates an EOW advisory message. The DRC master **shall** ⁽⁷⁾ send a DRC Request Message, i.e., a MANAGEMENT (TYPE 6) D_PDU containing a DRC Request (Type 1) EOW Message [2], with the parameters equal to the intended new transmit data rate for the DRC master.

The “Other Parameters” field of the Type 1 Management message **shall** ⁽⁸⁾ be set to indicate the data rate capabilities of the modem in use at the DRC master for this connection, and to indicate that this is a request for a change, and not an advisory message. In the scenario of Figure C-50, the modem at the DRC master has independent transmit and receive data rate.

The only state transition allowed due to timeout during the Data-Rate-Change procedure **shall** ⁽⁹⁾ be to the IDLE(UNCONNECTED) state as specified in Section C.6.1, with a D_CONNECTION_LOST primitive sent to the Channel Access Sublayer.

[Note: The management message stop-and-wait protocol, with the repetitions required below, is extremely robust. If it is not possible to complete the procedure in the management state, it is unlikely that returning to another connected state will be productive.]

Timeouts **shall** ⁽¹⁰⁾ be set to allow a number of retransmissions before failure is declared and the D_CONNECTION_LOST primitive issued; the number of retransmissions before a timeout **shall** ⁽¹¹⁾ be configurable in the implementation with a default value of 3.

If this procedure was initiated in response to a DRC Request (Type 1) Advisory message, the modem data rate and interleaving parameters **shall** ⁽¹²⁾ be identical to the parameters in the EOW Type 1 DRC Advisory message, unless they specify a speed for which the DRC master is not equipped.

D_PDUs containing a DRC Request (Type 1) EOW message **should** be repeated depending on the data rate at which it is transmitted, as well as on the specific waveform and interleaver used. The number of D_PDU transmissions **should** be specified to (nearly) fill the interleave buffer for the waveform in use.

[N.B. Table C-26 from Edition 1, and the requirement for specific values for DRC Request repetitions, has been deleted in Amendment 1.]

Repeated MANAGEMENT D_PDUs containing a DRC Request (Type 1) Management message **shall**⁽¹⁵⁾ have the same FRAME ID NUMBER in each of the copies.

Valid EOT information **shall**⁽¹⁶⁾ be supplied in each repeated D_PDU containing a DRC Request (Type 1) Management message, updated as required to denote the end of transmission of all D_PDU messages in the transmission interval (and not the end of the individual D_PDU containing the EOT field).

The same EOT value **shall**⁽¹⁷⁾ not be transmitted in each repeated D_PDU containing a DRC Request (Type 1) Management message unless necessary due to EOT resolution and round-up errors, i.e., because the D_PDU duration is less than half the EOT resolution.

A node (referred to as the DRC slave) that receives a MANAGEMENT D_PDU addressed to it and containing a DRC_Request Type 1 Management message **shall**⁽¹⁸⁾ transition to the Management State as specified in Section C.6.1.

The DRC slave **shall**⁽¹⁹⁾ respond to the DRC_Request D_PDU with a DRC_Response message, i.e., a TYPE 6 MANAGEMENT D_PDU containing a Type 2 DRC Response EOW message, as follows:

- The DRC Response message **shall**⁽²⁰⁾ indicate either “accept” or “refuse”, in accordance with the MANAGEMENT message specifications of Section 3.5, indicating the DRC Slave’s capability to change the modem parameters or not.
- If the DRC slave accepts the DRC_Request, the “reason” field **shall**⁽²¹⁾ indicate either “unconditional acceptance” or “Tx and Rx parameters must be the same”.
- If the DRC slave refuses the request, the reason field **shall**⁽²²⁾ indicate the reason for the refusal.
[Note: Only the reasons specified in the Section C.5.2 for the DRC Response message are valid reasons for refusing a DRC_Request. The reasons that apply to modem capabilities imply that all attempts to change that modem parameter should stop (for the duration of the current link with the node that gives this reason for refusing a DRC).]
- The “Not consistent with local conditions” parameter **shall**⁽²³⁾ only be used to refuse a DRC_Request which indicates a less robust mode (i.e., higher data rate or shorter interleaver) that cannot be supported with the node’s current local conditions for noise or error rate.
[Note: This response should only be used in the event that conditions have changed at the DRC slave between the time that the last EOW advisory message was sent and the DRC_Request was received. Subsequent DRC procedures may be initiated after additional information on channel conditions is obtained and exchanged.]

[Note: The DRC Slave’s response is shown as DRC_Response [3] in the figure. The figure shows an example in which the modem at the DRC slave also has independent transmit and receive data rate. Multiple copies of this D_PDU would be transmitted depending on the modem parameters in accordance with the specification; these repeated copies are not shown in the figure. *Reference to Table C-26 has been deleted*]

After receiving the DRC_Response message the DRC master **shall** ⁽²⁴⁾ review its contents and determine the appropriate response, e.g., the DT_ACK(only) [4], in accordance with Table C-27, below.

Table C-27. DRC_Responses and Allowed DRC Master Actions

| <i>DRC_Response</i> | <i>DRC_Response Reason</i> | <i>Action allowed by DRC master</i> |
|---------------------|--|--|
| accept | unconditional | a) Send DT_ACK [DT_ACK = TYPE 6 D_PDU w/ ACK field set] |
| accept | transmit and receive parameters must be the same | a) Send DT_ACK , or b) Send DRC_Response (cancel), or c) Send new DRC Request ^{note 1} |
| refuse | not possible to change modem data rate | a) send DRC_Response (cancel) ^{note 2} , or b) send DRC_Request ^{note 3} (with DT_ACK) |
| refuse | not possible to change modem interleave | a) Send DRC_Response (cancel) ^{note 2} , or b) Send DRC_Request ^{note 4} (with DT_ACK) |
| refuse | not possible to change modem data rate or interleave | a) Send DRC_Response (cancel) ^{note 2} (with DT_ACK) |
| refuse | not consistent with local conditions ^(note 5) | a) Send DRC_Response (cancel) ^{note 2} , or b) Send DRC_Request ^{note 6} (with DT_ACK) |

Notes to Table C-27 and further DRC requirements are as follows:

Note 1: If the procedure is initiated in response to EOW Type 1 message, the DRC master should already know that the DRC slave's transmit and receive parameters must be the same. Therefore, the DRC master **shall** ⁽²⁵⁾ reply with a DT_ACK, i.e., a MANAGEMENT (TYPE 6) D_PDU with the ACK field set equal to one, accepting that the new parameters will apply to both transmit and receive.

Note 2: A DRC Slave that refused a change request **shall** ⁽²⁶⁾ acknowledge the DRC Response (cancel) message with a DT_ACK only and then terminate the DRC procedure.

Note 3: If the DRC Slave refused the change request because it is not possible to change its modem data rate, a new DRC_Request may be sent by the DRC Master to request a different interleave setting at the same data rate.

Note 4: If the DRC Slave refused the change request because it is not possible to change its interleave setting, a new DRC_Request may be sent by the DRC Master to request a different data rate setting at the same interleave setting.

Note 5: As required by previous paragraphs, the DRC Slave may refuse the request with “Reason = not consistent with local conditions “ if and only if it is in response to a request for a less robust set of parameters. For example, a request for a higher data rate and/or shorter interleaver than currently in use would be rejected by the DRC Slave if it has determined that these cannot be supported by the current link conditions at the receiver.

Note 6: If the nodes make use of the EOW Type 1 (Advisory) message to initiate the DRC procedure, the master **shall** ⁽²⁷⁾ send the DRC_Response (cancel), and await an updated EOW recommendation before initiating another DRC procedure. If EOW Type 1 messages are not used, DRC_Request may be sent by master to request different modem parameters which may be consistent with the local conditions.

In the table and notes in the preceding paragraphs, a DT_ACK refers to a Data Transfer Sublayer acknowledgement of the preceding MANAGEMENT message (shown at [4] in Figure C-50). A DT_ACK reply indicates that the node has nothing further to communicate with respect to the DRC procedure.

If a DT_ACK (with no further management message) is sent in reply to a DRC_Response “accept” (as shown in Figure C-50), the nodes **shall** ⁽²⁸⁾ change their respective modem parameters and proceed to the “confirmation” phase.

The DRC slave **shall** ⁽²⁹⁾ NOT change its modem parameters until it has received a DT_ACK (with no further management message) from the DRC master.

If a DT_ACK (with no further management message) is sent by the DRC slave in reply to a DRC_Response “cancel”, both nodes **shall** ⁽³⁰⁾ abandon the procedure and return to the prior state without changing modem parameters.

After abandoning a DRC procedure because of failure, if node A (formerly the DRC slave) has no queued data or acknowledgements to send to node B, it **shall** ⁽³¹⁾ send a data D_PDU, expedited data D_PDU, or non-ARQ D_PDU, with zero data attached.

[Note: In the scenario given in Figure C-50, the slave’s DRC_Response with an “accept/unconditional” message generates the allowed DT_ACK from the DRC master. After sending the DT_ACK [4], the master changes its modem parameters and waits to receive a DRC Confirm message (type 2 MANAGEMENT message with response set to “confirm” and reason set to “none”) from node A (“confirmation phase”). After receiving the DT_ACK [4], the slave changes its modem parameters and transmits a DRC Confirm message [5] to the master.]

On receiving the DRC Confirm message, the DRC Master **shall** ⁽³²⁾ respond with a DT_ACK and then return to the processing state it was in before executing the DRC procedure.

After sending the DRC Confirm message [5] to the master and receiving the DT_ACK from the master, the slave **shall** ⁽³³⁾ return to the processing state it was in before executing the DRC procedure and send any queued D_PDUs to node B. If node A (formerly the DRC slave) has no queued data to send to node B, it **shall** ⁽³⁴⁾ send a DATA_ONLY D_PDU, ACK_ONLY D_PDU, DATA_ACK D_PDU, EXPEDITED DATA-ONLY D_PDU, EXPEDITED ACK-ONLY D_PDU, NON-ARQ D_PDU or EXPEDITED NON-ARQ D_PDU with zero data attached, and the C_PDU Segment Size field set equal to zero (0).

C.6.4.3 Additional DRC Scenarios

The following figures provide additional scenarios for the DRC procedure in different conditions, and are provided as implementation guidance only.

Figure C-51 gives a scenario in which node B, due to HF modem limitations, must operate with the same HF modem transmit and receive parameters. The following numbered paragraphs correspond to the numbers in [brackets] in Figure C-51.

1. In this transmission, the advisory message sent by A carries the recommended Tx speed for B (You should Tx at x bps) and also tells B about the capabilities of A's modem (that A's modem can operate at independent Tx and Rx data rates).
2. The DRC_Request message from B carries the requested new modem parameters for B: "I (B) will transmit at x bps with y interleaving"; these parameters will be as recommended by A, in accordance with Section C.6.4.2. The message also tells A about the capabilities of B's modem, in this case, if I change my Tx parameters (and your Rx parameters), I will have to change my Rx parameters (and your Tx parameters) as well.)
3. DRC_Response message accepts the requested change.
4. The DT_ACK confirms that B received the prior message and triggers B to change its parameters.
5. This DRC_Response (confirm) is a confirmation that the link is up at the new parameters.

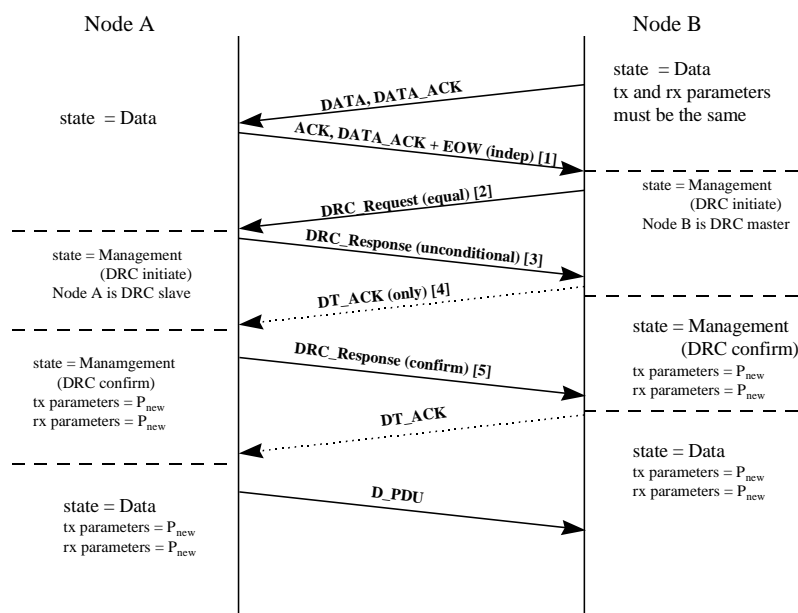


Figure C-51. Data Rate Change Procedure (Scenario 2)

Figure C-52 shows a scenario in which the procedure is refused by the slave, and hence cancelled by the DRC master. In this example, node B has a modem which must have the same transmit and receive parameters. The requested new parameters would then apply to the transmit and receive parameters at both nodes. The following numbered paragraphs correspond to the numbers in [brackets] in Figure C-50.

1. In this transmission, the EOW Type 1 advisory message sent by A carries the recommended Tx speed for B (You should Tx at x bps) and also tells B about the capabilities of A's modem (that A's modem can operate at independent Tx and Rx data rates).

2. The DRC_Request message from B carries the requested new modem parameters for B: “I (B) will transmit at x bps with y interleaving”; these parameters will be as recommended by A in accordance with Section C.6.4.2. The message also tells A about the capabilities of B’s modem, in this case, if I change my Tx parameters (and your Rx parameters), I will have to change my Rx parameters (and your Tx parameters) as well. The net result is that both nodes have the same Rx and Tx parameters.
3. DRC_Response message refuses the requested change; a likely reason would be that the local conditions at node A do not support use of the proposed parameters.
4. The DRC master responds by cancelling the procedure.
5. The DT_ACK confirms that B received the prior message and terminates the procedure.

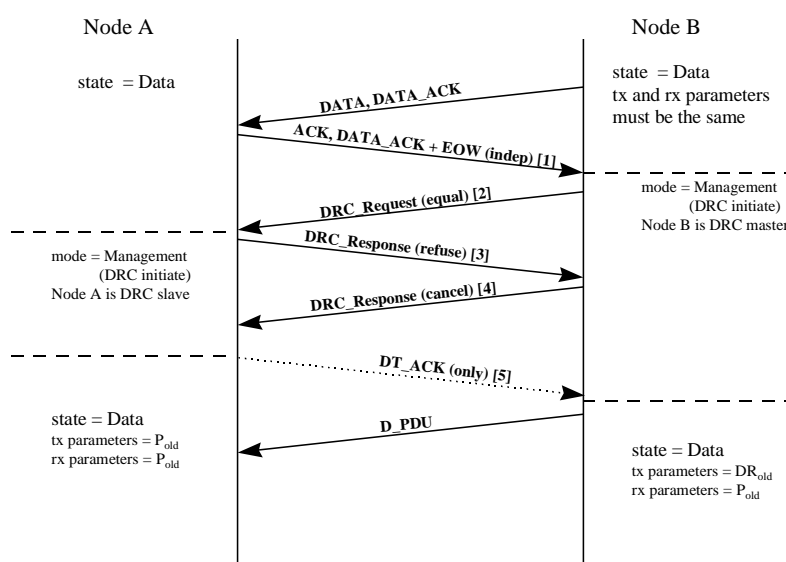


Figure C-52. Data Rate Change Procedure (Scenario 3)

C.6.5 Procedure for use of Type 3 RESET/WIN RESYNC D_PDU

The Type 3 RESET/WIN RESYNC D_PDU, specified in Section C.3.6, supports a number of different resynchronization functions. The procedure for the FULL RESET function is specified in this section.

A FULL RESET procedure **shall** ⁽¹⁾ initiated by a node sending a Type 3 RESET/WIN RESYNC D_PDU with field values as follows:

- The FULL RESET CMD flag **shall** ⁽²⁾ be set to 1;
- The RESET FRAME ID NUMBER **shall** ⁽³⁾ be selected from RESET FRAME ID NUMBER sequence;
- The NEW RECEIVE LWE field **shall** ⁽³⁾ be reset to zero;
- The RESET TX WIN RQST flag **shall** ⁽⁴⁾ be reset to zero;
- The RESET RX WIN CMD flag **shall** ⁽⁵⁾ be reset to zero.

The Type 3 D_PDU described immediately above is defined to be a FULL RESET CMD D_PDU. A node receiving a FULL RESET CMD D_PDU **shall** ⁽⁶⁾ proceed as follows:

- The node **shall** ⁽⁷⁾ set the transmit and receive window pointers to initial values (as defined in Section C.6.2);

- The node **shall** ⁽⁸⁾ discard from its transmit queue any partially completed C_PDUs;
- The node **shall** ⁽⁹⁾ flush its receive buffers;
- The node **shall** ⁽¹⁰⁾ respond to the originator of the FULL-RESET-CMD D_PDU by sending it a RESET/WIN RESYNC (Type 3) D_PDU with field values set as follows:
 - The RESET ACK flag **shall** ⁽¹¹⁾ be set to 1;
 - The NEW RECEIVE LWL and RESET FRAME ID NUMBER fields **shall** ⁽¹²⁾ be reset to zero;
 - The RESET TX WIN RQST, FULL RESET CMD and RESET RX WIN CMD flags **shall** ⁽¹³⁾ be reset to zero.

The D_PDU described immediately above is defined to be a FULL-RESET-ACK D_PDU. The FULL RESET ACK D_PDU **shall** ⁽¹⁴⁾ be sent only in response to the FULL RESET CMD D_PDU.

On receiving the FULL-RESET-ACK D_PDU, the node initiating the FULL RESET procedure **shall** ⁽¹⁵⁾ proceed as follows:

- The node **shall** ⁽¹⁶⁾ set the transmit and receive window pointers to initial values (as defined in C.6.2)
- The node **shall** ⁽¹⁷⁾ discard from its transmit queue any partially completed C_PDUs
- The node **shall** ⁽¹⁸⁾ flush its receive buffers

This concludes the FULL RESET procedure.

Annex D: Interface between Data Transfer Sublayer and Communications Equipment (mandatory)

The interface between the Data Transfer Sublayer and the communications equipment **shall** ⁽¹⁾ be as defined in this Annex. This is currently cryptographic equipment but in time, cryptographic services may move to the application layer.

If the Data-Transfer-Sublayer connection point to the communication equipment is not to a cryptographic device, this definition **shall** ⁽²⁾ apply to the interface between the Data Transfer Sublayer and the modem.

To accommodate requirements imposed by the communications equipment, an arbitrary number of 'pre-fill characters **may** ⁽¹⁾ be transmitted prior to the first valid DPDU in a transmission interval. Likewise, an arbitrary number of 'post-fill' characters **may** ⁽²⁾ be transmitted following the last valid DPDU in a transmission interval. The value of the EOT assigned to each DPDU in the transmission interval **must** be computed accounting for any post-fill characters in the transmission interval. Extraneous characters **shall not** be inserted between any valid DPDUs in the transmission interval.

The interface **shall** ⁽³⁾ be a synchronous serial digital data interface.

[Note: This requirement may be waived for an implementation if it can be shown that the communication equipment used with the sublayer protocols, i.e., the cryptographic equipment or modem, removes any start-bits, stop-bits, or other character-framing bits associated with the interface. Many current implementations of the STANAG 4285 and MIL-STD-188-110A waveforms transmit any start and stop bits that are present on the asynchronous baseband digital interface to the modem, but there is no real requirement in these respective standards for this. Modems may be implemented that allow independent specification of the character-framing and synchronization for the baseband interface and over the air gap

The line-drivers and receivers for the interface **shall** ⁽⁴⁾ be configurable for either balanced or unbalanced connection, in accordance with EIA-232D/423 for unbalanced connections and EIA-422 for balanced connections.

With respect to functional roles on the interface, the Data Transfer Sublayer **shall** ⁽⁵⁾ be hosted in a Data Terminal Equipment (DTE).

The clock source for the data output from the DTE (i.e, DTE data out) on the interface **shall** ⁽⁶⁾ be either configurable or from the DCE (i.e, either the cryptographic equipment or the modem).

The clock source for the data input to the DTE (i.e., DTE data input) **shall** ⁽⁷⁾ be from the DCE (i.e, either the cryptographic equipment or the modem).

The interface **shall** ⁽⁸⁾ provide full hardware-level handshaking for flow-control, in accordance with any standard recommendations.

Compatibility with MIL-STD-188-114 polarity, levels, and slew rates additionally may be required for interoperability with existing (cryptographic) equipment.

BLANK PAGE BLANCHE

Annex E: HF Modem remote control interface (information only)

This Annex defines a common remote control interface with a minimum set of modem commands required for operation of the system. This Annex has not been implemented or tested. The requirements of this Annex are provided as information to implementers and are not mandatory for an implementation to be in compliance with the STANAG.

The electrical interface for remote control of the modem shall be configurable, with EIA-232D/423 for unbalanced connections and EIA-422 for balanced connections. The remote control interface shall include transmit data and receive data.

Remote control commands shall consist of ASCII alpha-numeric strings with 8 data bits, even parity, and one stop bit, at 9600 bps.

At minimum, the following commands shall be supported. In all cases, the <address> field shall be replaced by a decimal number between 0 and 255 which shall indicate the address of a specific modem. The modem shall be configurable to select this address and shall respond only to properly addressed commands.

<address> INITIALIZE This command shall instruct the modem to load a predefined set of operating parameters. The mechanism for defining this set is not standardized.

<address> MODEM RATE <x> (replace x with 75, 150, 300, 600, 1200, 2400, etc; ie, <address> MODEM RATE <1200>); selects the coding applied to the user data before transmission.

<address> MODEM INTERLEAVE <x> (replace x with ZERO, LONG, SHORT, ie, <address> MODEM INTERLEAVE <SHORT>); selects the interleaving used in the modem

<address> MODEM SNR? Returns the signal to noise ratio of the received signal, in the following format:

<address> MODEM SNR=<value> where value is replaced with the SNR in dB.

<address> MODEM IDENTIFY? Returns the manufacturer, model, and software/firmware revision information for a modem, in the following format: <address> MFR <x> ; MODEL <y> SW/FW <software/firmware revision information> where the quantities in angle brackets are replaced by the appropriate ASCII strings. The angle bracket characters shall not appear in any of these strings.

<address> MODEM WAVEFORM <x> Replace x with 4285 (for 4285 with annex E coding), 4529, 110A (MIL-STD-188-110A single tone), or HIGH (for future high data rate waveforms); ie, MODEM WAVEFORM <4529> to select the STANAG 4529 waveform.

BLANK PAGE BLANCHE

ANNEX F. HF-SUBNETWORK CLIENT REQUIREMENTS

(Mandatory)

[N.B. This Amendment replaces Annex F Edition 1 in its entirety.]

This Annex specifies client interactions with the HF subnetwork defined in Annexes A, B, C, and D for a set of clients types (i.e., subnetwork clients) to which the HF subnetwork provides data transport service.

Requirements for these client types are specified and cross-referenced in this Annex in order to provide end-to-end interoperability in an HF data communications subnetwork. Cross-references to other standards documents for client-requirement definition are made to the maximum extent possible.

F.1 STANDARDIZED CLIENT REQUIREMENTS

This annex is a minimum requirement. Unless otherwise noted, compliance with this annex does not prohibit the definition of additional client types or protocols that are not defined herein.

Implementations of clients other than those defined herein **shall not** make assignments of Service Access Point Identifiers (SAP_IDs) or Application Identifiers (APP_IDs) in conflict with the requirements of this document. The SAP IDs and APP_IDs provide the mechanism through which end-systems clients may share and attach to an HF subnetwork without conflict.

F.1.1 Client Implementation for Protocol Conformance

Implementations of STANAG 5066 **shall** provide the MANDATORY interface types and clients noted in the Protocol Implementation Conformance column of Table F-1.

This annex standardizes the physical implementation of a Raw Subnetwork Interface Sublayer (SIS) Socket interface. Implementations of STANAG 5066 **shall** provide a TCP/IP socket-server interface as the physical channel for connecting a client to the HF subnetwork, in accordance with the requirements for the Raw SIS Socket interface defined elsewhere in this Annex.

Another mandatory client provides an IP interface to the HF subnetwork, and transport of IP datagrams over the HF subnetwork.

Implementation and performance requirements for these mandatory client interfaces are found elsewhere in this Annex in the sections cross-referenced by Table F-1.

Implementations of STANAG 5066 additionally **may** provide any of the OPTIONAL client types noted in the Protocol Implementation Conformance column of Table F-1. If provided, however, optional clients **shall** conform to the requirements noted and cross-referenced herein to ensure end-to-end interoperability in the HF data communications profile.

Table F-1: SAP ID Assignments

| Client / Application Type | SAP ID | Protocol Implementation Conformance | Client Requirements Defined in Annex |
|--|------------|-------------------------------------|--------------------------------------|
| Raw SIS Socket Server | all | MANDATORY | F.16 |
| Subnet management client | 0 | OPTIONAL | F.2 |
| Character-Oriented Serial Stream (COSS) Client | 1 | OPTIONAL | F.3 |
| STANAG 4406 Annex E - Tactical Military Message Handling (T-MMHS) Client | 2 | OPTIONAL | F.4 |
| HMTP (HF Mail Transfer Protocol) | 3 | OPTIONAL | F.5 |
| HFPOP (HF Post-Office Protocol) | 4 | OPTIONAL | F.6 |
| Operator orderwire (HFCHAT) | 5 | OPTIONAL | F.7 |
| Reliable Connection-Oriented Protocol (RCOP) w/ Extended Client * | 6 | OPTIONAL | F.8* |
| Unreliable Datagram Oriented Protocol (UDOP) w/ Extended Client * | 7 | OPTIONAL | F.9* |
| ETHER client | 8 | OPTIONAL | F.11 |
| IP client | 9 | MANDATORY | F.12 |
| RESERVED - for future assignment | 10-11 | OPTIONAL | F.13 |
| Compressed File Transport Protocol (CFTP) | 12 | OPTIONAL | F.14 |
| UNASSIGNED – available for arbitrary use | 13-15 | OPTIONAL | F.15 |

* NB: Other standardized clients and protocols will bind to the STANAG 5066 subnetwork using the RCOP or UDOP protocols and uniquely assigned Application Identifiers defined in Annex F.10

F.1.2 Standardized Assignment of Service Access Point Identifiers

Service Access Point Identifiers for all clients applications **shall** be assigned in accordance with Table F-1.

Implementation of client types or protocols that are not in Table F-1 **shall not** make use of field values for Service Access Point Identifiers identified in this Annex.

As noted in Table F-1, other OPTIONAL clients that have not been assigned a SAP ID of their own **shall** bind to the STANAG 5066 subnetwork as an Extended Client type using the RCOP or UDOP protocols and uniquely assigned Application Identifiers defined in Annex F.10.

Client standards and application identifiers for RCOP/UDOP defined in this Edition are cross-referenced in Table F-2. Additional clients **may** be added in future.

Table F-2 Current RCOP/UDOP-based extended client specifications for use with the HF Subnetwork.

| Client Name | Client Protocol Stack | Applicable Specifications | S'5066 Implementation Conformance |
|--|-----------------------|--|-----------------------------------|
| Tactical Military Message Handling System (T-MMHS) | T-MMHS + RCOP/UDOP | S'4406 Annex E, S'5066 Annexes F.4, F.8, F.9, F.10 | OPTIONAL |
| Basic File Transfer Protocol (BFTP) | BFTP + RCOP/UDOP | S'5066 Annexes F.8, F.9, F.10 | OPTIONAL |
| File-Receipt Acknowledgement Protocol (FRAP) | FRAP + RCOP | S'5066 Annexes F.8, F.10 | OPTIONAL |

F.1.3 Summary of Data Multiplexing Options within STANAG 5066

The standardization of client types, SAP IDs, Application IDs, and the mandatory provision of an Internetwork Protocol (IP) client provides a variety of options for multiplexing traffic through the STANAG 5066 protocol stack. Multiplexing options are summarized below:

1. By SAP_ID - 'Base Clients' have their own uniquely assigned Service Access Point Identifier (SAP ID) and traffic flows are based uniquely on the SAP ID;
2. By SAP_ID, IP Address and Socket Number - 'IP clients' attach at a defined SAP ID, and their traffic flows are further based on the socket numbers and IP addresses of the source and destination;
3. By SAP_ID and APP_ID - 'Extended Clients' attach as either a RDOP or UDOP base client, and their traffic flows are further distinguished by their assigned Application Identifier (APP_ID).

The client and interface types are shown in F-1. There is no requirement, or expectation, that a STANAG 5066 implementation would provide all of these interfaces or clients. These multiplexing mechanisms have been defined to enable support for a broad base and diversity of client types in future while minimising conflict in their use of the protocol stack in actual operations.

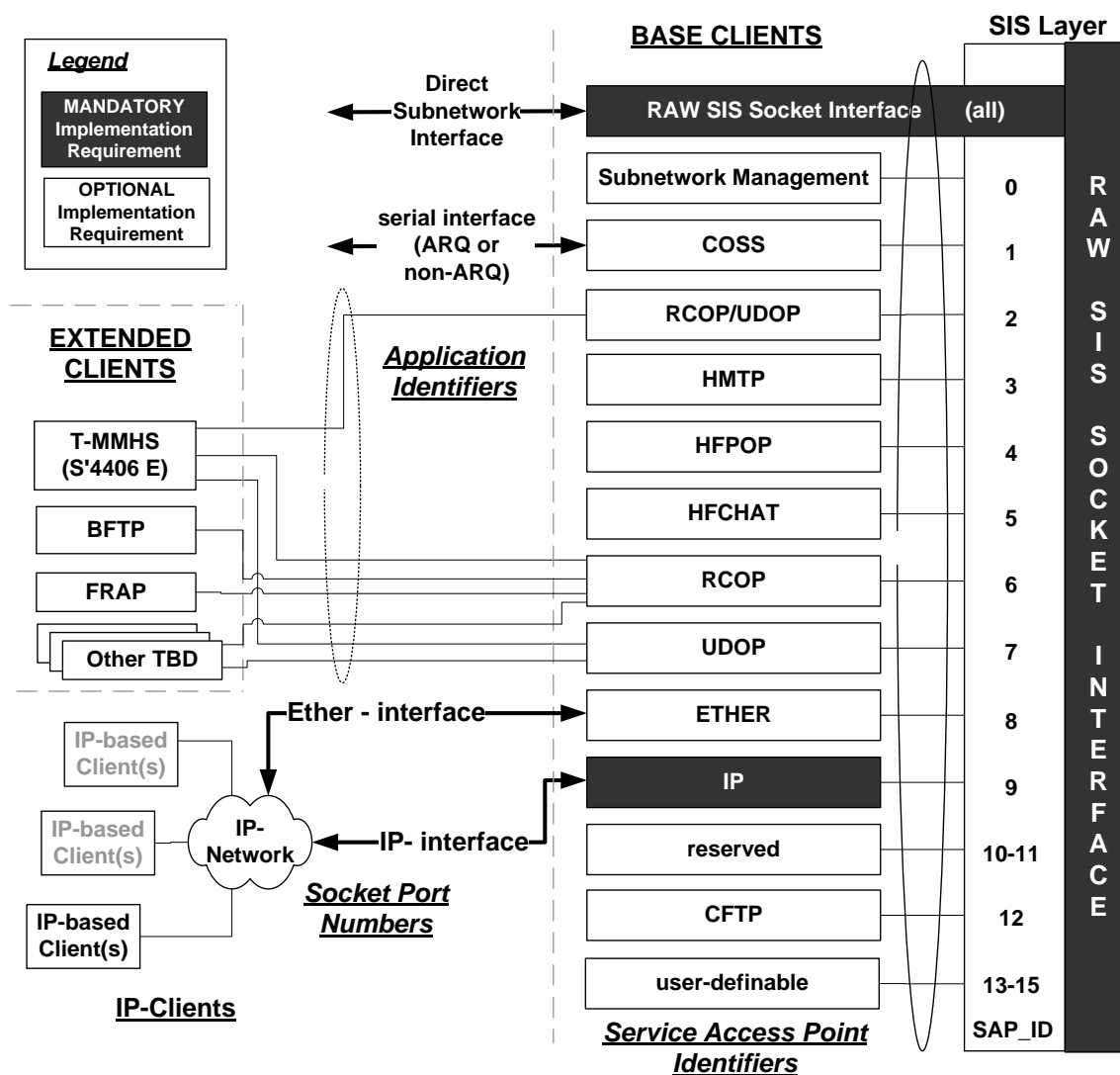


Figure F-1 - Client / Interface types and their attachment points to STANAG 5066 Subnetwork.

F.2 SUBNET MANAGEMENT CLIENT/SERVER

This section defines minimal requirements for control of the local node using the S_MANAGEMENT_MSG_REQUEST, S_MANAGEMENT_MSG_INDICATION, and S_SUBNET_AVAILABILITY primitives, and for coordination with distant subnet management clients or agents using the S_UNIDATA primitives.

Subnetwork management clients **shall** attach to the subnetwork using SAP ID 0. If the client is entitled to submit commands that will change the configuration of the node or subnetwork, the rank of the client **must** be 15.

The nature and specification of peer-to-peer level communication between Subnetwork Management Clients is beyond the scope of this STANAG. Implementations **should** use existing standards for network management such as the Simple Network Management Protocol (SNMP), with peer-to-peer communication through the HF subnetwork via the IP client defined herein.

The management information base (MIB) for STANAG 5066 is currently undefined.

F.3 CHARACTER-ORIENTED SERIAL STREAM (COSS) CLIENT

This section defines a character-oriented serial-transport service for the HF subnetwork. Reliable or unreliable modes of operation can be specified through appropriate selection of the service requirements for the client.

The character-oriented serial stream (COSS) service **may** be used in place of other HF serial transport services, for example a simple modem. When high reliability and end-to-end assurance of data delivery is required, the COSS client **may** use the STANAG 5066 ARQ mode to provide higher data reliability than simple transmission over a conventional modem might afford. When multicast address modes are required, or when data reliability is not, the COSS client **may** use the STANAG 5066 non-ARQ modes. Detailed service requirement specifications are provided at section F.3.3.

The interfaces for the Character-Oriented Serial Stream (COSS) Client are as shown in the Figure below.

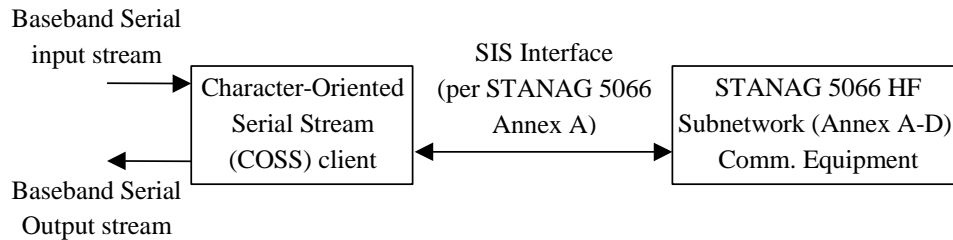


Figure F-2 - COSS Client Interfaces

Requirements for the COSS client are placed on its baseband serial interface, its interface to the HF Subnetwork Interface Sublayer (SIS), and its internal processing. Implementations of the STANAG 5066 **shall**⁽¹⁾ provide a COSS client with a baseband serial interface, and implementations **shall**⁽²⁾ be in accordance with the requirements stated herein.

F.3.1 Base-band Serial Interface

The COSS client **shall** provide a baseband serial interface meeting the following requirements.

a. Physical/Electrical: one of following physical interfaces **shall** be provided:

1. Signalling and connector conforming to EIA/RS-232, EIA/RS-530, configured as Data Communications Equipment (DCE);
2. Signalling and connector conforming to V.35 DCE.

b. Serial Transmission Mode: all of the following transmission modes **shall** be supported (as configuration options):

1. Asynchronous: 1 Start Character, selectable 5, 6, 7, or 8 data bits, and 1 or 2 stop bits.
2. Synchronous: provide/accept clock at 1x Data Rate (other rate-multipliers **may** be supported);
HDLC line control protocol (other synchronous line protocols **may** be supported in addition to HDLC)

c. Flow-Control: all of the following flow-control disciplines **shall** be supported (as configuration options):

1. RTS/CTS, DTR/DTS hardware handshaking;
2. XON/XOFF software flow-control;
3. None.

F.3.2 Character Sets Supported by COSS

The character sets shown in the Table below **shall** be supported by a COSS client:

Table F-3: Character Sets Supported by COSS

| Character Set | Comments |
|----------------|--|
| ITA-2 / Baudot | 5-bit character sets: w/ asynchronous protocol = ITA-2; w/ synchronous protocol = Baudot |
| 6-bit codes | 6-bit character codes |
| ITA-5 / ASCII | ~ASCII - 7-bit character set [NB: the 7.0-unit 64-ary ITA-2 code defined in STANAG 5030 is encoded in this form, in accordance with section F.3.4.4.1] |
| Octet data | Any data presented in arbitrary 8-bit formats |

F.3.3 Subnetwork Service Requirements for COSS

COSS clients **shall** bind to the HF Subnetwork at SAP ID 1.

Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. COSS clients **should not** bind using a Rank = 15.

A COSS client **shall** submit its PDUs to the HF subnetwork using the S_UNIDATA_REQUEST Primitives defined in Annex A of this STANAG.

The address in the primitive **shall** be a STANAG 5066 address corresponding to the HF subnetwork address of the host at which the destination COSS client(s) is/are located. For a COSS client using ARQ service modes, this address **shall** be a unicast (point-to-point) address. For non-ARQ service modes, the COSS client **may** specify either a unicast or a multicast (point-to-multipoint) STANAG 5066 address. [NB: COSS clients using non-ARQ services could use multicast addresses for tailored stream-broadcast applications.]

Specification of the STANAG 5066 destination address for the character stream is outside of the scope of this STANAG. Manual and dynamic modes could be foreseen.

[NB: Manual address configuration requires that an operator establish the destination address for the serial stream data in conformance with a standard operating procedure. This method provides data transparency. It is expected to be the most frequently used approach.

Dynamic address configuration presumes a separate process that scans the serial character stream to extract destination addresses, in some format, from the character stream and then map these

addresses into a valid STANAG 5066 address. This method is not transparent to the character stream (it assumes a well-defined format for the addresses embedded in the character stream). Dynamic addressing could however be used with certain well-defined applications and character streams however, e.g., in the transmission of ACP-127 formatted messages.]

Selection of further service requirements for the S_PRIMITIVE will be a function of STANAG 5066 address-type and the level of link reliability required.

F.3.3.1 Service Requirements for Reliable Serial-Stream Transmission and Point-to-Point Addressing

The default service requirements defined when the client binds to the subnetwork **shall** be as follows:

1. Transmission Mode = ARQ
2. Delivery Confirmation = NONE
3. Deliver in Order = IN-ORDER DELIVERY

F.3.3.2 Service Requirements for Non-ARQ Transmission or Point-to-Multi-Point Addressing

The default service requirements defined when the client binds to the subnetwork **shall** be as follows:

1. Transmission Mode = non-ARQ
2. Delivery Confirmation = NONE
3. Deliver in Order = IN-ORDER DELIVERY

F.3.4 Data Encapsulation Requirements

The characters from the character stream **shall** be encapsulated within S_PRIMITIVES using any one of the modes described herein. Implementation of all modes is **mandatory** for a COSS client.

Selection of any given mode for operation **shall** be a configuration parameter in a COSS client, and dependent on the character-set for which the COSS client is configured.

Selection of any given mode **must** be coordinated at the sending and receiving node for use on a given link, through either standard operating procedure or out-of-band coordination channel.

F.3.4.1 Encapsulation of Arbitrary Octet Data

Octet data in any arbitrary format for the COSS client **shall** be byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE.

The least-significant bit (LSB) of each character received on the serial interface **shall** be aligned with the LSB of the octet.

F.3.4.2 Encapsulation of ITA-5

Characters in ITA-5 format (or other 7-bit character format such as ASCII) for the COSS client **shall** be aligned with the octets in each U_PDU encapsulated in the S_Primitive, one character per octet, as follows.

| | MSB | | | Octet Data | | | | LSB |
|--------------------|-----|-----|---|------------|-------------------|---|---|-----|
| Octet / Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Octet _i | 0 | msb | - | - | ITA5 _j | - | - | lsb |

The least-significant bit (LSB) of each 7-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the MSB bit of each octet **shall** be set to zero for ITA5 Encapsulation.

F.3.4.3 Encapsulation of ITA-2

Two methods of encapsulation of ITA-2 characters are defined:

1. 'Loose-Pack ITA2 Encapsulation' (LPI2E), and
2. 'Dense- Pack ITA2 Encapsulation' (DPI2E).

A COSS client **shall** implement both methods of ITA2 encapsulation.

Selection of either method **must** be coordinated by sending and receiving node for use on a given link, through either standard operating procedure or out-of-band coordination channel.

F.3.4.3.1 'Loose-Pack Encapsulation of ITA-2 characters

The 'Loose-Pack' ITA-2 Encapsulation (LPI2E) algorithm **may** be used for any character set represented as 5-bit symbols. It transports one 5-bit symbol in each octet within the U_PDU field of S_primitives, using the basic packing arrangement defined in the Figure below.

| | MSB | | | Octet Data | | | | LSB |
|--------------------|-----|---|---|------------|---|-------------------|---|-----|
| Byte / Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Octet _i | 0 | 0 | 0 | msb | - | ITA2 _j | - | lsb |

Figure F-3 - Loose-Pack Encapsulation of ITA-2 Characters

The least-significant bit (LSB) of each 5-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the three most-significant bits of each octet **shall** be set to zero for LPI2E.

F.3.4.3.2 'Dense-Pack' Encapsulation of ITA-2 characters

The 'Dense-Pack' ITA-2 Encapsulation (DPI2E) algorithm **may** be used for any character set represented as 5-bit symbols. It efficiently transports three 5-bit symbols in a pair of octets, called an Encapsulation Pair, within the U_PDU field of S_primitives, using the basic packing arrangement defined in the Figure below.

| | MSB | | | Octet Data | | | | LSB |
|-------------|---------------------------|---|---|-------------------------------|---|---|-------------------------|-----|
| Octet / Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| i = 2n | ITA2 _{j+1} - lsb | | | msb - ITA2 _j - lsb | | | | |
| i = 2n + 1 | DP_FLG | msb ITA2 _{j+2} lsb (defined only if DP_FLG = 1) | | | | | msb ITA2 _{j+1} | |

Figure F-4 : Nominal Dense-Pack ITA-2 Encapsulation Pair

F.3.4.3.2.1 Encapsulation Pairs

The first octet of an Encapsulation Pair **shall** be an even-numbered octet in an S_Primitive's U_PDU field (i.e., $i = \{0, 2, 4, 6, \dots\}$, with $i = 0$ the first octet in the U_PDU).

The second octet of an Encapsulation Pair **shall** be an odd-numbered octet in an S_Primitive's U_PDU field (i.e., $i = \{1, 3, 5, 7, \dots\}$, with $i = 0$ the first octet in the U_PDU).

The MSB of the second octet of an Encapsulation Pair **shall**⁽¹⁾ be the DP_FLG ('dense-pack flag') that indicates whether the Encapsulation Pair contains three ITA-2 characters (DP_FLG = 1) or two ITA-2 characters (DP_FLG = 0). Encoding of these cases **shall**⁽²⁾ be performed as follows:

- The first case is defined to be a "Three-into-Two Encapsulation Pair", which **shall** be encoded in accordance with this figure

| Byte / Bit | MSB | | | Octet Data | | | | LSB |
|------------|---------------------------|-----|---|-------------------------------|---|---|-----|-------------------------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| i = 2n | ITA2 _{j+1} - lsb | | | msb - ITA2 _j - lsb | | | | |
| i = 2n + 1 | DP_FL G = 1 | msb | | ITA2 _{j+2} | | | lsb | msb ITA2 _{j+1} |

Figure F-5 : Three-Into-Two Encapsulation Pair

- the second case is defined to be a "Two-into-Two Encapsulation Pair", which **shall** be encoded in accordance with this figure:

| Byte / Bit | MSB | | | Octet Data | | | | LSB |
|--------------|--------------------|---|---|------------|---|----------------|-----|--------------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| $i = 2n$ | $ITA2_{j+1}$ - lsb | | | msb | - | $ITA2_j$ - lsb | | |
| $i = 2n + 1$ | DP_FL G = 0 | 0 | 0 | 0 | 0 | 0 | msb | $ITA2_{j+1}$ |

Figure F-6 : Two-Into-Two Encapsulation Pair

F.3.4.3.2.2 Maintaining Character-Count Integrity with DPI2E

The 'Dense-Pack' ITA-2 Encapsulation (DPI2E) algorithm **shall not** add or delete ITA-2 characters from the character stream being transported. This property is denoted character-count integrity.

To maintain character-count integrity, the DPI2E algorithm **shall** depend on the number of characters that remain after all initial characters have been encapsulated as 3-into-2 Encapsulation Pairs.

For a buffer of size B characters encapsulated within a U_PDU field of length L, the DPI2E packing algorithm is defined as follows:

1. **Case $R = (B \bmod 3) = 0$.** In this case, no characters remain after all initial characters are packed in Three-into-Two Encapsulation Pairs:
 - all ITA-2 characters **shall**⁽¹⁾ be densely packed as Three-into-Two Encapsulation Pairs. [i.e.,: Each Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1].
 - The U_PDU length **shall**⁽²⁾ be set to the value $L = (2 * B/3)$.
2. **Case $R = (B \bmod 3) = 1$.** In this case, one character remains after all initial characters are densely packed in Three-into-Two Encapsulation Pairs:
 - the first (B-1) ITA-2 characters **shall**⁽¹⁾ be densely packed as 3-into-2 Encapsulation Pairs. [i.e.: each of the Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1].
 - The last remaining ITA-2 character **shall**⁽²⁾ be loosely packed in the last octet of the U_PDU in accordance with the 'Loose-Pack Encapsulation of ITA-2 characters specification' of section F.3.4.3.1.
 - The U_PDU length **shall**⁽³⁾ be set to the value $L = (2 * ((B-1)/3) + 1)$.
3. **Case $R = (B \bmod 3) = 2$.** In this case, two characters remain after all initial characters are densely packed in Three-into-Two Encapsulation Pairs:
 - the first (B-2) ITA-2 characters **shall**⁽¹⁾ be densely packed as 3-into-2 Encapsulation Pairs. [I.E.: Each Encapsulation Pair will contain three ITA-2 characters, with DP_FLG =1]
 - the two remaining ITA-2 characters **shall**⁽²⁾ be packed as a 2-into-2 Encapsulation Pair. [I.E.: Each Encapsulation Pair will contain two ITA-2 characters, with DP_FLG =0; the bit

positions corresponding to the third ITA-2 character in the Encapsulation Pair will be set to zero.].

- the U_PDU length **shall**⁽³⁾ be set to the value $L = (2 * ((B-2)/3) + 2)$.

F.3.4.3.3 Character Unpacking Requirements for DPI2E

In accordance with standard operation procedure or out-of-band coordination circuit, the receiver **must** be configured to perform Dense-Pack ITA-2 Encapsulation.

The receiving client **shall** perform the inverse DPI2E algorithm to unpack the ITA2 characters the U_PDUs contained within an S_UNIDATA_INDICATION primitive:

For a U_PDU of length L,

- If $L = 1$, the receiving client **shall** unpack the single ITA2 character from the loosely packed octet and send it to the output serial stream;
- If $L > 1$ and L even, the receiving client **shall**⁽¹⁾ unpack the ITA2 characters in order from each Encapsulation Pair of octets, continuing in order for each successive Encapsulation Pair in the U_PDU. Unpacked ITA2 characters **shall**⁽²⁾ be sent in the order in which they are unpacked to the output serial stream. Receiving nodes **may** log a processing error if any Encapsulation Pair except the last has DP_FLG = 0.
- If $L > 1$ and L odd, the receiving client **shall**⁽¹⁾ unpack the ITA2 characters in order from each Encapsulation Pair of octets, continuing in order for each successive Encapsulation Pair in the U_PDU, and unpacking the last ITA2 character from the single octet (last, and not part of an Encapsulation Pair) in the U_PDU. Unpacked ITA2 characters **shall**⁽²⁾ be sent in the order in which they are unpacked to the output serial stream. Receiving nodes **may** log a processing error locally if any Encapsulation Pair has DP_FLG = 0. Receiving nodes **may** log a processing error locally if the three most-significant bits of the octet are nonzero.

F.3.4.4 Encapsulation of 6-bit Character Codes

Characters in 6-bit formats for the COSS client **shall** be aligned with the octets in each U_PDU encapsulated in the S_Primitive, one character per octet.

| | MSB | | Octet Data | | | | | LSB |
|--------------------|-----|---|------------|---|----------------------------------|---|---|-----|
| Octet / Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Octet _i | 0 | 0 | msb | - | 6_bitCha <i>r_j</i> | - | - | lsb |

The least-significant bit (LSB) of each 6-bit character received on the serial interface **shall** be aligned with the LSB of the octet.

The value of the MSB bit of each octet **shall** be set to zero for 6-bit character encapsulation.

The special case of the 64-ary ITA-2 character code defined for STANAG 5030 is specified below.

F.3.4.4.1 Encapsulation of 64-ary ITA-2 (i.e., STANAG 5030) character codes

The 64-ary ITA-2 code as defined in STANAG 5030 for Single and Multichannel VLF/LF Broadcasts actually uses a 7-bit character. It consists of 6-bits for information plus a 7th bit (the Stop Bit) that does not change. This allows the possibility that some applications may choose a simple approach to shortening the STANAG 5030 64-ary ITA-2 code to a true six-bit code (i.e., by deleting the stop bit from the code, relying on other measures to provide character synchronization).

In 7-bit format (i.e., unshortened), the 7.0 unit STANAG 5030 64-ary ITA-2 code **shall** be encapsulated as specified in section F.3.4.2.

As a shortened 6-bit code (i.e., with the 7th/stop-bit removed, the STANAG 5030 64-ary ITA-2 code **shall** be encapsulated as specified in section F.3.4.4.

As the overhead from both approaches is equivalent, there is no reason with respect to STANAG 5066 operation to shorten the code. Consequently, use of the unshortened 7.0 unit STANAG 5030 64-ary ITA-2 code is preferred. Other external considerations may apply however that would favor use of the shortened code.

F.3.4.5 Character-Flush Requirements

It is assumed that the COSS client will be implemented with an input buffer for temporary storage of characters from the stream prior to their encapsulation in S_PRIMITIVES for transmission over the subnetwork. The events that trigger transfer of characters from this buffer to an S_PRIMITIVE (i.e., that triggers a 'character-flush' operation) need to be specified for the client. Of concern are the performance tradeoffs that exist for various character-flush disciplines. For instance, frequent character-flush operations will reduce end-to-end latency and increase the overhead, while for infrequent character-flush operations, triggered only when the size of the input buffer is the subnetwork's Maximum Transmission Unit Size (MTU), the opposite is true. As this is a largely performance issue and not an interoperability issue, a number of different behaviours could be defined.

The COSS client **shall** have a capability to configure the behaviour of its input-buffer character-flush discipline.

Characters **shall** be flushed from the COSS input buffer and encapsulated in an S_PRIMITIVE for transmission over the subnetwork when one or a combination of the following events occur:

1. The number of characters in the input buffer exceeds a configurable threshold value, COSS_BUF_FLUSH_THRESHOLD [NB: if the specified threshold value is greater than the subnetwork MTU size, then COSS_BUF_FLUSH_THRESHOLD shall equal the MTU value.];
2. A carriage-return/line-feed input character-pair is detected in the input character stream. [NB: this behaviour provides line-by-line transmission of a character stream organised as lines of text.]
3. A configurable timeout interval has occurred following the arrival in the input buffer of the last received character. [NB: this behaviour ensures that characters in the input buffer are eventually transmitted if one of the first two events has not occurred.]

Other behaviours for the character-flush disciplines **may** be defined as additional and configurable implementation options, e.g., triggering a character-flush operation on detection of a user-specifiable character sequence defined as an End-Of-Message (EOM) sequence. [NB: such operation would be useful to support legacy systems such as the ACP-127 messaging systems, which use a defined character sequence as a message delimiter.]

F.4 TACTICAL MILITARY MESSAGE HANDLING SYSTEM (T-MMHS) CLIENT - Interfacing STANAG-4406-Annex-E Compliant Systems w/ a STANAG 5066 Subnetwork

A Formal Military Message is different from an interpersonal message in that it is a message sent on behalf of an organization, in the name of that organization, that establishes a legal commitment on the part of that organization under military law, and has been released in accordance with the policies of the originating nation. Examples are military orders. Individuals may send organizational Messages to other individuals on behalf of their respective organizations. Formal Military Messages support the additional services associated with ACP 127, and is therefore interoperable with ACP 127 systems.

Formal Military Messages are handled by Military Message Handling Systems (also called High Grade Messaging Service). An MMHS takes responsibility for the delivery, formal audit, archiving, numbering, release, emission and distribution of received formal messages on behalf of the originating organization. An MMHS is accountable under military law to provide a reliable, survivable and secure messaging service on behalf of the originating organization. The MMHS fulfills the military messaging service requirements of ACP-121 and ACP-127 NATO Supplement and offer high standards of messaging reliability and security. The formal messaging service is seen as the vehicle for secure mission critical, operational military applications.

NATO support for the X.400 protocols in Military Message Handling Systems (MMHS), as defined by NATO STANAG 4406, is mandated in the NATO C3 Technical Architecture (NC3TA), Volume 4 - NATO Common Standards Profile (NCSP). STANAG 4406 Edition 1 Annex E (S'4406E) specifies an adaptation of the X.400/X.500 protocols in STANAG 4406 for Tactical Military Messaging Handling Systems (T-MMHS), with cross-references to STANAG 5066 as one example of a low-bandwidth

bearer service to which it has interfaces. This section summarises and specifies additional requirements for the interface between S'4406E-based T-MMHS and a STANAG 5066-compliant HF subnetwork.

The classes of Tactical Messaging Interfaces (i.e., peer-to-peer interfaces between messaging systems) defined by S'4406E are summarised below and in the accompanying Figure (from S'4406E):

1. TMI-1: the Tactical Messaging Interface between two Light Message Transfer Agents (LMTA) [NB: requirements for LMTAs are defined in S'4406E.]
2. TMI-2: the Tactical Messaging Interface between an LMTA and a Light User Agent (LUA) [NB: requirements for LUA are defined in S'4406E.]
3. TMI-3: the Tactical Messaging Interface between an LUA and a Light Message Store (LMS) [NB: requirements for LMSs are defined in S'4406E.]
4. TMI-4: the Tactical Messaging Interface between two LUAs.
5. TMI-5: the Tactical Messaging Interface between an ACP127 Access Unit (defined in STANAG 4406 Annex D) and a conventional ACP127 Military Messaging System.

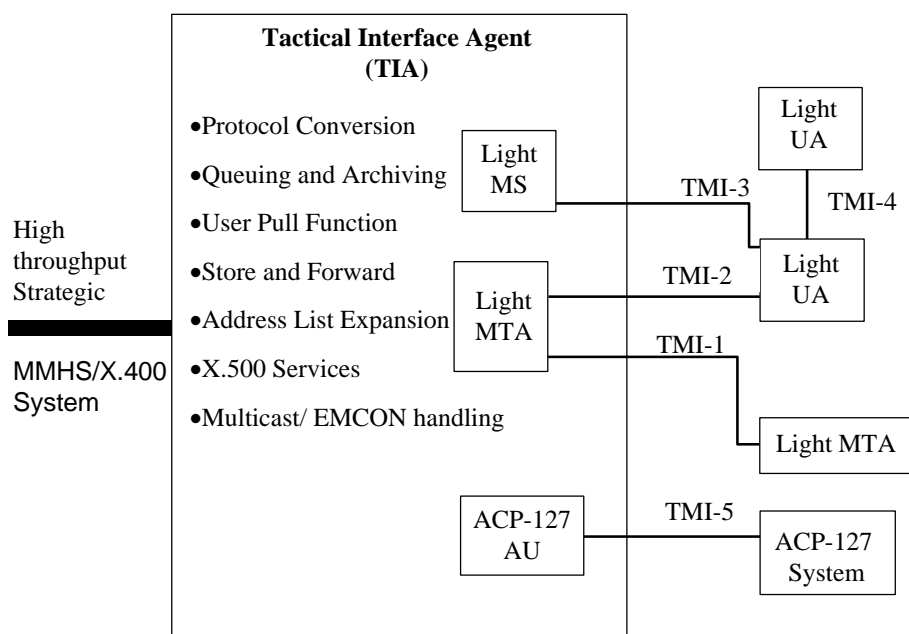


Figure F-7 - Tactical Messaging Interfaces in STANAG 4406 Annex E (from the STANAG).

Base requirements for the Tactical Interface Agent (TIA) and 'Light' modifications to the peer-to-peer protocols and operation of X.400-derived mail-system objects (e.g., the Light Message Transfer Agent) are defined in S'4406E and summarised here only for context in defining the interface to STANAG 5066 subnetworks its requirements. The TIA constitutes an application-level gateway

between the heavyweight protocols of STANAG 4406 used within strategic networks and the lightweight protocols specified for use within low-bandwidth, high-latency tactical subnetworks.

The S'4406E protocol stack incorporates many techniques in the Tactical Adaptation Sublayer to reduce the overhead of the X.400/X.500 protocols. These techniques include the use of performance-enhancing proxies at the edges of a low-bandwidth high-latency communication system, and enforced compression of the message-object to reduce the offered traffic load to the subnetwork. The resulting protocol stack is shown in Figure F-8 and its requirements defined in detail in S'4406E.

A WAP Transport layer within the T-MMHS S'4406E, Appendix A, defines use of two classes of interfaces to a STANAG 5066 subnetwork, depending on whether or not IP network services are required. Further requirements on these interfaces are specified in the subsections below.

Implementations of STANAG 5066 **may** provide a T-MMHS client. If provided, the T-MMHS client **shall** conform to the requirements noted herein and STANAG 4406 Annex E.

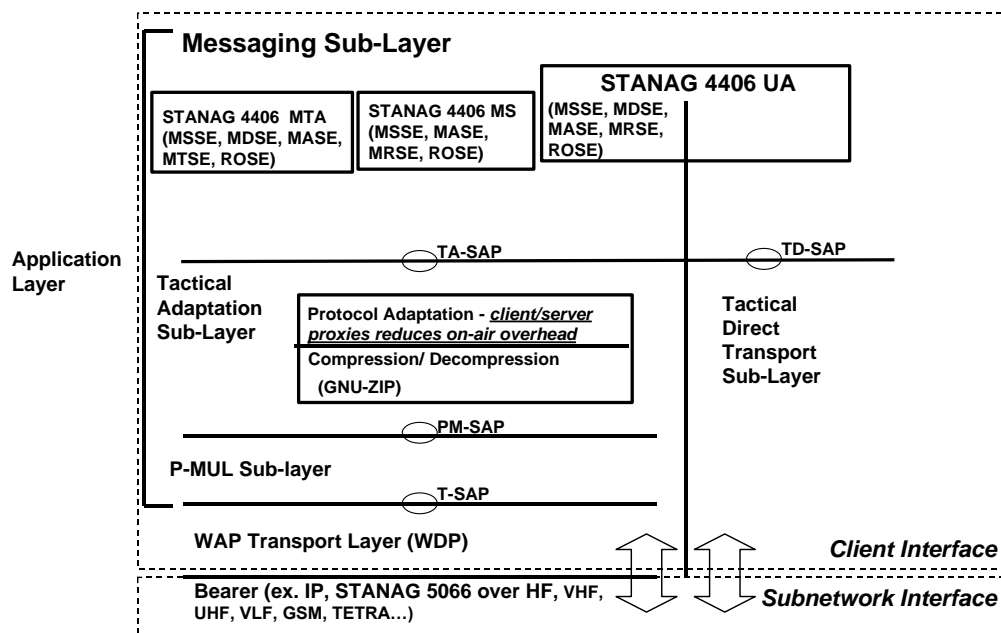


Figure F-8 - STANAG 4406 Annex E - Protocol Stack, Proxy placement, and Interfaces

F.4.1 General Interface Requirements

S'4406 T-MMHS systems **should not** interface to the STANAG 5066 subnetwork via the T-MMHS client's Tactical Direct Transport Sublayer (TDTS). [NB: there are no performance-enhancing proxies used in the TDTS, and the X.400 messaging protocols **should not** be used directly over the STANAG 5066 subnetwork.]

The provisions of this section **shall** apply only to S'4406E complaint T-MMHS clients when using tactical messaging interfaces TMI-1, TMI-2, TMI-3, or TMI-4 (i.e., when using X.400-based messaging and the Tactical Adaptation Sublayer interfaces).

T-MMHS clients using TMI-5, i.e., T-MMHS clients that are interfacing to the STANAG 5066 subnetwork as an ACP-127 complaint messaging system in accordance with STANAG 4406 Annex D (DELTA), **shall** interface as a Character-Oriented Stream Service (COSS) client defined in Section F.3 of this Annex.

When IP-network services are required, S'4406E T-MMHS systems **shall** interface to the STANAG 5066 subnetwork as an IP client, conforming to the requirements of Section F.12 of this Annex.

When IP-network services are not required, S'4406 T-MMHS systems **shall** interface to the STANAG 5066 subnetwork using the reliable-connection-oriented services or unreliable datagram services as defined in Section F.4.2 below.

F.4.2 Subnetwork Service Requirements for STANAG-4406-Annex-E Compliant T-MMHS.

S'4406E-compliant T-MMHS clients **may** interface directly to the STANAG 5066 HF subnetwork as an Extended Client type using either a Reliable-Connection-Oriented Protocol (RCOP) or an Unreliable Datagram Oriented Protocol (UDOP), in accordance with the service and routing requirements of the T-MMHS system.

S'4406E-compliant T-MMHS clients **may** bind to the HF Subnetwork at SAP ID 2, which is the SAP ID reserved for T-MMHS [NB: Note that since it is a compliant RCOP or UDOP client, a T-MMHS client **may also** bind to the subnetwork at SAP IDs 6 or 7, respectively]. Client rank and priority are configuration-dependent and implementation-dependent parameters. T-MMHS clients **should not** bind using a Rank = 15.

The RCOP and UDOP protocol data units for T-MMHS clients **shall** conform to the requirements of Section F.8.1 and F.9.1 (for structure) and Section F.10 (for the assigned Application Identifier(s) (APP_ID) assigned to T-MMHS systems). [NB: For example, T-MMHS clients could use unique APP_IDs to distinguish the type of Tactical Messaging Interface supported over the HF subnetwork. Such use would require registration of the assigned APP_IDs by the T-MMHS peers].

Protocol data units from the T-MMHS Adaptation Sublayer **shall** be directly encapsulated within the Application Data Field of the RCOP or UDOP PDUs.

Use of RDOP and UDOP **may** be mixed within a client attached at SAP ID 2. [NB: note that the additional service markings of the S_PRIMITIVES can distinguish between RDOP and UDOP operations, and separate SAP IDs are not strictly required for the T-MMHS client.].

Further specification of service requirements is left to definitions made within STANAG 4406 Annex E.

F.5 HF MAIL TRANSFER PROTOCOL (HMTF): GUIDELINES FOR THE USE OF INTERNET SMTP OVER STANAG 5066 SUBNETWORKS

NATO support for Internet Standard 10, the Simple Mail Transfer Protocol (SMTP) defined by RFC821, is mandated in the NATO C3 Technical Architecture (NC3TA), Volume 4 - NATO Common Standards Profile (NCSP). As noted in Internet Standard 60 for SMTP Service Extensions for Command Pipelining (defined in RFC 2920), SMTP's basic one-command, one-response model will introduce large delays when used over networks that have high latency. This section defines minimal requirements for an HF Mail Transfer Protocol (HMTF), based on the Internet Simple Mail Transfer Protocol (SMTP) with enforced Command Pipelining that operates efficiently and directly over a STANAG-5066-compliant HF transport service. While SMTP is most often used on TCP/IP networks, the SMTP standard notes in its Annexes to RFC 821 that use over other transport protocols is possible. Direct use of SMTP without the use of the TCP/IP protocols over the HF subnetwork reduces overhead and is suitable in a variety of scenarios; the HMTF client enforces the use of some optional provisions of the SMTP extensions to reduce overhead and delay even further when compared to the basic SMTP protocol. HMTF would be suitable, for example, to provide a communications service for SMTP E-mail over HF radio, allowing a client (i.e, user-agent) to submit mail to a server (i.e, mail transfer agent). Alternatively, HMTF is suitable as a mail-transfer service between two SMTP mail-systems connected by a S'5066 HF subnetwork.

Implementations of STANAG 5066 **may** provide an HMTF client. If provided, the HMTF client **shall** conform to the requirements defined herein.

F.5.1 General Requirements

The HMTF protocol **may** be used by a mail client to submit a mail-object to a mail server using the HF transport profile defined in Annexes A-D of STANAG 5066. Alternatively, HMTF **may** be used as the protocol to transfer a mail object over the HF subnetwork from one mail server to another. In this latter case, the mail server that initiates the transfer assumes the role of a mail client in the HMTF protocol.

The HMTF protocol **shall not** be used to send and receive Formal or High Grade Military Messages (i.a. military orders). For Formal or High Grade Military Messaging, STANAG 4406 Annex E **shall** be used (see section F.4). The HMTF protocol **may** be used for informal interpersonal e-mail only.

An HMTF client or server **shall** use the HF subnetwork protocol stack directly, without intervening transport or network protocol, by encapsulating the SMTP commands, replies, and mail objects within the S_Primitives defined in STANAG 5066 Annex A.

[NB: SMTP mail systems that incorporate TCP/IP protocols in their profile **may** interface to the HF subnetwork as a ETHER client (section F.11) client or as an IP client (section F.12) rather than as an HMTF client as described here. For efficiency in use of the HF subnetwork, SMTP mail systems that include TCP/IP in their transport protocol profile **should** follow the recommendations of RFC2821 and

extended herein regarding use of the command pipelining and other extended-service capabilities defined for SMTP.]

The HMTTP mail model and mail-objects, including the commands, responses, and semantics of the HMTTP protocol, **shall** be defined by the following standards:

- A) the Internet Standard 10 for the Simple Mail Transfer Protocol [RFC 821],
- B) the proposed consolidated SMTP standard defined in RFC2821 (intended as the replacement for, and with precedence in requirements over, RFC821),
- C) the Internet Standard 60 defining the SMTP Service Extension for Command Pipelining [RFC2920], and
- D) the amendments defined herein that mandate certain SMTP options for efficiency in use over the HF channel.

In particular:

- 1. An HMTTP client and server **shall** implement the minimal required set of SMTP Service Extensions defined in RFC2821.
- 2. An HMTTP server **shall** implement the SMTP PIPELINING Service, in accordance with RFC 2920.
- 3. An HMTTP client **should**⁽¹⁾ enforce use of the Internet Standard 60 for SMTP PIPELINING Extension defined in RFC2920, i.e., an HMTTP client **should**⁽²⁾ submit its commands and mail information immediately following the initial 'EHLO' command without waiting for a reply from the HMTTP server. [NB: this is an amendment of RFC2920, which states that an SMTP client continues "once [it] confirms that support exists for the pipelining extension", i.e., the standard implies that the client waits for a response. There is no effect on the SMTP server, which is prepared to accept the initial EHLO and commands as a block. See the comparison discussion in Section F.5.x]
- 4. HMTTP clients and servers **should** implement the 8-BITMIME service extension, in accordance with the recommendations of RFC2821. [NB: this promotes efficiency in the channel by matching the MIME content-type-encoding for binary data to the binary-transparent channel provided by the STANAG 5066 subnetwork.]
- 5. Mail objects **may** be of any type recognized by the SMTP protocol (e.g., messages in RFC822 format), or with MIME Extensions for SMTP, as defined in RFCs 2045, 2046, 2047, and 2049, and S-MIME. In particular, mail-object data **must** observe the requirements of these standards as to line-length and data-transparency.

F.5.2 Comparison of SMTP, SMTP w/ Command Pipelining, and HMTTP (Informative)

The original SMTP model was based on a one-command/one-response dialog, shown by example in Table F- 4. Each command or response in the dialog is a line of text encoded in ASCII format. The command/response dialog in the example requires fourteen steps to establish the connection. These

steps designate the source and destination addresses for the mail, validate the source and destination addresses, transfer the mail message, acknowledge receipt of the mail, and close the connection. Definition of the commands and responses are from RFC821 and RFC2821.

The SMTP Service Extensions for Pipelining defined in Internet Std 60 (i.e, in RFC2920) group the client-server messages in a way that significantly reduces the number of transactions. The grouping provides the same assurance of reliability in the protocol, albeit delayed until the server completes each response to a grouped set of client requests. A set of transactions for SMTP with Command Pipelining also is presented in Table F.2 , identical in function and result to that for the SMTP/Basic Service, but with fewer transactions required.

In order to increase efficiency and reduce response time over high-latency channels, the enforced command pipelining of HMTTP combines even more of the steps that are separate in the SMTP. Unlike SMTP w/Command-Pipelining which first checks for a valid response that confirms a peers capability to use the pipelined commands, HMTTP proceeds under the assumption that the peer-level process is fully compliant with its pipelined/grouped SMTP commands. This streamlines the process to use the minimum number of transactions between the client and server, as shown in the final column of Table F.2 comparing the three approaches. The disadvantage is that if the peer-level mail process is not compliant with HMTTP, then the transactions are lengthy to no purpose, since the mail will not be transferred correctly but the transmissions could take significant time on the channel before this is determined.

Note that the enforced command pipelining defined here is an alternative to the use of performance-enhancing proxies (PEPs), e.g., such as those defined in the tactical adaptation sublayer of S'4406E, placed at the edges of the HF subnetwork. While the protocol overhead required by SMTP is still transmitted over the HF subnetwork, the command-pipelining requirement to match commands and responses in the dialog between mail client and server reduces end-to-end delays much in the way that PEPs do.

Table F-4- Comparison of Basic SMTP, SMTP w/ Pipelining (per RFC-21920) and HMTTP

| Transaction Number: Source (C/S) | SMTP (basic service, w/o Service Extensions) | SMTP w/ Command Pipelining | HMTTP (SMTP w/ Enforced Command Pipelining) |
|-------------------------------------|--|--|--|
| 1:C | HELO <server_name> | EHLO <server_name> | EHLO <server_name> MAIL FROM: <user@client_name> RCPT TO: <other@destination> RCPT TO: <another@destination> DATA Message blah, blah, blah More blah, blah, blah . QUIT |
| 2:S | 250 <client_name> | 250-<client_name> 250-PIPELINING 250 8-BIT MIME | 250-<client_name> 250-PIPELINING 250 8-BIT MIME 250 sender <user@client_name> OK 250 recipient <other@destination> OK 250 recipient <another@destination> OK 354 enter mail, end with line containing only ". 250 message sent 221 goodbye |
| 3:C | MAIL FROM: <user@client_name> | MAIL FROM: <user@client_name> RCPT TO: <other@destination> RCPT TO: <another@destination> DATA Message blah, blah, blah More blah, blah, blah . | |
| 4:S | 250 sender <user@client_name> OK | 250 sender <user@client_name> OK 250 recipient <other@destination> OK 250 recipient <another@destination> OK 354 enter mail, end with line containing only ". 250 message sent 221 goodbye | |
| 5:C | RCPT TO: <other@destination> | QUIT | |
| 6:S | 250 OK | 250 message sent 221 goodbye | |
| 7:C | RCPT TO: <another@destination> | | |
| 8:S | 250 OK | | |
| 9:C | DATA | | |
| 10:S | 354 <enter full mail text, ending with a line that contains only a ".> | | |
| 11:C | Message blah, blah, blah More blah, blah, blah . | | |
| 12:S | 250 message sent | | |
| 13:C | QUIT | | |
| 14:S | 221 goodbye | | |

F.5.3 Subnetwork Service Requirements for HMTF

Clients and Servers for the HF Mail Transfer Protocol **shall** bind to the HF Subnetwork at SAP ID 3. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. HMTF clients **should not** bind using a Rank = 15.

The commands, replies, and mail-object data associated with the HF Mail Transfer Protocol **shall** be submitted to the HF subnetwork using the S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

The address in the primitive will be an individual node address corresponding appropriately to the HF subnetwork address of the host on which the HMTF client (or server) is collocated.

The encoded data for commands, replies, and mail-object data in HMTF **shall** be bit- and byte-aligned with the octets in an S_Primitive's U_PDU, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

Message data encoded as seven-bit symbols (e.g., ITA5 or the ASCII character sets) **shall**⁽¹⁾ be bit-aligned, LSB to LSB, with the octets of the S_Primitive. The unused eighth (i.e, MSB) of the octet **shall**⁽²⁾ be set to zero in compliance with RFC2821.

F.5.4 SMTP Support over Simplex Channels (w/ STANAG 5066Non-ARQ Service)

For use over simplex channels with non-ARQ service modes in STANAG 5066, e.g., in an HF broadcast environment, SMTP support **may** be provided through the use of the Batch-SMTP Media Type defined in Internet RFC2442, using a UDOP client (defined in Section F.9) and suitable Application Identifier (defined in Section F.10).

RFC2442 defines "a MIME content type suitable for tunnelling an ESMTP [RFC-821, RFC-1869] transaction through any MIME-capable transport." [NB: the quoted statement predates RFC2821, which encompasses all of the requirements for Extended SMTP defined in RFC1869.] The Batch-SMTP type allows grouping of SMTP-commands and mail objects within an "application/batch-SMTP" MIME type, and subsequent transmission over a simplex channel. The only requirements on the simplex channel are that it supports MIME data transfer. Consequently, it is a standards-based approach for broadcasting SMTP messages over a STANAG 5066 subnetwork.

F.6 POP3 FOR USE OVER STANAG 5066 TRANSPORT (HFPOP)

The Internetwork Standard 53, the Post-Office Protocol-Version 3 (POP3) defined in RFC1939, is a mandatory standard within the NATO Common Standards Profile (NCSP) of the NATO C3 Technical Architecture. While SMTP is used by a mail client to submit mail-objects to a server, POP3 is used to retrieve mail-objects from a server. This section provides minimal requirements for the efficient and direct support of the POP3 protocol using an HF subnetwork compliant with the requirements of STANAG 5066 Annexes A-D. The profile specified herein requires that certain optional features of the POP3 protocol be implemented for efficiency in operation over the HF channel. The resulting protocol is called the HF Post Office Protocol (HFPOP).

Implementations of STANAG 5066 **may** provide an HFPOP client. If provided, the HFPOP client **shall** conform to the requirements defined herein.

F.6.1 General Requirements

The HFPOP protocol **may** be used by a mail client to retrieve a mail object from a mail server using the transport profile defined in Annexes A-D of STANAG 5066.

The HFPOP protocol **shall not** be used for Formal or High Grade Military Messaging (i.a. military orders). For Formal or High Grade Military Messaging, STANAG 4406 Annex E **shall** be used (see section F.4). The HFPOP protocol **may** be used for informal interpersonal e-mail only.

An HFPOP client and server **shall** use the HF protocol stack directly, without intervening transport or network protocol, by encapsulating the POP3 commands, replies, and mail objects within the S_Primitives defined in STANAG 5066 Annex A.

[NB: POP3 mail clients/servers that incorporate TCP/IP protocols in their profile **may** interface to the HF subnetwork as a ETHER client (section F.11) client or as an IP client (section F.12) rather than as an HFPOP client as described here. POP3 mail clients and servers that include TCP/IP in their transport protocol profile **should** follow the recommendations of RFC2449 regarding use of the overlapping commands and other extended-service capabilities defined for POP3.]

The HFPOP mail model and mail-objects, including the commands, responses, and semantics of the HFPOP protocol, **shall** be defined by the following:

- A) the Internet Standard 53 for the Post Office Protocol Version 3 [defined in RFC1939],
- B) the Internet Standards-Track Service-Extensions model for the POP3 protocol defined in RFC2449, and
- C) the amendments defined herein which mandate certain optional features of RFC1939 and RFC2449 for efficiency in use over the HF channel.

In particular:

1. An HFPOP client and server **should** implement the set of POP3 Extension Mechanisms defined in RFC2449.

2. An HFPOP server **shall** implement the POP3 PIPELINING Extension defined in RFC2449.
3. HFPOP clients and servers **should** implement the 8-BITMIME service extension, in accordance with the recommendations of RFC2821.

F.6.2 HFPOP Subnetwork Service Requirements

Clients and Servers for the HF Post Office Protocol **shall** bind to the HF Subnetwork at SAP ID 4. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. HFPOP clients **should not** bind using a Rank = 15.

The commands, replies, and mail-object data associated with the HF Post Office Protocol **shall** be submitted to the HF subnetwork using S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

The address in the primitive will be an individual node address corresponding appropriately to the HF subnetwork address of the host on which the HFPOP client or server is located.

The encoded data for commands, replies, and mail-object data in HFPOP **shall** be bit- and byte-aligned with the octets in an S_Primitive's U_PDU, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

Message data encoded as seven-bit symbols (e.g., ITA5 or the ASCII character sets) **shall**⁽¹⁾ be bit-aligned, LSB to LSB, with the octets of the S_Primitive. The unused eighth (i.e., MSB) of the octet **shall**⁽²⁾ be set to zero.

F.7 OPERATOR ORDERWIRE CLIENT (HFCHAT)

A STANAG 5066 implementation **may** provide an Operator Orderwire Client for short text-message exchange (referred to as HFCHAT) between HF subnetwork operators. If provided, the HF CHAT client **shall** conform to the minimal requirements defined herein. The HFCHAT client is intended as a simple orderwire to allow subnetwork operators to test and coordinate their system configurations.

F.7.1 General Requirements for HFCHAT

HFCHAT clients **shall** use the ITA5 / ASCII character set to exchange short orderwire messages between subnetwork operators.

Orderwire messages **shall** consist of sequences of characters terminated by a carriage-return/line-feed pair (i.e., terminated by the octet-pair 0x0D, 0x0A).

The orderwire messages length, i.e., the number of octets in the character sequence and including terminating carriage-return/line-feed pair, **shall** not exceed the subnetwork MTU size.

In general, methods of presentation and display to the operator of orderwire messages, as well as methods for orderwire-message entry, are beyond the scope of this STANAG and left as implementation options. The following implementation guidelines are recommended, however:

- HFCHAT clients **should** provide a common entry and display area for orderwire messages.
- HFCHAT clients **should** provide a short, viewable history of previous messages sent and received (e.g., of the last N messages, N a value (configurable or not) in the range [10,100]).
- HFCHAT clients **should** provide an indication of the source of any orderwire messages that it receives (e.g., by displaying the STANAG 5066 address of the originator of the orderwire message)
- HFCHAT clients **should** provide an indication of the time-of-receipt of any orderwire messages that it receives.
- HFCHAT clients **should** provide confirmation-of-delivery indications for orderwire messages it sends when they are sent using ARQ delivery service.
- HFCHAT clients **should** provide additional status indications of the state of its interface to the subnetwork, including summary display of S_PRIMITIVES as they are received from the subnetwork that would indicate the health of the interface; these include but are not limited to S_BIND_ACCEPT, S_BIND_REJECT S_UNIDATA_REQUEST_CONFIRM, S_UNIDATA_REQUEST_CONFIRM, etc. Key parameters (e.g., reject reasons) from these messages **should** also be displayed.

F.7.2 Subnetwork Service Requirements

An HFCHAT client **shall** bind to the HF Subnetwork at SAP ID 5. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. HFCHAT clients **may** bind using a Rank = 15.

HFCHAT clients **may** use either point-to-point or point-to-multi-point addressing modes to send orderwire messages.

The default subnetwork-service requirements when using the point-to-point addressing mode **shall** be as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

For point-to-point operation, the address in the primitive will be an individual node address corresponding appropriately to the HF subnetwork address of the remote HFCHAT client to which the orderwire message is addressed.

The default subnetwork-service requirements when using the point-to-multipoint addressing mode **shall** be as follows:

- Transmission Mode = non-ARQ; number of repeats (configurable)
- Delivery Confirmation = NONE
- Deliver in Order = IN-ORDER DELIVERY

For point-to-multipoint operation, the address in the primitive will be a multicast-address corresponding to the group of remote HFCHAT clients to which the orderwire message is addressed. Establishment of the multicast address for the group may be done through standard operating procedure or out-of-band channel.

The orderwire-message's sequence of characters **shall**⁽¹⁾ be bit- and byte-aligned with the octets in an S_Primitive's U_PDU, with the least-significant bit (LSB) of each character aligned with the LSB of the octet. The unused eighth (i.e, MSB) of the octet **shall**⁽²⁾ be set to zero.

Orderwire messages **shall** be encapsulated and sent within S_PRIMITIVES, one message per S_PRIMITIVE, with the message-terminating carriage-return/line-feed pair encapsulated in the S_PRIMITIVE as the last two octets of the U_PDU field.

F.8 RELIABLE CONNECTION-ORIENTED PROTOCOL

This subsection specifies a simple Reliable Connection-Oriented Protocol (RCOP) for reliable data connections between applications using the ARQ services of the HF subnetwork. RCOP provides a minimal header to support multiplexed connections between a pair of nodes through a single hard or soft link. Applications using an RCOP connection are identified uniquely by a field in the header of the RCOP PDU.

F.8.1 RCOP Protocol Data Unit (RCOP_PDU)

The format of all RCOP U_PDUs **shall** be as shown in the Figure below.

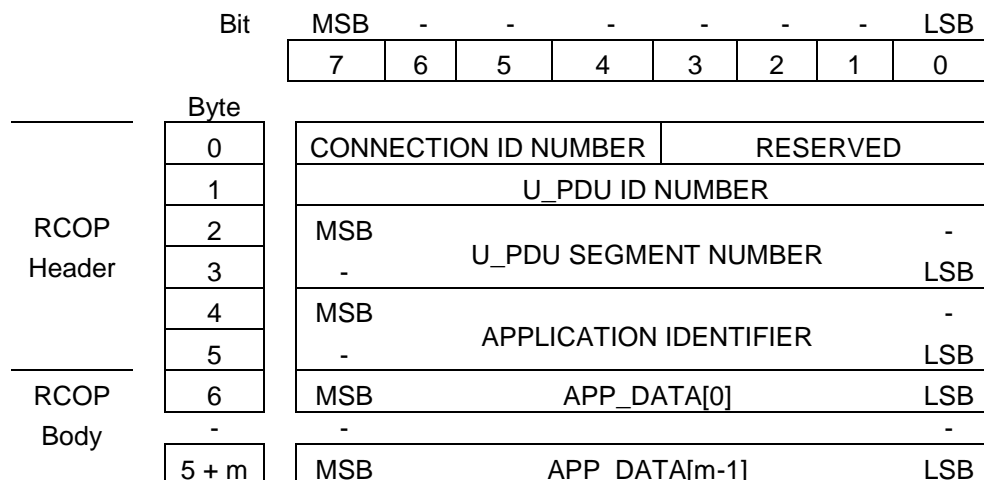


Figure F-9. Format for Reliable Connection-Oriented Protocol Data Units

The following are required for RCOP PDUs:

1. RCOP clients may provide multiplexed transport service for more than one application simultaneously by establishing multiple connections, each identified by its CONNECTION_ID_NUMBER field. The CONNECTION_ID_NUMBER field **shall** be a value from 0-15.
2. New values for the CONNECTION_ID_NUMBER field **shall** be dynamically assigned as new connections are established. Further details regarding assignment and co-ordination of connection ID numbers are not specified here.
3. Connection ID number 0 **shall** be reserved for non-multiplexed connections or the first multiplexed connection.
4. The reserved bits **shall** be set to 0.
5. U_PDU ID numbers **shall** be assigned consecutively to user PDUs (U_PDUs) serviced by the connection.
6. The U_PDU segment number **shall**⁽¹⁾ be assigned consecutively to segments within a single U_PDU. The first segment transmitted **shall**⁽²⁾ be assigned segment number 0. If a U_PDU is not segmented, the single segment that is transmitted **shall**⁽³⁾ be assigned number 0.
7. The APPLICATION_IDENTIFIER **shall** be assigned in accordance with the requirements of section F.10 of this Annex. This field serves to identify the application (i.e., higher-level

protocol) using the connection. End-to-end interoperability can be achieved only if this is a unique value for any given end-user application, otherwise there is the (likely) chance that the applications connected by the RCOP client and subnetwork will be incompatible.

8. The APP_DATA[] field **shall** contain the m-bytes of application data sent over the connection. Segmentation and reassembly rules for mapping the application's data into the APP_DATA[] field are in general outside of the scope of this STANAG, and are application dependent.

F.8.2 RCOP Subnetwork Service Requirements

RCOP clients **shall** bind to the HF Subnetwork at SAP ID 6. Client rank and priority are configuration-dependent and implementation-dependent parameters. RCOP clients **should not** bind using a Rank = 15.

Each RCOP_PDU sent over the subnetwork **shall** be embedded in an S_UNIDATA_REQUEST primitive, each byte of the RCOP_PDU corresponding to a byte in the U_PDU field of the S_Primitive (see Figure). RCOP_PDUs will be encapsulated in like manner when delivered by the subnetwork to the destination client in an S_UNIDATA_INDICATION.

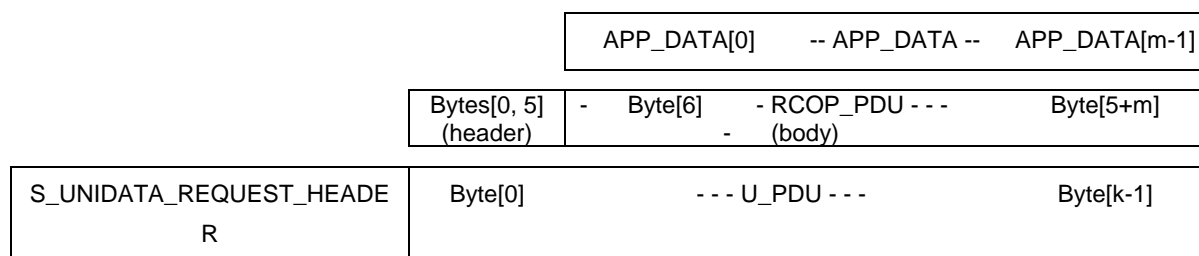


Figure F-10 - Mapping of RCOP_PDU into an S_UNIDATA_REQUEST Primitive for transmission.

The encoded data for the RCOP client **shall** be bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

Note that, in accordance with the provisions of Annex A of this STANAG, if the subnetwork interface sublayer receives a S_UNIDATA_REQUEST primitive with an RCOP Protocol Data Unit larger than the maximum MTU size, the S_UNIDATA_REQUEST will be rejected.

An RCOP client **shall** set the default service requirements for S_UNIDATA_REQUEST primitives as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY or AS_THEY_ARRIVE

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host at which the destination RCOP client is located.

F.8.3 RCOP Segmentation and Reassembly Requirements

. Note that, in accordance with the provisions of Annex A of this STANAG, if the subnetwork interface sublayer receives a S_UNIDATA_REQUEST primitive with an RCOP Protocol Data Unit larger than the maximum MTU size, the S_UNIDATA_REQUEST will be rejected. RCOP clients **must** segment their data and place it into the APP_DATA[] field of the PDU accordingly.

If IN_ORDER delivery-order is specified, an RCOP client **shall** simply take the U_PDU segments in the sequence in which they arrive to construct a copy of the original U_PDU sent by the source. If AS_THEY_ARRIVE delivery-order is specified, an RCOP client **shall** be responsible for reassembling the U_PDU segments it receives in proper sequence order.

RCOP clients **may** devise any algorithm of their own choice for segmentation and reassembly, but the RCOP_PDU fields for CONNECTION_ID_NUMBER, U_PDU_ID NUMBER, U_PDU SEGMENT NUMBER are available for such use. Segmentation and reassembly algorithms **shall** not use the APPLICATION IDENTIFIER field.

Note that, in general, a local RCOP client could be receiving data on two different connections, each established by another remote RCOP client as the remote client's sole connection. In this case, the remote clients would each have specified a connection ID number of zero. As an alternate but similar scenario, two RCOP clients on the same remote node but attached to different SAP IDs could connect to the local RCOP client. In this case also, data could be received with the same connection ID number. Other scenarios in which the same connection ID has been assigned to data received by an RCOP client might also occur, as dynamic connections are made and broken with different nodes. Thus, to reassemble segmented application data without ambiguity, an RCOP client **must** distinguish U_PDU segments for its receive connections using the unique combination of (SOURCE_ADDRESS, SOURCE_SAP_ID, CONNECTION_ID_NUMBER). The SOURCE_ADDRESS is the address of the originator of the RCOP_PDU, and the SOURCE_SAPID is the SAP_ID to which the remote RCOP client is attached. All three parameters can be obtained unambiguously from the S_UNIDATA_INDICATION primitive in which the RCOP_PDU is delivered to the client, as can the CONNECTION_ID_NUMBER, U_PDU_ID NUMBER, U_PDU SEGMENT NUMBER.

F.9 UNRELIABLE DATAGRAM-ORIENTED PROTOCOL (UDOP)

This section defines a simple Unreliable Datagram-Oriented Protocol (UDOP) using the non-ARQ services of the HF subnetwork, with a minimal header to support multiplexed datagram delivery. Since non-ARQ services are used, the UDOP may support a multicast service through the use of

group addresses within the HF Subnetwork. Applications using a UDOP non-ARQ delivery service are identified uniquely by a field in the header of the UDOP PDU.

F.9.1 UDOP Data Unit

UDOP Protocol Data Units **shall** be defined and used identically to those defined for the Reliable Connection-Oriented Protocol in section F.8.1.

F.9.2 UDOP Subnetwork Service Requirements

UDOP clients **shall** bind to the HF Subnetwork at SAP ID 7. Client rank and priority are configuration-dependent and implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

A UDOP client **shall** submit its U_PDUs to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = non-ARQ
- Delivery Confirmation = none
- Deliver in Order = IN-ORDER DELIVERY or AS_THEY_ARRIVE

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host on which the destination message stores are located. If AS_THEY_ARRIVE delivery-order is specified, the UDOP client at the destination **may** be responsible for reassembling the U_PDU segments in proper sequence order.

The encoded data for the UDOP client **shall** be bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

F.10 EXTENDED CLIENT DEFINITION USING RCOP/UDOP APPLICATION IDENTIFIERS

The APPLICATION_IDENTIFIER field **may** be used in the definition of so-called 'Extended Clients', all of which use the RCOP or UDOP protocol as their base client type, but which are uniquely distinguishable without prior system configuration or end-to-end coordination.

This section lists assigned values for the APPLICATION_IDENTIFIER field and concludes with discussion of some known examples of Extended Client Applications.

F.10.1 Assigned Values

With the 16-bit APPLICATION_IDENTIFIER field, RCOP/UDOP clients can uniquely identify a large number of applications or other upper layer protocols. [NB: There is no assumption that the HF subnetwork can adequately support even a very small fraction of 2^{16} clients simultaneously!]

A portion of the 16-bit range will be reserved and managed by NATO. The remainder will be available for arbitrary vendor use.

APPLICATION_IDENTIFIER field values **shall** be made in accordance with the Table shown below.

Table F-5 – Application Identifier (APP_ID) Assignments

| Identifier Value | Application or Upper-Layer Protocol | Comment |
|------------------|--|--|
| 0x0000 – 0x7FFF | <u>Various</u> | Reserved for NATO Administration: within this block, the following APP_IDs are currently assigned. |
| 0x1002 | Basic File Transfer Protocol (BFTP) File Transfer Service | Backward compatibility assignment with the initial two-byte value specified in the BFTP client of STANAG 5066 Annex F, Edition 1 |
| 0x100B | File-Receipt/Acknowledgement Protocol | Backward compatibility assignment with the initial two-byte value specified in the BFTP client of STANAG 5066 Annex F, Edition 1 |
| 0x100C | File-Receipt/Acknowledgement Protocol Version 2 | Provides acknowledgement of a given file, supporting pipelined BFTP operation sending multiple files |
| 0x2000 | STANAG 4406 Annex E complaint Tactical Military Message Handling Systems (i.e, T-MMHS Clients per F.4) | Base APP_ID assignment for TMI-1 (in support of the LMTA-to-LMTA interface). |
| 0x2001 | STANAG 4406 Annex E | Base APP_ID assignment for TMI-2 (in support of the LMTA-to-LUA interface). |
| 0x2002 | STANAG 4406 Annex E | Base APP_ID assignment for TMI-3 (in support of the LMS-to-LUA interface). |
| 0x2003 | STANAG 4406 Annex E | Base APP_ID assignment for TMI-4 (in support of the LUA-to-LUA interface). |
| 0x2004 | STANAG 4406 Annex E ACP-127 Access Unit | Base APP_ID assignment for TMI-5 (in support of the ACP-127 AU interface). |
| 0x8000 – 0xFFFF | | Available for User-defined applications; uniqueness of Application ID values in this range cannot be guaranteed. |

End-to-end interoperability requires that the value assigned to the APPLICATION_IDENTIFIER field be unique for any given application type. Otherwise, there is the likely chance that two applications connected by the RCOP/UDOP client and HF subnetwork will be incompatible. Vendors

developing applications for use with the STANAG 5066 HF Subnetwork **should** register their application and obtain an application identifier in the reserved range administered by NATO.

F.10.2 Extended-Client Definition using Application Identifiers: Examples

This section defines the use of an RCOP (or UDOP) client as the basis for additional client definitions. An "Extended Client" is a client of the STANAG 5066 HF subnetwork that uses RCOP (or UDOP) as its basic end-to-end transport protocol. Protocols for several example applications are discussed in the sections below that use the extended client definition as their basis. Implementations of STANAG 5066 **may** provide any of these extended clients. If provided, the client **shall** be implemented in accordance with the requirements defined herein.

The extended clients presented here are the following:

1. T-MMHS Clients - the T-MMHS Clients defined in Section F.4 are, by definition, Extended Clients since they are based on RCOP and UDOP, although they have a uniquely assigned SAP ID as well. As noted in Section F.4, implementations of T-MMHS Clients will be governed by the requirements of STANAG 4406 Edition 1, Annex E Version 1.0, with the additional provisions on the client-subnetwork interface defined herein.
2. Basic File Transfer Protocol (BFTP) - BFTP **may** be used for end-to-end file transport over the STANAG 5066 subnetwork profile (i.e., the protocols defined in Annexes A through D). Implementations of STANAG 5066 **may** provide a BFTP client. If provided, the BFTP client **shall** conform to the requirements of Section F.10.2.2 below. BFTP implements a very simple file transfer protocol based on the ZMODEM protocol. [NB: BFTP was the first client for which end-to-end interoperability was demonstrated over different vendor implementations over of the STANAG 5066 subnetwork profile. Historically, it has served as a simple benchmark of the end-to-end interoperability of the STANAG 5066 subnetwork profile (Annexes A-D) and is retained in this Amendment as such.]
3. File-Receipt Acknowledgment Protocol (FRAP) - FRAP **may** be used to provide an acknowledgement of file receipt for files sent over the STANAG 5066 subnetwork using some other protocol, such as BFTP or CFTP (see below). If provided, the FRAP extended client **shall** conform to the requirements Section F.10.2.3 below. [NB: FRAP was provided with BFTP in the earliest implementations of STANAG 5066 and used for interoperability verification of the profile.]
4. File-Receipt Acknowledgment Protocol-Version 2 (FRAPv2) - FRAPv2 **may** be used as an alternative to FRAP. FRAPv2 provides support for pipelined or overlapping file transfer by providing an unambiguous indication of which file has been acknowledged. If provided, the FRAPv2 extended client **shall** conform to the requirements Section F.10.2.4 below.

F.10.2.1 Tactical Military Message Handling System (T-MMHS) Clients

The T-MMHS Client has been defined in Section F.4 as an Extended Client to allow provision for uniquely distinguishing the type of Tactical Messaging Interface (TMI) supported, i.e., to distinguish the traffic flows between LMTAs, LUAs, or LMSs being supported by the HF subnetwork. As noted in Section F.4, T-MMHS clients may also attach to the subnetwork at SAP ID 6 (when using RDOP) or SAP ID 7 (when using UDOP).

For further definition and discussion of the T-MMHS Client requirements, refer to Section F.4 of this Annex and to STANAG 4406 Edition 1 Annex E Version 1.0.

F.10.2.2 BASIC FILE TRANSFER PROTOCOL (BFTP)

The Basic File Transfer Protocol (BFTP) **may** be used to transfer files via a STANAG 5066 subnetwork using an RCOP or UDOP client. It is a very simple protocol (base on the ZMODEM protocol) and is not designed to be especially robust.

F.10.2.2.1 BFTP_PDU Specification

The format for the basic-file-transfer-protocol data unit (BFTP_PDU) **shall** be in accordance with the following Figure, which defines a header part and a file-data part for the BFTP_PDU.

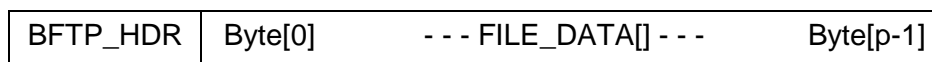


Figure F-11: Basic FTP Protocol Data Unit (BFTP_PDU)

The detailed structure of the BFTP_PDU shall be in accordance with the following Figure, and provide the following information fields:

1. BFTP PDU Header Part:
 - SIZE_OF_FILENAME - one octet in size.
 - FILE_NAME - a variable length field, equal in size to the value specified by the SIZE_OF_FILENAME field.
 - SIZE_OF_FILE - a four-octet field.
2. BFTP PDU Body Part:
 - FILE_DATA[] - a variable length field, equal in size to the value specified by the SIZE_OF_FILE field.

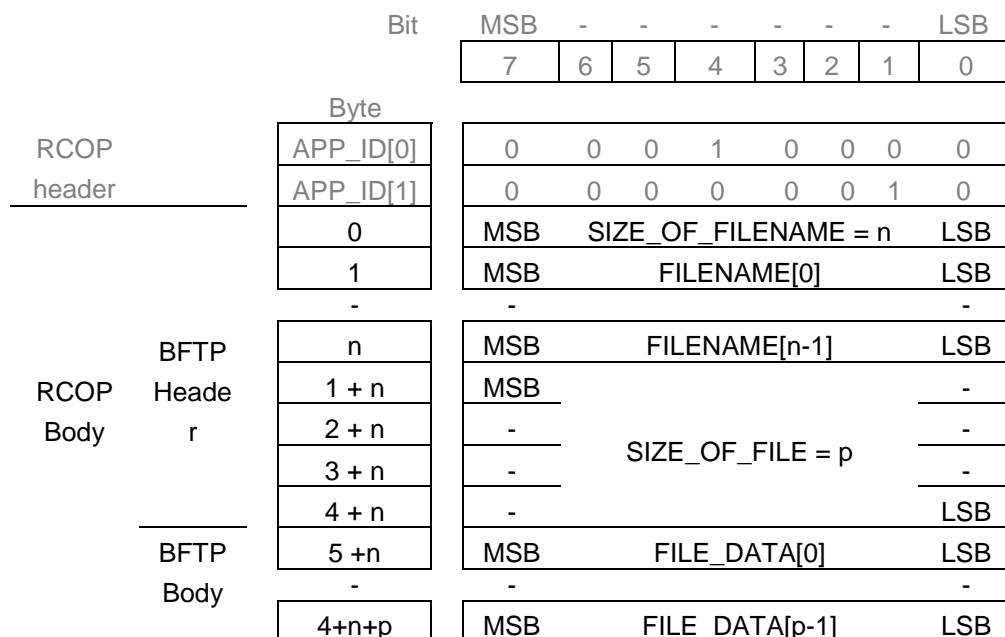


Figure F-12: BFTP Protocol Data Unit Structure

For backwards compatibility with the BFTP specification in earlier versions of STANAG 5066 Annex F Edition 1, the Application Identifier (i.e., the last two bytes of the RCOP header) for BFTP **shall** be 0x1002. [NB: The two-bytes in the Application Identifier correspond in value and position to the first two bytes (i.e., the synchronization bytes) of the BFTP header as defined Annex F Edition 1. They represent the control bytes DLE (Data Link Escape) and STX (Start of Text), which started the Z-modem protocol.]

The SIZE_OF_FILENAME field **shall**⁽¹⁾ be a 1-octet fixed-length field whose value (n) **shall**⁽²⁾ equal the number of octets used to encode the FILENAME field.

The FILENAME field is **shall**⁽¹⁾ be a variable-length field, the size of which **shall**⁽²⁾ be specified by the value (n) of the field SIZE_OF_FILENAME. This field represents the name of the file sent using the Basic File Transfer Protocol. The first byte of the filename **shall**⁽³⁾ be placed in the first byte of this field, with the remaining bytes placed in order. The semantics of file names and naming conventions are beyond the scope of this STANAG.

The SIZE_OF_FILE field **shall**⁽¹⁾ be a 4-octet fixed-length field whose value **shall**⁽²⁾ specify the size (p) in octets of the file to be sent. The first octet of the SIZE_OF_FILE field **shall**⁽³⁾ be the highest order byte and the last byte the lowest order byte of the field's binary value.

F.10.2.2.2 BFTP Segmentation and Reassembly Requirements

If the BFTP_PDU exceeds the maximum size of the data field permitted in the RCOP_PDU (i.e, if the BFTP_PDU is larger than the MTU_size less 6 octets (i.e., MTU-6)), the BFTP client **shall**

segment the BFTP_PDU, placing successive segments in RCOP_PDUs with consecutive U_PDU sequence numbers.

When received, the BFTP client **shall**⁽¹⁾ reassemble the BFTP_PDU if it determines that the BFTP_PDU has been segmented. Subject to local-host file naming conventions, the BFTP client **shall**⁽²⁾ store the received file with the name transmitted in the header with the file. *[NB: there is no guarantee therefore that the file will be stored on the destination host with the same name that it was sent.]*

F.10.2.3 File-Receipt Acknowledgement Protocol (FRAP)

Files sent over STANAG 5066 using a BFTP extended client **may** be acknowledged using the File-Receipt Acknowledgement Protocol (FRAP). *[NB: This protocol was used to acknowledge receipt of files sent using BFTP in earlier implementations of STANAG 5066. Therefore, this protocol has relevance to the earliest proofs of interoperability between different vendor implementations of STANAG 5066.]*

Implementations of STANAG 5066 **may** provide support for the FRAP protocol if they also provide a BFTP (see Section F.10.2.2) or CTFP client (see Section F.14).

Other Extended Client types **may** use FRAP.

To acknowledge receipt of a file using FRAP, on receiving the last byte of a file, the receiving client **shall** send an RCOP_PDU Header Part with an Application Identifier field value = 0x100B; the RCOP_PDU Body Part is null. This serves to confirm that the entire file has been received. This is equivalent to the "ZEOF" message of the Z-modem protocol.

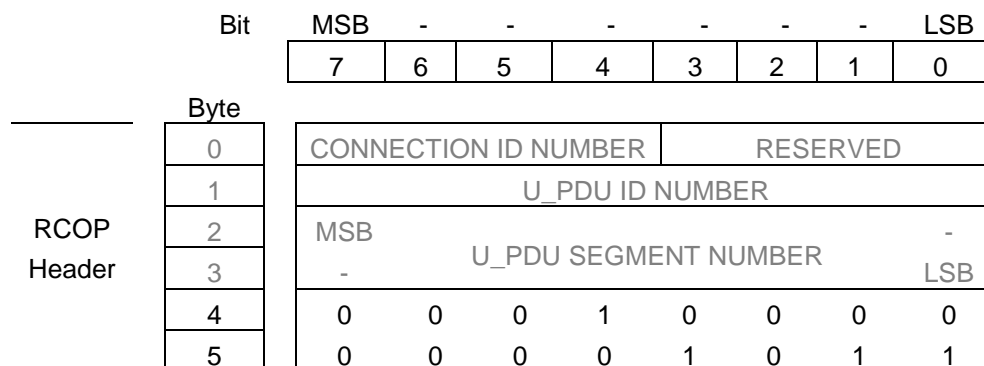


Figure F-13: Application Identifier for the File-Receipt Acknowledgement Protocol (the full RCOP PDU is shown ... the RCOP PDU Body Part is null).

The CONNECTION_ID_NUMBER in the FRAP PDU (i.e., the RCOP_PDU Header part with FRAP APP_ID) **shall**⁽¹⁾ be set to the same value as the CONNECTION_ID_NUMBER of the connection over which the file was received. The U_PDU ID value in the FRAP PDU **shall**⁽²⁾ be set to the same value as the U_PDU ID of the file being acknowledged. This provision corresponds to current practice in some implementations of STANAG 5066 that use the File-Receipt Acknowledgement Protocol.

"Matching U_PDU_ID_NUMBERS" allows use of FRAP under limited conditions to support pipelined file-transfer operations, where a new file transfer can be initiated (with a different U_PDU_ID_NUMBER) before receipt of the acknowledgement for preceding files.

F.10.2.4 File-Receipt Acknowledgement Protocol Version 2 (FRAPv2)

The original FRAP specification provided support for file transfers one at a time, or, in limited scenarios using Matching CONNECTION_ID_NUMBERS. To avoid potential ambiguity in FRAP, receipt of specific file can be acknowledged by name using FRAPv2. FRAPv2 includes a copy of the BFTP header file for the acknowledged file, as shown in the Figure below.

| | | | | | | | | | |
|--|-------|------------------------------|---|---|---|----------|---|---|-----|
| | Bit | MSB | - | - | - | - | - | - | LSB |
| | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | Byte | | | | | | | | |
| RCOP Header | 0 | CONNECTION ID NUMBER | | | | RESERVED | | | |
| | 1 | U_PDU ID NUMBER | | | | | | | |
| | 2 | MSB - | | | | | | | |
| | 3 | - U_PDU SEGMENT NUMBER LSB | | | | | | | |
| | 4 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 5 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| RCOP Body = BFTP Header of the file being acknowledged | 5 + 0 | MSB SIZE_OF_FILENAME = n LSB | | | | | | | |
| | 5 + 1 | MSB FILENAME[0] LSB | | | | | | | |
| | - | - - | | | | | | | |
| | 5 + n | FILENAME[n-1] | | | | | | | |
| | 6 + n | | | | | | | | |
| | 7 + n | | | | | | | | |
| | 8 + n | SIZE_OF_FILE = p | | | | | | | |
| | 9 + n | | | | | | | | |
| | | | | | | | | | |

Figure F-14: File-Receipt Acknowledgement Protocol Version 2 - RCOP PDU and contents

Implementations of STANAG 5066 **may** provide support for the FRAPv2 protocol if they also provide a BFTP (see Section F.10.2.2) or CTFP (see F.14) client.

Other Extended Client types **may** use FRAPv2

The Application Identifier for FRAPv2 shall be 0x100C.

To acknowledge receipt of a file using FRAPv2, on receipt of the last byte of a file the receiving client **shall** send an RCOP_PDU Header Part with an Application Identifier field value = 0x100C and RCOP_PDU Body Part consisting of a copy of the BFTP Header of the file being acknowledged.

F.11 ETHER CLIENT

The Ether client provides multi-protocol support over the HF subnetwork, encapsulating a two-byte Ethertype Field with the protocol data unit. Protocols that may be supported include the following:

- (a) Internet Protocol Version 4 (RFC 791)
- (b) Internet Protocol Version 6 (RFC 2460)
- (c) Compressed IP (e.g., van Jacobsen's TCP/IP header compression, RFC 1144; Robust Header Compression, RFC 3843)
- (d) Address Resolution Protocol (ARP, RFC 826) and its associated variants (Reverse ARP, RFC 903; Inverse ARP, RFC 2390)
- (e) Point-to-Point Protocol (PPP, RFC 1661)
- (f) other network protocol for which an Ethertype field is defined.

F.11.1 General Requirements

Implementations of STANAG 5066 **may** provide an Ether Client.

The requirements of this section **shall** apply to any client that attaches directly to the HF subnetwork interface sublayer to send or receive Etherframes over the HF subnetwork. Such clients include end systems (e.g., web-servers or mail systems) and intermediate systems (e.g., routers).

F.11.2 EC_FRAME Encapsulation

The ether client provides multi-protocol bearer services over the HF subnetwork using an EC_FRAME structure derived from the Ethernet Type II Frame Format by discarding the ethernet addresses and CRC check, leaving only the Ethertype and Data fields as shown in the Figure below. STANAG 5066 Address fields for the HF subnetwork are contained within the S_Primitives within which the EC_FRAME is encapsulated. The EC_FRAME, consisting of a two-byte Ethertype field and the Data Field, are the S_Primitive's User-Protocol Data Unit.

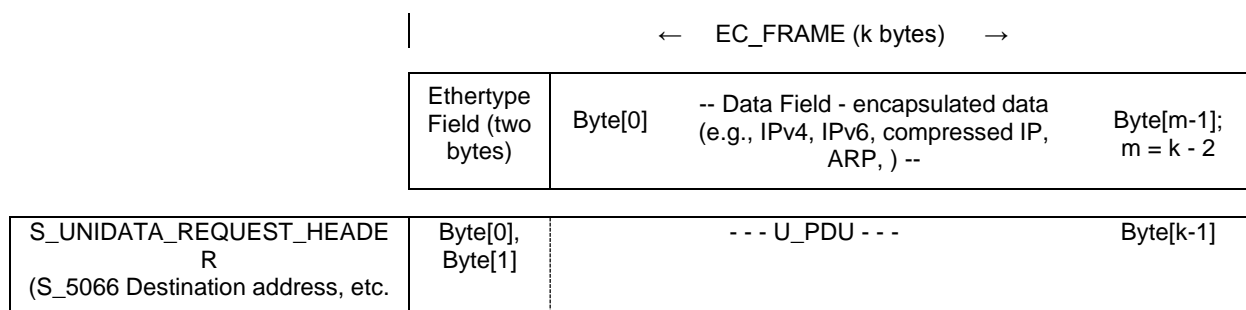


Figure F-15 - Mapping of EC_FRAME into an S_UNIDATA_REQUEST Primitive for transmission.

The EC_FRAME **shall**⁽¹⁾ be submitted to the HF subnetwork encapsulated in an S_UNIDATA_REQUEST Primitive and received from the subnetwork encapsulated in an S_UNIDATA_INDICATION Primitive. These S_Primitives provide all necessary framing information to mark the beginning and end of EC_FRAMES when transported over the HF subnetwork.

The first two bytes of the EC_FRAME (i.e, the Ethertype field) **shall**⁽²⁾ be placed in the first two bytes of the U_PDU field within the primitive, and so on to the last byte of the EC_FRAME and S_Primitive U_PDU field, as shown in the Figure. Encoding of the Ethertype field shall conform to 'network byte order', with the most-significant byte of the Ethertype field the first byte in the U_PDU field and the least-significant byte of the Ethertype field the second byte in the U_PDU field.

The encoded data of the EC_FRAME **shall**⁽³⁾ be bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) of each character aligned with the LSB of the octet.

F.11.3 Ethertype Field Encoding

Encoding of the Ethertype field **shall** conform to IEEE assigned numbers and requirements for this field. Encoding is in network-byte order, i.e., most-significant-byte first. Field values of interest for protocols that **should** typically be supported by the Ether client are given in the Table below.

Table F-6 – Ethertype Field Encoding for some protocols supportable by the Ether Client

| Field Value (hex) | Encapsulated Protocol (i.e., Data Field contents) | Comments |
|----------------------|--|---|
| 0x0800 | IPv4 | IPv4 datagram payload per Section F.11.5.1 |
| 0x86DD | IPv6 | IPv6 datagram payload per Section F.11.5.2 |
| 0x876B | TCP/IP Header Compression | 'Van Jacobsen' header compression per RFC 1144 |
| 0x0180 * | ROHC | Robust Header Compression per RFC 3843 and RFC 3095 |
| 0x0806 | ARP | IP Address Resolution per RFC 826 |
| 0x880B | PPP | Point-to-Point Protocol per RFC 1661 |

* no Ethertype value has been assigned officially for ROHC over Ethernet; this is a value assigned from the 'Experimental' block of Ethertype values.

F.11.4 Ether-Client Subnetwork Service Requirements

Ether clients **shall**⁽¹⁾ bind to the HF Subnetwork at SAP ID 8. Client rank and priority are implementation-dependent parameters for this application. Use of Rank = 15 is discouraged.

The STANAG 5066 address in the S_Primitive will be an node group address corresponding to the HF subnetwork address of the host to which the EC_FRAME is being sent.

HF Subnetwork implementations **shall**⁽²⁾ provide MTU sizes sufficient to support placement of an entire EC_FRAME (up to 2048 bytes) within a single S_Primitive.

All EC_FRAMEs **shall**⁽³⁾ be submitted to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements for transmission mode, delivery-confirmation mode, and deliver-order defined as required by the protocol supported (see section F.11.5 for minimal requirements for know protocols).

Requirements for other service parameters — such as Transmission Mode (ARQ/non-ARQ), Delivery Confirmation, Deliver-in-Order — **should** be defined as a function of the service requirements of the EC_FRAME data field.

F.11.5 Minimal Requirements for Supported Protocols

The range of protocols that could be supported by Ethernet, and consequently could be supported (in theory) over a STANAG 5066 Ether client is broad. Detailed requirement specifications are beyond the scope of this Annex.

Some minimal requirements for supportable protocols of interest are given below.

F.11.5.1 IPv4-over-Ether-client Service

With few exceptions — noted here — the requirements for the IP-client given in section F.12 apply when supporting IPv4-over-Ether client. In particular:

- (a) IPv4-over-Ether-client service **shall** use SAP ID 8, per the Ether client service requirements;
- (b) IP-datagram encapsulation requirements for IPv4-over-Ether client operation modify the IP-datagram encapsulation requirements of section F.12.2; the IP datagram **shall**⁽¹⁾ be encapsulated within the EC_FRAME Data Field as shown in the figure below rather than in the U_PDU field; other IPv4 datagram-encapsulation requirements of section F.12.2 **shall**⁽²⁾ apply;

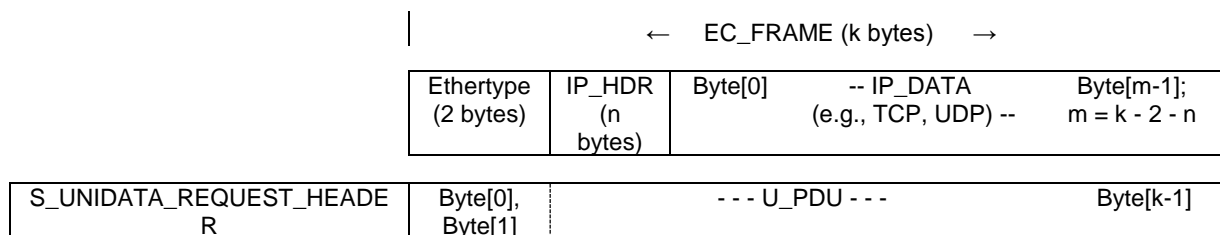


Figure F-16 - IPv4-datagram mapping into EC_FRAMEs for subnetwork transmission as an S_UNIDATA_REQUEST Primitive.

- (c) ARQ and non-ARQ services **shall** be used in accordance with section F.12.3;

- (d) the Quality of Service (QOS) requirements of section F.12.4 **shall** apply;
- (e) the Segmentation and Reassembly requirements of section F.12.5 **shall** apply;
- (f) the Internetwork Control Message Protocol (ICMP) **shall** be supported
- (g) requirements for address-resolution and mapping between IP-address and STANAG 5066 addresses given in section F.12.7 **shall** apply; in addition, IPv4-over-Ether operation **should** be capable of dynamically resolving addresses using an ARP-over-Ether client.

F.11.5.2 IPv6-over-Ether-client Service

IPv6 over Ether-client **may** be supported. In general, implementations **should**⁽¹⁾ conform to the requirements for the transmission of IPv6 over Ethernet (RFC2464), with exceptions (e.g., formation of interface identifiers for stateless autoconfiguration) as noted here.

IPv6 capabilities that depend on or are derived from an *interface identifier* (see RFC 2373, section 2.5.1)— such as the IPv6 Stateless Address Autoconfiguration protocol (RFC 2462) — **shall** use as a 32-bit interface identifier their STANAG 5066 address in the format prescribed by STANAG 5066 Annex A, sec A.2.2.28.

ARQ and non-ARQ service characteristics **should**⁽²⁾ be defined as a function of an IPv6-datagram's destination address and QoS requirements, using the IPv4 requirements of Section 12.3 as guidelines. Further requirements are currently beyond the scope of this Annex.

F.11.5.3 Compressed-IP-over-Ether-client Service

There are a number of IP compression techniques that could be used with the Ether client. All are suitable only for point-to-point (i.e., unicast-addressed) datagrams. Consequently, compressed-IP-over-Ether-client service **shall not** be enabled for multicast-IP traffic.

Van Jacobsen (RFC 1144) defined a popular TCP-/IP-header compression technique (sometimes called VJ-compression). VJ-compressed IP-datagrams **shall**⁽¹⁾ use the IEEE required Ethertype of 0x876B for transmission over Ether-client service.

RObust Header Compression (ROHC) has been defined for a number of profiles including IP-only (RFC 3843), and UDP/IP and ESP (RFC 3095). Ethertype specification for these profiles have not been standardized within IETF at this writing. Consequently, an Ethertype value of 0x0180 —from the block of values for 'experimental' use — **shall**⁽¹⁾ be used for ROHC-over-Ether-client service..

Whether used experimentally or as a later standard, IP-compression techniques over the Ether-client **should** use ARQ service and IN-ORDER Delivery mode as most of these techniques are sensitive to errors and to out-of-order delivery. As ROHC is tolerant of errors, it **may** use non-ARQ service and IN-ORDER Delivery mode.

F.11.5.4 ARP-over-Ether-client Service

The address resolution protocol (ARP - RFC 826) **may** be supported over the Ether client interface, allowing nodes to develop a table a translation table between STANAG 5066 addresses and higher-level protocol (e.g., IP) addresses. However, when possible, these tables **should** be provided as static configuration data for HF-subnetwork nodes, to reduce traffic loading and HF subnetwork congestion.

Resolution of STANAG 5066 addresses (i.e., addresses in the format in Annex A, sec A.2.2.28) to their corresponding upper-layer (e.g. IP) address **shall**⁽¹⁾ be made by representing STANAG 5066 addresses as a 48-bit 'psuedo-Ethernet address' in any ARP-packet format. The 48-bit 'psuedo-Ethernet address' shall encapsulate the STANAG 5066 address (including the size-of-address field and group-address flag) in the format defined in sec A.2.2.28), pre-pended with the two-byte field 0x5066 as shown in the Figure below:

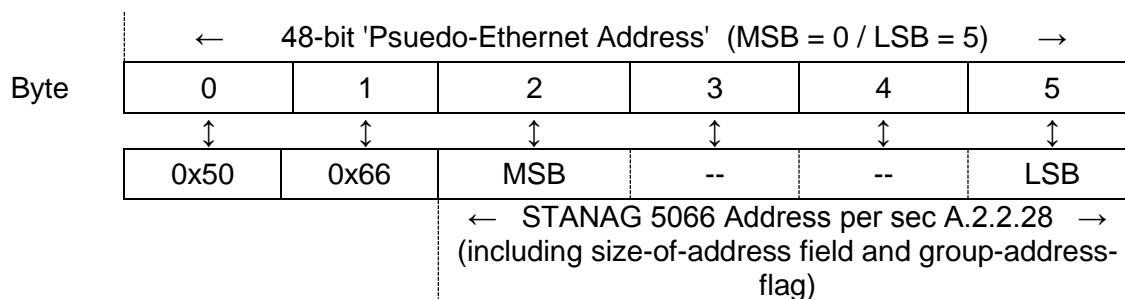


Figure F-17 - Cross-mapping between Psuedo-Ethernet Addresses used for ARP-over-Ether-client and STANAG 5066 Addresses

Other operation of ARP-over-the Ether-client **shall**⁽²⁾ be in accordance with RFC 826.

The mapping defined above between psuedo-Ethernet addresses and STANAG 5066 addresses **shall**⁽³⁾ be used for other protocols, e.g., Reverse ARP (RFC 903) and Inverse ARP (RFC 2390) that use the ARP packet-format of RFC 826.

F.11.5.5 PPP-over-Ether-client Service

Implementations of STANAG 5066 **may** provide a PPP service over the Ether client.

PPP over Ether-client service **shall**⁽¹⁾ use the IEEE Ethertype value of 0x880B.

A PPP client **shall**⁽²⁾ construct PPP frames in accordance with Section 2 of RFC 1661, using the 16-bit Protocol-field to identify the information encapsulated within the PPP and its upper-layer source.

Framing characters (e.g., the HDLC framing symbols) **shall not** be included and are not allowed. The PPP packet is submitted to the HF subnetwork encapsulated in an S_UNIDATA_REQUEST Primitive and received from the subnetwork encapsulated in an S_UNIDATA_INDICATION Primitive. These S_Primitives provide all necessary framing information to mark the beginning and end of PPP frames when transported over the HF subnetwork.

The two bytes of the PPP frame (i.e, the Protocol field) **shall**⁽³⁾ be placed in the first two bytes of the U_PDU field within the primitive, and so on to the last byte of the PPP frame and S_Primitive U_PDU field, as shown in the Figure.

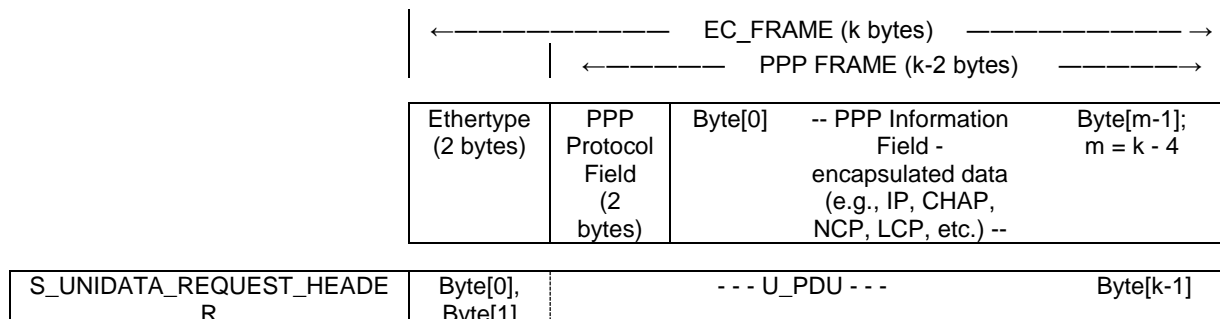


Figure F-18 - Mapping of PPP-over Ether-client service into an S_UNIDATA_REQUEST Primitive for transmission.

HF Subnetwork implementations **shall**⁽⁴⁾ provide MTU sizes sufficient to support placement of an entire PPP frame (up to 2048 bytes) within a single S_Primitive.

All PPP frames **shall**⁽⁵⁾ be submitted to the HF subnetwork using the normal S_UNIDATA_REQUEST Primitives, with the default service requirements defined as follows:

- Transmission Mode = ARQ
- Delivery Confirmation = NODE DELIVERY or CLIENT DELIVERY
- Deliver in Order = IN-ORDER DELIVERY

IN-ORDER Delivery is specified because the PPP protocol requires a serial, sequenced bearer service

The address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host to which the PPP frame is being sent.

The encoded data of the PPP frame **shall**⁽⁶⁾ be bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_PRIMITIVE, with the least-significant bit (LSB) or each character aligned with the LSB of the octet.

Implementers of PPP-over-Ether clients for STANAG 5066 HF subnetworks **shall**⁽⁷⁾ provide support for the PPP service-negotiation protocol that initiates any PPP session.

Implementers of PPP clients **should** provide support for the data compression capabilities of the Point-to-Point Protocol as a means of promoting HF subnetwork efficiency. Note that differences in data compression capabilities between clients will be resolved by PPP's initial service-negotiation protocol

F.12 INTERNET PROTOCOL (IP) CLIENT

NATO support for the Internetwork Protocol is mandated within the NC3TA and NCSP. The requirements for HF-subnetwork support of the Internet Protocol (Version 4), Internet Standard 5, Request for Comments (RFC) 791, are defined here. The Internet Protocol is a well known protocol whose requirements are not repeated here.

F.12.1 General Requirements

Implementations of STANAG 5066 **shall**⁽¹⁾ provide an IP client to the HF subnetwork, with some suitable implementation-dependent but standards-based local area network (LAN) interface for external access, such as Ethernet with 100/10 Base T / RJ-45 physical connector. The LAN interface for external access to the IP Client may be shared with the Raw SIS Socket Server (see Section F.16 of this Annex.).

The requirements of this section **shall**⁽²⁾ apply to any client that attaches directly to the HF subnetwork interface sublayer to send or receive IP datagrams over the HF subnetwork. Such clients include end systems (e.g., web-servers or mail systems) and intermediate systems (e.g., routers).

Over the HF subnetwork interface, the IP client **shall**⁽³⁾ be capable of sending and receiving encapsulated IP datagrams with unicast (i.e., point-to-point) IP addresses, using both ARQ- and non-ARQ-transmission modes in STANAG 5066.

Over the HF subnetwork interface, the IP client **shall**⁽⁴⁾ be capable of sending and receiving encapsulated IP datagrams with multicast (i.e., point-to-multipoint) IP addresses, using non-ARQ-transmission modes.

F.12.2 Encapsulation of the IP Datagram within HF Subnetwork S_PRIMITIVES

IP datagrams **shall**⁽¹⁾ be encapsulated within the U_DPU field of S_UNIDATA_REQUEST Primitives submitted to the HF subnetwork for transmission, and delivered to clients at the destination node(s) within the U_DPU field of S_UNIDATA_INDICATION Primitives. There are no framing characters required or allowed.

The first byte of the header of the IP datagram **shall**⁽²⁾ be aligned with the first byte of the U_PDU field within the primitive, and so on to the last byte of the IP datagram and U_PDU field.

The encoded bytes of an IP datagram submitted for transmission over the subnetwork **shall**⁽³⁾ be bit-/byte-aligned with the octets in each U_PDU encapsulated in the S_UNIDATA_REQUEST primitive.

The least-significant bit (LSB) of each octet in the IP datagram **shall**⁽⁴⁾ be aligned with the LSB of the U_PDU's octet.

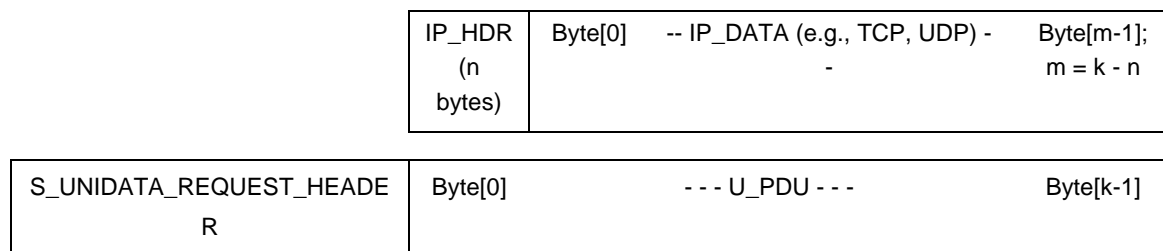


Figure F-19 - Mapping of IP Datagram into an S_UNIDATA_REQUEST Primitive for transmission.

F.12.3 IP-Client Subnetwork Service Requirements

IP clients **shall**⁽¹⁾ bind to the HF Subnetwork at SAP ID 9.

The client rank is a scenario-dependent parameter. Use of Rank = 15 is discouraged for anything but an IP client performing subnetwork management functions.

IP support by the HF subnetwork **shall**⁽²⁾ be configurable to use either ARQ or non-ARQ delivery services.

Selection of the subnetwork's default service requirements (established in the S_BIND_REQUEST message) and delivery mode (established in the S_UNIDATA_REQUEST message), and traffic priority **shall**⁽³⁾ be based on the type of IP address and IP Quality-of-Service (QoS) labels.

An IP client **may** set the subnetwork's default service requirements in the S_BIND_REQUEST message as a function of the most likely traffic that it expects to process in accordance with the requirements of section F.12.4

An IP client **shall**⁽⁴⁾ be capable of overriding the subnetwork's default Service Type requirements and dynamically setting the S_UNIDATA_REQUEST Delivery Mode for each IP Datagram submitted to the HF subnetwork in accordance with the requirements of Section F.12.4.

F.12.3.1 IP Support using ARQ Service

HF subnetwork support using reliable point-to-point delivery between a pair of nodes is preferred for efficiency in the IP and higher-layer protocols, but in general cannot support IP-multicast protocols. [NB: The exceptional case is when an IP multicast address can be mapped to a STANAG 5066 unicast address, e.g., when tunnelling multicast traffic over an HF point-to-point link.]

The service definition for reliable-IP datagram delivery using the ARQ service **shall**⁽¹⁾ be as follows:

1. Transmission Mode = ARQ,
2. Delivery Confirmation = NODE DELIVERY,
3. Deliver in Order = IN-ORDER DELIVERY or AS-THEY-ARRIVE, as determined by QoS service requirements per section F.12.4.

The destination address in the primitive will be an individual node address corresponding to the HF subnetwork address of the host to which the IP datagram is being sent. NOTE that this may NOT be the IP address contained within the datagram itself, because of the relay and routing properties of the IP protocol.

Determination of the correspondence between HF subnetwork and IP addresses **may** be accomplished through the use of static look-up tables or the Address Resolution Protocol (ARP), Internet Standard 37 (RFC 826); use of ARP shall be supported using the ARP-over-Ether-client service defined in Section F.11.

IP client implementations **shall**⁽²⁾ provide a static look-up table for address-resolution for at least 10 IP addresses (unicast, multicast, or IP subnetwork).

F.12.3.2 IP Support using non-ARQ Service

If IP-multicast address groups must be supported within the HF subnetwork environment (i.e., an application wishes to take advantage of the broadcast nature of the HF channel to support IP multicast), then the HF subnetwork **shall**⁽¹⁾ be configured in non-ARQ mode to support this requirement.

IP support using non-ARQ service **may** be used or specified for IP unicast services, e.g., to meet the Quality-of-Service (QoS) management requirements of Section F.12.4.

For IP datagrams using non-ARQ service, the HF subnetwork service **shall**⁽²⁾ be configured as follows:

1. Transmission Mode = non-ARQ,
2. Delivery Confirmation = none,
3. Deliver in Order = IN-ORDER DELIVERY or AS-THEY-ARRIVE, as determined by QoS service requirements per section F.12.4.

The number of repeats for the D_DPUs in the service **may** be set to a value greater than one to provide some increased probability of receipt and reliability when using the subnetwork for IP multicast support.

F.12.4 Quality-of-Service Requirements for IP Datagrams

The IP client **shall**⁽¹⁾ have the capability of turning Quality-of-Service (QoS) management ON or OFF. More than one QoS management mode **may** be selectable when QoS is turned ON.

An IP client **may** support QoS management for IP datagrams using the IP Type-of-Service (TOS) Byte in accordance with the following subparagraph on RFC1349 TOS.

An IP client **shall**⁽²⁾ support QoS management for IP datagrams using the Differentiated Services (DiffServ) in accordance with the following subparagraphs.

F.12.4.1 QoS Management using IP Type-of-Service (TOS) i.a.w. RFC1349

The IPv4 datagram has had subfields defined within the datagram's type-of-service (TOS) byte as shown in the following Figure.

| | | | | | | | |
|------------|---|---|-------|---------|------------------|------|------|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Precedence | | | Delay | Thruput | Relia- bility | Cost | ---- |

Figure F-20 – IP Packet Type-of-Service (TOS) Byte, with Precedence and Quality of Service subfields

When configured to support QoS management using the IP TOS byte, an IP client shall specify the priority and delivery mode for S_UNIDATA_REQUEST primitives as follows:

1. The three-bit precedence field in the IP TOS byte **shall** be mapped onto the priority field of the 5066 S_UNIDATA_REQUEST primitive in accordance with the following table.

Table F-7 Mapping between IP Datagram Precedence Levels and S_PRIMITIVE Priority Values

| TOS Byte Precedence | | | S_PRIMITIVE Priority Field (for S_UNIDATA_REQUEST and S_UNIDATA_INDICATION) | | | | Comments |
|------------------------|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 7 | 6 | 5 | 4 | Bit Position within the octet (note the reversal of bit positions) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | Lowest IP priority level |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | Highest IP priority level; reserves one level higher |

2. The IP-address type, i.e., whether the address is a unicast address or multicast address, and the remaining bit-fields in the TOS byte **shall** be used to select ARQ or non-ARQ delivery modes in accordance with the table shown.

Table F-8 Delivery and Transmission Mode Selection as a Function of IP-Address Type
TOS parameters (X = a 'don't-care' state)

| IP-address Type | Delay | Throughput | Reliability | Cost | Transmission (Delivery) Mode | Comments |
|-----------------|-------|------------|-------------|------|------------------------------|--|
| Multicast | X | X | 0 | X | Non-ARQ | - this selection may be overridden in the case where an IP multicast address is resolvable into a unicast STANAG 5066 address. |
| Multicast | X | X | 1 | X | Non-ARQ w/ repeats > 1 | - this selection may be overridden in the case where an IP multicast address is resolvable into a unicast STANAG 5066 address. |
| Unicast | 1 | 0 | 0 | 0 | Non-ARQ | "Minimize-delay" [RFC1349] |
| Unicast | 0 | 1 | 0 | 0 | ARQ | "Maximize throughput" [RFC1349] |
| Unicast | 0 | 0 | 1 | 0 | ARQ | "Maximize Reliability" [RFC1349] |
| Unicast | 0 | 0 | 0 | 1 | Non-ARQ | "Minimize Cost" [RFC1349] |
| Unicast | | | | X | | Other combinations are allowed and remain to be specified |

F.12.4.2 QoS Management using Differentiated Services (DiffServ) Code Points (DSCP)

As a preferred mode of Quality of Service (QoS) management for IP packet transmission, the IP client **shall**⁽¹⁾ provide QoS management based on the Differentiated Services (DiffServ) field. [NB: the bit locations for the TOS-byte and DS field in the IPv4 header are identical. Differentiated Services redefines the meaning of these bits as noted in RFC 2474, RFC 2597, and RFC 2598.].

As defined in RFC 2474 section 3, the Differentiated Services field in the IPv4 header is the Type-of-Service byte, with bit positions defined as in the figure below:

| | | | | | | | |
|---|---|---|---|---|---|---------------------------|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Differentiated Services Code Point (DSCP) | | | | | | CU = currently unassigned | |

Figure F-21 – IP Packet Differentiated Services Field

The IP client **shall**⁽²⁾ provide a mechanism for mapping the Differentiated Services Code Point (DSCP) for each transmitted IP packet to the priority and service requirements of S_PRIMITIVE that encapsulates it.

The mapping between given DSCP values and corresponding STANAG 5066 service and priority markings **shall**⁽²⁾ be configurable.

The default mapping between given DSCP values and corresponding STANAG 5066 service and priority markings **shall**⁽³⁾ be as shown in the table below:

Table 9 - Default DSCP to STANAG 5066 Service and priority mapping

| DSCP | IP Precedence Name | DSCP Name | S'5066 Priority Name | S'5066 Priority Value | Transmission Mode (*) |
|--------|--------------------|--------------|----------------------|-----------------------|-----------------------|
| | | | <i>reserved</i> | 1111 | |
| 111xxx | Network | | Network | 1110 | ARQ/non-ARQ |
| | | | <i>Reserved</i> | 1101 | |
| 110xxx | Internet | | Internet | 1100 | ARQ/non-ARQ |
| | | | <i>Reserved</i> | 1011 | |
| 101xxx | Critical | EF | Expedited | 1010 | ARQ/non-ARQ |
| | | | <i>Reserved</i> | 1001 | |
| 100xxx | Flash-Override | AF4 | Flash-Override | 1000 | ARQ/non-ARQ |
| | | | <i>reserved</i> | 0111 | |
| 011xxx | Flash | AF3 | Flash | 0110 | ARQ/non-ARQ |
| | | | <i>Reserved</i> | 0101 | |
| 010xxx | Immediate | AF2 | Immediate | 0100 | ARQ/non-ARQ |
| | | | <i>Reserved</i> | 0011 | |
| 001xxx | Priority | AF1 | Priority | 0010 | ARQ/non-ARQ |
| | | | <i>Reserved</i> | 0001 | |
| 000xxx | Routine | Default DSCP | Routine | 0000 | ARQ/non-ARQ |

* transmission mode mapping: unicast IP datagram -> ARQ mode; multicast IP datagram -> non-ARQ mode

F.12.5 Segmentation and Reassembly at the Subnetwork Interface

Any S_PRIMITIVE containing IP data (in particular the S_UNIDATA_REQUEST or S_UNIDATA_INDICATION primitives) **shall**⁽¹⁾ contain a complete IP datagram.

An IP client **may** implement IP Datagram Segmentation and Reassembly in accordance with the applicable Internet Standards and consistent with the Maximum Transmission Unit (MTU) of the HF subnetwork. Note however that such implementation in accordance with applicable Internet standards will result in the encapsulation of a complete and fully formed IP datagram within an S_PRIMITIVE (whether of 'request-' or 'indication-' type). [NB: implementation of IP Segmentation and Reassembly protocols within an IP client is discouraged, in accordance with the recommendations of the Phil Karn et al., "Advice for Internet Subnetwork Designers", Internet Engineering Task Force, July 2000.]

An IP client **shall**⁽²⁾ conform to the requirements of the Internet Path MTU discovery (PMTU) Protocol [RFC1191] for IP datagrams marked with the DONT_FRAGMENT flag that also exceed the HF subnetwork MTU size.

F.12.6 Internetwork Control Message Protocol

An IP client **shall** implement the Internetwork Control Message Protocol (ICMP). [NB: this requirement is levied to support error-reporting to an IP traffic source related to its use of the HF subnetwork, e.g., to support the PMTU protocol, and flow-control over the narrow-bandwidth HF subnetwork.]

F.12.7 Requirements for Address Resolution and Mapping between IP Address Formats and STANAG 5066 Addresses

An IP client **shall**⁽¹⁾ translate addresses between the IP unicast address domain and 5066 unicast address domain.

An IP client **shall**⁽²⁾ provide address translation between the IP-multicast address domain and 5066-multicast-address domain.

Users of the IP client **shall**⁽³⁾ be capable of defining a static lookup table to perform address translation.

Other methods of establishing correspondence between IP and STANAG 5066 address domains **may** be implemented.

F.12.8 Filtering Requirements

An IP client **may** filter IP management packets to prevent them from being sent over the subnetwork, reducing the traffic load required by IP management functions.

F.12.9 Proxy Requirements

An IP client **may** proxy the Internetwork Group Management Protocol (IGMP) for multicast destinations on the HF subnetwork; proxy configuration is performed statically in the basic client.

F.12.10 IPv6 Support

Support for INTERNET PROTOCOL Version 6 **may**⁽¹⁾ be provided by use of IPv6-over-IPv4 tunnels (encapsulation of IP datagrams within the payload segment of another IP datagram) and other evolution strategies in accordance with INTERNET Request for Comments (RFC) 2893.

Support for INTERNET PROTOCOL Version 6 **may**⁽²⁾ be provided by use of the Ether client of section F.11.

F.13 RESERVED SERVICE ACCESS POINT IDENTIFIERS

Some Service Access Point Identifiers numbers are reserved for future use.

Clients **shall** not bind to Service Access Point Identifiers marked in Table F-1 as reserved for future assignment.

F.14 COMPRESSED FILE-TRANSFER PROTOCOL (CFTP) CLIENT

The Compressed File Transfer Protocol (CFTP) **may** be used to meet requirements in the NATO C3 Technical Architecture and the NATO Common Standards Profile (NCSP) for use of the SMTP Protocol. CFTP is used to reliably send compressed SMTP e-mail over a STANAG 5066 HF subnetwork from one SMTP mail server to another.

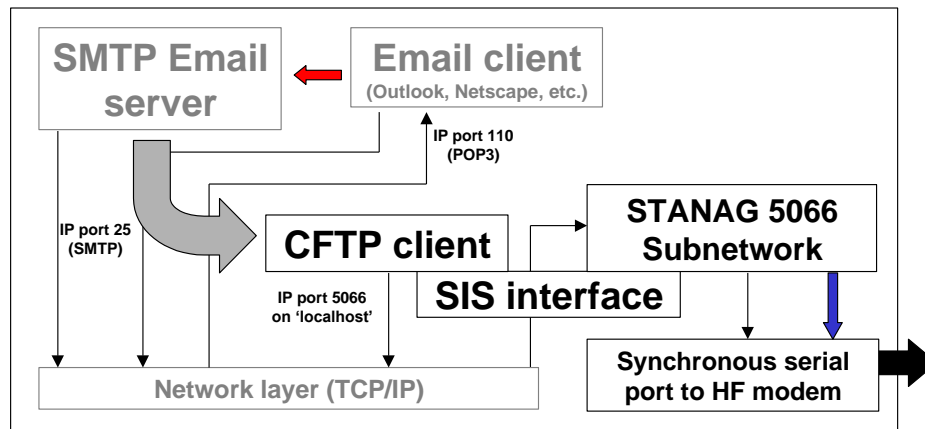
CFTP is a vendor-defined standard derived from client definitions made in earlier, non-mandatory, versions of STANAG 5066 Annex F, and subsequently adopted by MIL-STD-188-110B as a mandatory provision. [NB: MIL-STD-188-110B has adopted HMTP as a requirement only for interoperability with NATO HF networks using HMTP.]

F.14.1 General Requirements

Implementations of STANAG 5066 **may** provide a CFTP client. If provided, a CFTP client **shall**⁽¹⁾ conform to the requirements specified herein.

The CFTP protocol **shall not** be used for Formal or High Grade Military Messaging (i.a. military orders). For Formal or High Grade Military Messaging, STANAG 4406 Annex E **shall**⁽²⁾ be used (see section F.4). The CFTP protocol **may** be used for informal interpersonal e-mail only.

CFTP **shall**⁽³⁾ operate within the node model shown, providing transfer services from one SMTP e-mail server to another via the CFTP client.



F - 22 - CFTP Client and Node model

In general, the interaction of the CFTP client with the mail-server is beyond the scope of this STANAG. In operation, when an email message is received at a 5066 node, it is placed in an incoming mail folder (mail spool directory). The CFTP client, also called the Delivery Agent (DA), removes mail from this incoming folder and processes the mail for delivery over HF via 5066. The CFTP DA compresses the message and information about the message, e.g. size, id, recipients, etc. into a file. This compressed file is then transferred to the destination 5066 node(s) using the original form of the Basic File Transfer Protocol (BFTP), referred to here as BFTPv1. The original BFTPv1 format is incorporated directly in the CFTP specification below to free it from dependencies on (and incompatibilities with) BFTP specification found in Section F.10.2 of this Amendment 1.

F.14.2 CFTP Subnetwork Service Requirements

CFTP clients **shall** bind to the HF Subnetwork at SAP ID 12.

F.14.3 CFTP Connection-Oriented Protocol

The CFTP application **shall** use the earlier form of the RCOP Protocol Data Unit ("RCOPv1") defined in the Figure below. [NB: As a historical note, this corresponds to the definitions of the RCOP protocol found in the original information-only Annex F Edition 1. It does not include the Application Identifier within the PDU Header that has been added in this Amendment 1.].

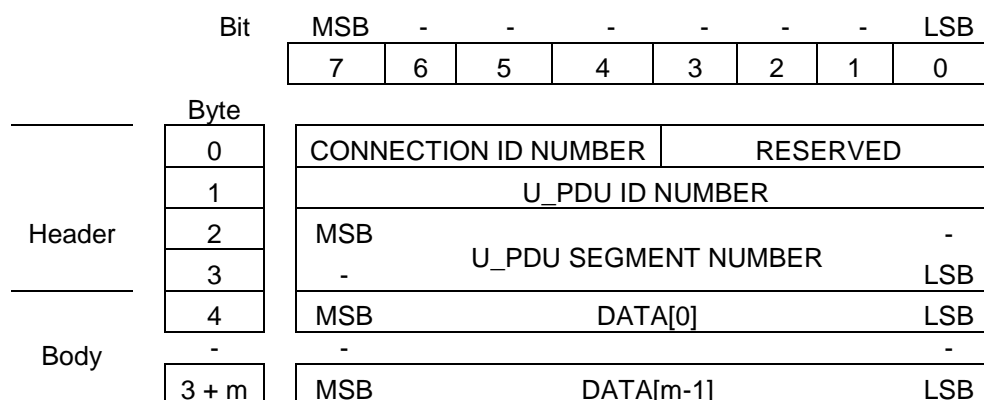


Figure F-23. CFTP Protocol Data Units (identical in format to (Original) RCOP Protocol Data Units (RCOPv1) from STANAG 5066 Annex F Edition 1)

The following are required for RCOPv1 PDUs:

1. The connection ID number **shall** be a value from 0-15. Connection ID number 0 shall be reserved for non-multiplexed connections.
2. The reserved bits **shall** be set to 0.
3. The U_PDU ID numbers **shall** be assigned consecutively to U_PDU (i.e., files).
4. The U_PDU segment number **shall** be assigned consecutively to segments within a single U_PDU. The first segment transmitted **shall** be assigned segment number 0. If a U_PDU is not segmented, the single segment transmitted **shall** be assigned number 0.

F.14.3.1 Compressed File-Delivery and Delivery-Confirmation

Compressed files **shall**⁽¹⁾ be transferred from one CFTP client to another using the Edition 1 Basic File Transfer Protocol ('BFTPv1') as defined in the subsections below.

Client Delivery confirmation **shall**⁽²⁾ be provided using the CFTP Message Acknowledgement, defined in the subsequent section (see F.14.3.2), as the body-part of the PDU.

In principle, up to 256 files could be transferred concurrently using the unique RCOPv1 U_PDU ID number for each transfer, using the U_PDU_ID as the identifier to match acknowledgements to the file acknowledged. As there is no negotiation protocol currently defined to determine if a given receiving node supports this capability, a sending node **must** have prior knowledge that a given receiving node supports concurrent multiple-file delivery.

Consequently, the Client Delivery confirmation protocol is nominally stop-and-wait — a new file **should not**⁽¹⁾ be sent with a given U_PDU ID until a message acknowledgement has been received. However, this recommendation **may**⁽¹⁾ be relaxed to allow concurrent multiple-file delivery when the sending node has prior knowledge that the receiving node supports the capability. New implementations of CFTP **should** support concurrent multiple file delivery.

F.14.3.1.1 BFTPv1 Specification [NB: corresponding to the original Edition 1 BFTP specification]

The format for the basic-file-transfer-protocol data unit Version 1 (BFTPv1) **shall**⁽¹⁾ be in accordance with the following Figure, which defines a header part and a file-data part for the BFTP_PDUv1.

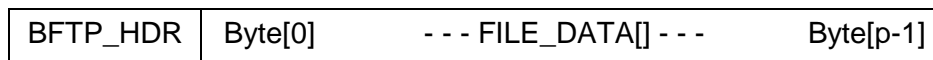


Figure F-24: Basic FTP Version 1 Protocol Data Unit (BFTPv1 PDU)

The detailed structure of the BFTPv1_PDU **shall**⁽²⁾ be in accordance with the following Figure, and provide the following information fields:

1. BFTPv1_PDU Header Part:
 - SYNCHRONIZATION - two bytes corresponding to the control bytes DLE (Data Link Escape) and STX (Start of Text).
 - SIZE_OF_FILENAME - one octet in size.
 - FILE_NAME - a variable length field, equal in size to the value specified by the SIZE_OF_FILENAME field.
 - SIZE_OF_FILE - a four-octet field.
2. BFTPv1_PDU Body Part:
 - FILE_DATA[] - a variable length field, equal in size to the value specified by the SIZE_OF_FILE field.

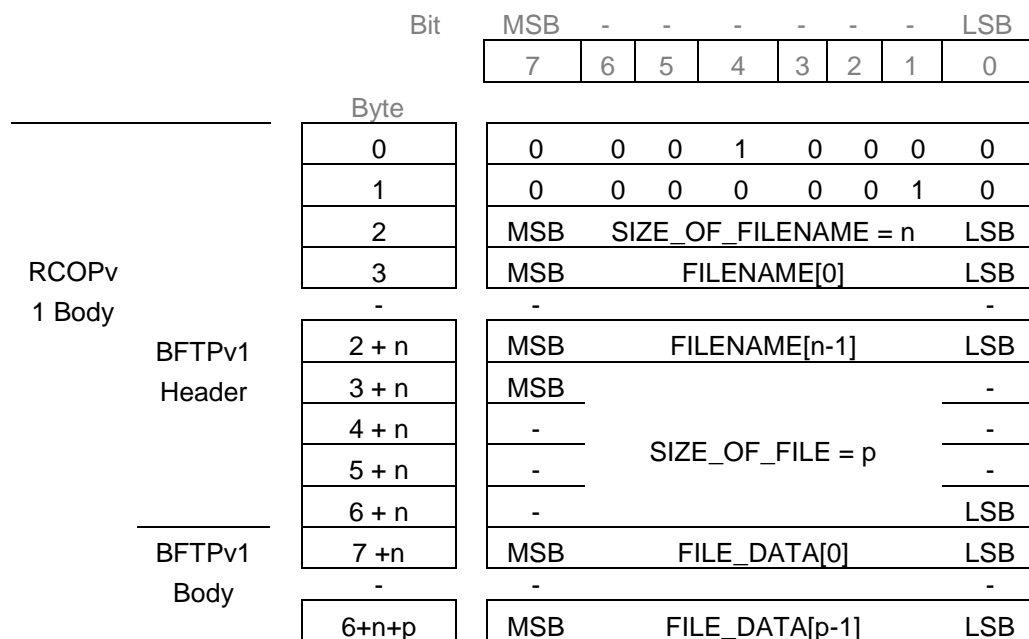


Figure F-25: BFTPv1 Protocol Data Unit Structure

The SIZE_OF_FILENAME field **shall**⁽¹⁾ be a 1-octet fixed-length field whose value (n) **shall**⁽²⁾ equal the number of octets used to encode the FILENAME field.

The FILENAME field is **shall**⁽¹⁾ be a variable-length field, the size of which **shall**⁽²⁾ be specified by the value (n) of the field SIZE_OF_FILENAME. This field represents the name of the file sent using the Basic File Transfer Protocol. The first byte of the filename **shall**⁽³⁾ be placed in the first byte of this field, with the remaining bytes placed in order. The semantics of file names and naming conventions are beyond the scope of this STANAG (e.g., there is no requirement that the filename be a null-terminated character string.)

The SIZE_OF_FILE field **shall**⁽¹⁾ be a 4-octet fixed-length field whose value **shall**⁽²⁾ specify the size (p) in octets of the file to be sent. The first octet of the SIZE_OF_FILE field **shall**⁽³⁾ be the highest order byte and the last byte the lowest order byte of the field's binary value.

F.14.3.1.2 BFTPv1 Segmentation and Reassembly Requirements

If the BFTPv1_PDU exceeds the maximum size of the data field permitted in the RCOPv1 PDU (i.e, if the CFTP_PDU is larger than the MTU_size less 4 octets (i.e., MTU-4)), the CFTP client **shall** segment the BFTPv1 PDU, placing successive segments in RCOPv1 PDUs (original Edition 1 format) with consecutive U_PDU sequence numbers.

When received, the CFTP client **shall**⁽²⁾ reassemble the BFTPv1 PDU if it determines that the BFTPv1 PDU has been segmented. Subject to local-host file naming conventions, the CFTP client **shall**⁽²⁾ store the received file with the name transmitted in the header with the file. *[NB: there is no guarantee therefore that the file will be stored on the destination host with the same name that it was sent.]*

F.14.3.2 Message Acknowledgement

Client Delivery confirmation **shall**⁽¹⁾ be provided using the Message Acknowledgement defined below, sent as the body-part of a CFTP/RCOPv1 PDU.

| | | | | | | | | | | | | | | | | | |
|---|--|--|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Bit | | | | | | | | | | | | | | | |
| | | MSB - - - - - - - LSB | | | | | | | | | | | | | | | |
| | | <table><tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr></table> | | | | | | | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | | | | | | | | | |
| <hr/> BFTPv1 ACK (RCOPv1 body part) | Byte | | | | | | | | | | | | | | | | |
| | 0 | <table><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td></tr></table> | | | | | | | | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | | | | | | | | | |
| 1 | <table><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td></tr></table> | | | | | | | | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | | | | | | | | | | |

Figure F-26: BFTPv1 Message Acknowledgement Structure

On receiving the last byte of the message, the receiving client **shall**⁽²⁾ send the Message Acknowledgement (0x10 0x0B) — with the same RCOPv1 U_PDU_ID NUMBER and CONNECTION

ID NUMBER as the message being acknowledged — to confirm that the entire message has been received. (N.B. This is equivalent to the "ZEOF" message of the Z-modem protocol.)

F.14.4 CFTP Compression/Decompression

The compressed file **shall** be created and decompressed in accordance with RFCs 1950, 1951 and 1952 (i.e., the gzip utility defined in RFC1952).

F.14.5 CFTP Compressed File Data Format

The compressed file data **shall** be formatted as a series of fields. The fields are separated by the linefeed <LF> character 0x0A. The fields and the order in which they are compressed are described in the following table.

Table F-10 CFTP Mail File Structure

| Order of Compression | Field Name | Description |
|----------------------|---------------|--|
| 1 | MessageID | The MessageID field is represented by an arbitrary string that serves as the ID for the message. The MessageID is unique to an e-mail message. It is not the same as the ID in the email message that follows "Message-ID:" in the header. When the compressed file is decompressed, the MessageID is used as the root filename for the decompressed components. The MessageID must be less than 256 characters and is composed of upper/lowercase alphanumeric characters. |
| 2 | RecipientList | The RecipientList is a string containing e-mail addresses extracted from the e-mail message, each address delimited by the "," character (0x2c). The first address in the recipients list is the "Return-Path". There can be cases where there is no return path, e.g. the mail is being bounced by a Mail Transfer Agent. In these cases, the first address will be an empty string (i.e., either a single space [0x20] character or no characters at all) and it will be followed by a comma (0x2c). The recipients list must be less than 10240 characters. |
| 3 | MessageSize | The MessageSize is encoded as a decimal number in string format. It represents the size (in bytes) of the Message field that follows the MessageSize field. |
| 4 | Message | Actual message as received by an SMTP receiver <i>i.e. including the terminating sequence <CRLF>.<CRLF> and any additional characters that may be required for transparency as defined by RFC 821 para 4.5.2 .</i> |

Note 1. All characters are 8 bits.

Note 2. The terminating sequence <CRLF>.<CRLF> is that shown in Example 1 of RFC 821 and equates to the 5 ASCII Characters with codes, in hexadecimal, of 0x0D, 0x0A, 0x2E, 0x0D, 0x0A.

F.14.6 Detailed Description of CFTP

- 1) An e-mail client is used to send an e-mail to an SMTP server.
- 2) The CFTP application extracts the e-mail message from the directory in which it was placed by the SMTP server. An example e-mail message in the correct format is shown in Figure F-23 below.
- 3) The CFTP mail-file **shall** be built as follows:

| | |
|---|--|
| <code><MessageID><LF></code> | <code>//</code> The MessageID field shall ⁽¹⁾ be represented by a character string. The MessageID shall ⁽²⁾ be unique to an e-mail message. It is not the same as the ID in the email message that follows "Message-ID:" in the header. When the compressed file is decompressed, the MessageID shall be used as the root filename for the decompressed components. The MessageID must be less than 256 characters and may be composed of upper/lowercase alphanumeric characters. |
| <code><RecipientsList><LF></code> | <code>//</code> The RecipientList shall ⁽¹⁾ be a character string containing e-mail addresses extracted from the e-mail message, each address separated by "," character (0x2c). The first address in the recipients list shall ⁽²⁾ be the "Return-Path". There may be cases where there is no return path, e.g. the mail is being bounced by a Mail Transfer Agent. In these cases, the first address shall ⁽³⁾ be an empty string (i.e., either a single space [0x20] character or no characters at all) followed by a comma (i.e, a "," character with octet value = 0x2c). The recipients list must be less than 10240 characters. |
| <code><MessageSize><LF></code> | <code>//</code> The MessageSize shall be encoded as a decimal number in string format terminated by the linefeed character. It represents the size (in bytes) of the Message field that follows |
| <code><Message></code> | <code>//</code> The Message field shall contain the e-mail message body part(s) extracted from the SMTP envelope. |

- 4) The CFTP message (including header) **shall** be compressed in accordance with RFCs 1950, 1951 and 1952 using an application such as gzip.
- 5) The compressed CFTP message **shall** be encapsulated within a BFTPv1 PDU (i.e., it has a BFTPv1 header prepended to it, and the CFTP message shall be byte aligned within the FILE_DATA[] field of the BFTPv1 PDU.
- 6) The BFTPv1 message (i.e., BFTPv1 PDU) **shall** be segmented if necessary.
- 7) Each BFTPv1 PDU segment **shall** have an RCOPv1 header added (in accordance with Annex F.14.3.).
- 8) Each RCOPv1 packet **shall** be packaged into an S_UNIDATA_REQUEST and transferred using a Soft Link Data Exchange.

- 9) On reception the BFTPv1 message **shall** be reassembled, if required, and decompressed using a method compliant with RFC 1952 and the CFTP message reconstructed.
- 10) The received email messages **shall** be forwarded to an SMTP server using a standard SMTP dialogue based on information extracted from the CFTP header and inserting the "message" field into the payload of the SMTP message generated.

```
Received: from northampton (unverified [127.0.0.1]) by northampton.pdw<CRLF>
(Rockliffe SMTPRA 4.2.4) with SMTP id <B0000000133@northampton.pdw> for
<root@essex.pdw>;<CRLF>
Wed, 9 May 2001 12:09:03 +0100<CRLF>
Message-ID: <001f01c0d878$74da90d0$0d02a8c0@pdw><CRLF>
From: "Northampton" <administrator@northampton.two><CRLF>
To: <root@essex.pdw><CRLF>
Subject: Test <CRLF>
Date: Wed, 9 May 2001 12:09:03 +0100<CRLF>
MIME-Version: 1.0<CRLF>
Content-Type: text/plain;<CRLF>
        charset="iso-8859-1"<CRLF>
Content-Transfer-Encoding: 7bit<CRLF>
X-Priority: 3<CRLF>
X-MSMail-Priority: Normal<CRLF>
X-Mailer: Microsoft Outlook Express 5.50.4522.1200<CRLF>
X-MimeOLE: Produced By Microsoft MimeOLE V5.50.4522.1200<CRLF>
<CRLF>
This is the body of the test email<CRLF>
<CRLF>
.<CRLF>
```

Figure F-27 Example email in the Correct Format

The red and blue text above is the message body with text in blue being the terminating sequence <CRLF>.<CRLF> i.e 0x0D, 0x0A, 0x2E, 0x0D, 0x0A.

F.15 UNASSIGNED SERVICE ACCESS POINT IDENTIFIERS

Service Access Point Identifiers numbers 13, 14, and 15 **may** be used by any arbitrary client type.

Client types that already have assigned SAP_IDs **may** use one of these unassigned Service Access Point Identifiers under various usage scenarios, for example to allow multiple clients of the same type to attach to the same subnetwork simultaneously.

Other currently undefined clients of arbitrary type **may** bind using an unassigned SAP ID. For example, these might include new client types that do not wish to bind as an RCOP or UDOP client.

There **will be**, in general, no guarantee that a client of a given type is bound to any of these unassigned SAP IDs. Standard operating procedure or out of band coordination **will be** required in a given scenario to provide such guarantee.

F.16 RAW SIS SOCKET SERVER

Implementations of the STANAG 5066 subnetwork profile **shall** provide a TCP/IP socket-server interface as the physical channel for connecting a client to the Subnetwork Interface Sublayer (SIS) of the STANAG 5066 subnetwork. TCP/IP socket servers that conform to the requirements of this section are referred to as a Raw SIS Socket Server. [NB: The requirements of this section formalize some of the informational remarks given in STANAG 5066 Annex A.]

NB: A Raw SIS Socket Server **shall** be provided even for implementations that host client and subnetwork software on the same host computer.

The Raw SIS Socket Server **shall** be accessible by clients that are not co-hosted with the HF subnetwork software via a standards-based Local-Area-Network interface. Use of Ethernet 100/10 Base T or similar common standard is **recommended**.

The placement and scope of the Raw SIS-Socket-Server interface are illustrated in the following Figure.

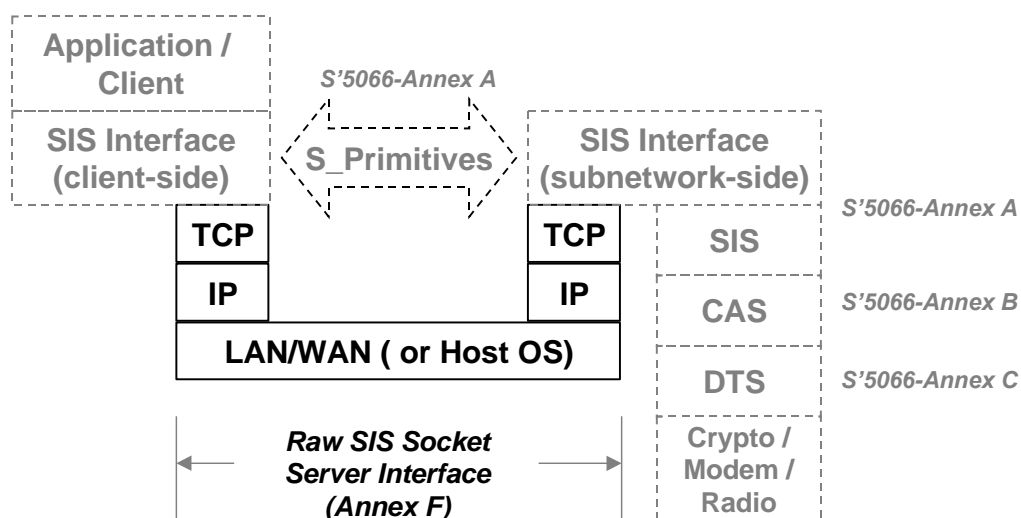


Figure 28 - Scope and placement of the Raw SIS-Socket-Server Interface

The Default SIS Port Number for the Raw SIS Socket Server **shall** be the decimal number '5066', a number registered with the Internet Assigned Number Authority (IANA) for this purpose. STANAG 5066 implementations, both clients and subnetworks, **should** be configurable to use any other valid port number as the default.

The Raw SIS Socket Server **shall** listen on the Default SIS Port Number for connection requests from clients.

The Raw SIS Socket Server **shall** accept a minimum of five connections simultaneously. Implementations **should** make the number of simultaneous connections a configurable parameter to allow support for a larger number.

Once a connection between client and subnetwork has been established over the Raw SIS-Socket-Server Interface, further communication between a client and the HF subnetwork **shall** be in accordance with the requirements of STANAG 5066 Annex A, using the S_PRIMITIVES defined in that Annex.

The Raw SIS-Socket-Server Interface **shall** be bi-directional, allowing a client to send S_PRIMITIVES to the subnetwork, and the subnetwork to send S_PRIMITIVES to the client.

S_PRIMITIVES **shall** be sent over the Raw SIS-Socket-Server Interface without any separating characters or framing data other than the generic S_PRIMITIVE encoding elements (e.g., the preamble, version, and size-of-primitive fields) currently defined in STANAG 5066 A (section A.2.2.2).

Only S_PRIMITIVES **shall** be sent over the Raw SIS-Socket-Server Interface; no other messages between client and subnetwork. Implementation-dependent communication between a client and subnetwork over the Raw SIS-Socket-Server Interface for the purposes of system management shall be encapsulated within the S_MANAGEMENT_MSG_REQUEST and S_MANAGEMENT_MSG_INDICATION primitives defined in Annex A. [NB: This does not preclude implementation-dependent communication over another socket on a different port number, but such use is outside of the scope of this STANAG and is not recommended.]

Annex G: Use of Waveforms at Data Rates Above 2400 bps (information only)

G.1.0 Introduction

This Annex presents guidelines and requirements for the use of the STANAG 5066 protocol profile with modems and waveforms at rates above 2400, when these waveforms become standardized and available in future.

G.2.0 Background

Currently, NATO standards for HF waveforms such as STANAG 4285 and 4529 are limited to data rates of 2400 bps and below. These standards form the basis for support for the STANAG 5066 HF protocol profile in its current form. However, on-going research, development, and standardization attempts for high-speed HF waveforms show promise of increasing the usable data rate on HF long-haul channels to 9600 bps and higher. Early draft versions of STANAG 5066 Annex G included a detailed specification for high-speed single-tone waveform and convolutional forward-error-correction coding with data rates up to 9600 bps, and this formed the basis for more than one vendor implementation of a commercial product. Subsequent waveform designs also have been submitted and considered for inclusion in this STANAG. A design for a second single-tone waveform with multi-level Reed-Solomon-coding was submitted for consideration, also with rates to 9600 bps. Finally, a third waveform design based on Orthogonal Frequency Division Multiplexing and turbo-coding was submitted, with projected data rates to over 10 kb/s. No timely agreement has been reached on which, if any, of these waveforms should form the basis for a NATO standardization agreement or be included in STANAG 5066.

G.3.0 Implementation Guidance for STANAG 5066 Operation at Higher Rates

It is clear that higher throughput will be available for the HF long-haul channel in near future (i.e, post 2000). What is not clear is the final form of the waveform standard or standards that will provide these data rates. Accordingly, this Annex focuses only on the generic requirements and guidance implementers of the STANAG 5066 Profile for HF Data Communications must follow when using waveforms at speeds higher than 2400 bps. Guidance is of the form of the maximum data rates for which an implementation must be sized, and on the identification of system configuration parameters and requirements that are dependent on data rates and will likely change as the result of implementing the protocol profile with higher-rate modems as support.

G.3.1 *Maximum Data Rate*

Implementations of STANAG 5066 should be designed to operate at maximum data rates on the rough order of 20 kilobits per second. As a 'rough-order' estimate, this recommended maximum data rate could be expected to vary by a factor of 2, larger or smaller, as the discussions presented here demonstrate.

Given foreseeable technology, dense signaling constellations operating within a 3kHz bandwidth at rates as high as 7 bits or 8 bits per hertz might be achievable in future, giving satisfactory performance, even for long-haul fading channels, for FEC codes at $\frac{1}{2}$ -rate, $\frac{3}{4}$ -rate, or even $\frac{7}{8}$ -rate. For example, current standardization proposals for high-speed HF waveforms include a proven implementation based on 64-QAM at 2400 baud operating in a nominal 3 kHz channel, with a $\frac{3}{4}$ -rate FEC code. This design yields a signaling rate of 10,800 bps. The inclusion of known data sequences in the waveform to support adaptive equalization algorithms for long-haul fading channels yields a final data rate of 9600 bps. Extension of such a design to 128-QAM might be achievable in future, and this would yield a user data rate on the order of 19.2 kbps.

Note that with more optimistic assumptions on the capabilities of future signal processing technology for HF transmission, the recommended maximum speed could have been established at an even higher value on the order of 38.4 kbps. This higher maximum data rate could also be achievable if wider bandwidths are assumed for the HF channel, or if channels are assumed with less stressing fading conditions than those that exist on the long-haul skywave channel. For example, wider bandwidth to 6 kHz can be achievable with independent side-band operation, though given the congestion in the HF band, this is not considered a likely prospect from a frequency-management perspective. Alternatively, channel coherence is known to be more stable for surface wave operation at HF where one signal path is non-fading. Signal design for surface channels can yield reliable and satisfactory performance in a waveform design where the number of known data symbols required for equalization in severe channels is reduced from that required for long-haul operation, allowing an increase in the available user data rate.

As a further note, the STANAG 5066 protocols could be used on 25-kHz (or other) LOS or SATCOM channels at UHF as a means of combating bursty interference rather than the channel fades characteristic of HF. ARQ protocols have been used on UHF-SATCOM channels to successfully combat unintentional burst interference from uncontrolled LOS terrestrial sites within the satellite coverage area. The STANAG 5066 protocols could be used in similar fashion but with LOS/SATCOM modems that offer higher data rates because of their greater bandwidth.

The recommended maximum data of 20 kilobits per second is therefore considered a reasonable design margin for implementation of STANAG 5066 for the foreseeable future.

G.3.2 Buffer Sizes

The required sizes for data stores and buffers within an STANAG 5066 HF subnetwork implementation depend on the expected arrival rate for traffic from both the local client, via the Subnetwork Interface, and remote clients, via the interface to the communications equipment. Subnetwork implementers will be aware that the buffer sizes they allocate will likely need to change when implementing to support the higher maximum data rate.

G.3.3 Data Rate Control Protocol

The Data Rate Control (DRC) Protocol in STANAG 5066 uses the Type 1 Engineering Orderwire (EOW) messages embeddable in D_PDUs to specify the desired or actual data rates of the sender and receiver on a link, as specified in Annex C.5.1.

Mandatory requirements currently specify field values that allow the DRC protocol to control data rates up to 9600 b/s. Unused field values will be assigned for data rates above 9600 should these become necessary. Proposed values are given in the following table, but may be changed in future to support the final waveform(s) ratified for high-speed HF transmission:

**Table G-1. Proposed Extensions to
Data Rate Parameter Message Type 1 EOW Message**

| MSB - LSB | Interpretation |
|----------------|----------------------|
| <i>0 0 0 0</i> | <i>As in Annex C</i> |
| <i>through</i> | <i>.....</i> |
| <i>1 0 1 1</i> | <i>As in Annex C</i> |
| 1 1 0 0 | 14400 bps |
| 1 1 0 1 | 16000 bps |
| 1 1 1 0 | 19200 bps |
| 1 1 1 1 | reserved |

Implementers of STANAG 5066 may wish to adopt the use of the proposed values for higher data rates for the Data-Rate-Parameter field of the Type 1 EOW message. If so, they are advised that the specific values may change in future editions of 5066, and their software should be based on site- and modem- configuration parameters that allow change in these values without recompilation of their software.

Likewise, the addition of new waveform STANAGs to support higher data rates may entail changes to the allowable choices for modem interleaver. This may also effect the interpretation of values for the Interleaver-Parameter field of the Type 1 EOW message. At present, however, there are no proposed changes to the Interleaver Parameter planned for future editions.

G.3.4 EOT Calculation

Support for higher data rates must be accounted for in calculation of the End-of-Transmission field required in each Data Transfer Sublayer PDU (D_PDU).

Calculation of the End-of-Transmission field required in each Data Transfer Sublayer PDU (D_PDU) depends on a number of factors:

- the size of the D_PDU,
- the sizes of D_PDUs that follow it in the transmission,
- the interleaver settings for the modem (because of fill-bits inserted by the modem when data does not completely fill an interleaver or coding block),
- and the data rate of the transmission.

Since compliant implementations of the STANAG 5066 protocols that also implement the Data Rate Change capability must factor the variable data rate into calculations of the EOT, support for a higher maximum data rate is not considered a major impact.

G.3.5 Waiting and Response Times

Waiting and response times for STANAG 5066 protocols that depend on the data rate of the supporting link will need to be determined with allowance for the greater range in data rates. As in the case of EOT calculations, waiting and response times may be modified during execution of the DRC protocol and resultant changes in the link data rate, and therefore support for higher data rate is not considered a major impact on the implementation in this area.

Annex H. Implementation Guide and Notes (information only)

This Annex contains rules and guidelines for adaptive speed control and other implementation issues based on NC3A experience with earlier systems. The best and most complete source of information on implementation topics is NC3A TM-937 "Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio".

H.1 Flow Control

Flow control imposed by a client on the Subnetwork could cause the receiving queues of the HF Node to be filled, which in turn could cause older queued data for other clients to be discarded or result in a temporary pause in accepting and acknowledging error free PDUs. This situation is not acceptable since a client, of even a low Rank, could, in principle, cause a deterioration of the service provided to the other clients connected to the Node. Implementers are encouraged to implement one of the following approaches:

- minimize or eliminate the use of flow control on the subnetwork-to-client (i.e., receive) interface;
- employ an implementation-dependent queuing strategy that allocates buffers separately for each client;
- allocate sufficient buffer space that overflow has a low-probability of occurrence.

H.2 Reasons for Data Transfer PDUs (D PDUs) with Different Rules

Annex C to this document defines a number of different D_PDUs which may be used to transfer data: normal data transfer (D_PDU types 0-3), expedited data (D_PDU types 4 and 5), management message (D_PDU type 6), and non-ARQ (types 7, 8, and 15). This section reviews the use of the different D_PDUs and the reasons for having them.

H.2.1 Normal Data Transfer

The normal data transfer D_PDUs are intended in most cases for use in the transfer of encapsulated U_PDUs. It is difficult to efficiently handle high priority data in the context of this type of D_PDU alone. When a PDU from a higher layer reaches the data transfer sublayer, it is segmented into a number of D_PDUs that are assigned sequence numbers. While it may be possible to "unqueue" or cancel parts of higher level PDUs, this will either result in the loss of data (cancellation) or delays in transmitting the high priority traffic. Thus, the additional types of D_PDUs have been introduced to efficiently accommodate high-priority traffic.

H.2.2 Expedited Data

Expedited data D_PDUs are intended for use to support subnetwork peer-to-peer communications (primarily making and breaking physical links), and exceptionally, to provide a path for U_PDUs of the highest priority which bypasses all existing queues. An example of the first would be if it were desired to immediately break a link that has long queues of traffic pending. If this "break" C_PDU was handled as a normal data PDU, it might (depending on the system implementation) have fairly long delays (minutes) before it could be transmitted. With the expedited data mechanism, the C_PDU bypasses all the normal data queues and is transmitted at the beginning of the next transmission interval.

Expedited data bypasses all existing normal data queues (which are maintained during the handling of the expedited data). A number of expedited data D_PDUs (corresponding to, and not more than, a single expedited data C_PDU) may be transmitted in a single transmission. A stop and wait protocol is used for the acknowledgement of (the group of) expedited data D_PDUs. This service is intended for occasional use to transmit small amounts of data. Frequent use, or use with large U_PDUs, will degrade system performance, and therefore the use of the expedited data modes for client traffic is strongly discouraged.

H.2.3 Management

A third type of service is provided exclusively for system management functions, for example, to coordinate the adaptive change of the HF modem data rate. This function, and others like it, requires a service with the smallest possible delay and with maximum robustness. This is provided by the management data D_PDU. Management D_PDUs bypass all pending data D_PDUs (when the system enters the management mode, both normal and expedited data queues are put on hold).

Transmission of the management D_PDU type follows a D_PDU-by-D_PDU stop-and-wait protocol, with D_PDUs repeated as necessary to fill the HF modem interleave buffers and provide maximum robustness and efficiency. Only a single Management (Type 6) D_PDU may be outstanding (unacknowledged) at any moment; the D_PDU is repeated until acknowledged (or the link fails because of a time out).

H.2.4 Non-ARQ

Non-ARQ, or unacknowledged, D_PDU types are provided in order to allow the transfer of data which does not require acknowledgement or cannot reasonably be acknowledged. This is useful for “broadcast” or “multicast” modes of operation, in which nodes are not allowed to transmit because of EMCON restrictions or where D_PDUs are addressed to multiple nodes. It is also required to support functions that occur when the node is not in an ARQ-processing state, such as the peer-to-peer communication by the Channel Access Sublayer when establishing a connection.

H.3 Other topics

Several short topics are addressed in the subsections below.

H.3.1 EOT Calculation

The definition of the End-of-Transmission (EOT) field in D_PDUs gives a maximum transmission interval of 127.5 seconds, or just over two minutes. When computing the value of the EOT, the results of the calculation must be rounded up to avoid collisions between node transmissions on half-duplex/single-frequency channels.

H.3.2 Error Detection for D_PDU Headers

Because the D_PDU header is generally shorter than the data, errors are more likely in the data part of a D_PDU than the header. Protecting the header with its own CRC allows the possibility to detect and use uncorrupted header information even if the data part of a D_PDU contains errors. For this reason, the added overhead of the CRC-on-header field was deemed warranted in the design of STANAG 5066.

H.3.3 DROP PDU Processing

If a D_PDU is to be sent with the DROP PDU bit set, it is inefficient to send the data portion of the D_PDU. However, all D_PDUs that make up the PDU to be dropped must be sent to maintain window synchronisation. A D_PDU that is received with the DROP PDU flag set must still be acknowledged.

H.3.4 Size of User Data Field

The 10-bit field SIZE OF USER DATA indicates the size of the information field in bytes. Its value does not include the size of the CRC-ON-SEGMENTED-C_PDU field. Note that full-size D_PDU transmissions can involve significant transmission time, as, for example, $2^{10}=1023$ bytes=8184 bits=109 seconds at 75 bps. A balance should be struck between protocol efficiency, which would encourage the use of large D_DPUs, and access and turnaround time, which would encourage small D_DPUs. An additional balancing factor in protocol efficiency must be weighed as well, since high-noise or fading environments will encourage the use of small (normally 100 to 400 byte) D_DPUs because of the increased likelihood of error (and therefore decreased throughput) with large D_PDUs. Subnetwork implementations that allow adaptation of the maximum D_PDU size to promote protocol efficiency in varying channel conditions are not necessarily precluded so long as they do not violate the mandatory requirements in this STANAG or prevent interoperability with subnetwork implementations that do not support such adaptation algorithms. This issue is addressed further in Section H.7

H.3.4 Forwarding of D_PDUs to the Channel Access Sublayer

To allow effective monitoring of channel activity by the Channel Access Sublayer, the Data Transfer Sublayer should deliver all identifiable *D_PDUs* (i.e. D_PDUs in which no errors were detected in the header) to the Channel Access Sublayer, even when they are *not addressed* to the receiving node. The Channel Access Sublayer filters the received C_PDUs according to address and type and, as appropriate, locally processes the C_PDU or passes the C_PDU to the Subnet Interface Sublayer.

In the non-ARQ modes of operation of the Data Transfer Sublayer, all identifiable *C_PDUs* (i.e. C_PDUs in which the header portion of one or more associated D_PDU's) was received without error are delivered to the Channel Access Sublayer. This procedure was adopted so that:

- i) the Channel Access Sublayer could effectively monitor the channel activity, and
- ii) a client could specify that only error free or, alternatively, that all identified S_PDUs be delivered.

The second option may be useful if the client handles printable text and/or implements additional error control functionality.

H.4 Synchronisation of the ARQ Machine

Establishment of a "connection" requires the initial synchronisation of the peer protocol ARQ machines in the Data Transfer Sublayer. This process can be viewed as automatic in the sense that the peer ARQ machines associated with a new connection (i.e. one for which an ARQ machine must be established) will be automatically reset, and valid numbers established for the upper and lower window edges at transmitter and receiver. The on-going assumption that the ARQ machines associated with a revived data state connection (i.e. one for which an existing ARQ machine is re-activated) remain in

synchronization over long periods of time may not be strong, however, because of node failure, undetected message error, or other anomaly. For these reasons, the procedures for synchronization of the ARQ machine on a reliable link were deemed sufficiently important for correct link operation to be defined as mandatory requirements for subnetwork operation. These synchronization procedures involve two steps:

- 1) on-going tests of ARQ machine operation through validation of the upper- and lower- window edge values in D_PDUs exchanged over the link (defined in Annex C section C.6.3) and
- 2) definition of the negotiated FULL-RESET ARQ resynchronization protocol using the RESET/WIN-RESYNC (Type 3) D_PDU (defined in Annex C section C.6.5).

Note that in addition to the negotiated FULL-RESET procedure of Annex C, the destination node can independently re-synchronise its receive LWE and UWE pointers to the indicated TX FRAME SEQ # *UWE* pointer of the originating node. This is equivalent to signaling a group ACK to all frames transmitted by the originating node. All synchronization options result in some loss of data although, in general, a negotiated re-synchronisation results in the loss of a smaller number of frames.

H.5 Use of Rank and Priority Arguments (Subnet Interface Sublayer)

The rank of subnet clients is used to determine the allocation of subnet resources. As an example, if a new client attempts to come on-line (bind to a node) but not enough resources are available, the Subnetwork Interface Sublayer is permitted to unilaterally declare a client with lower rank off-line in order to release resources for the higher-ranked client.

Rank is also used to determine management privileges. The node shall not accept command-type S_MANAGEMENT_MSG_REQUEST primitives from a client with rank less than 15. The node shall accept request-type S_MANAGEMENT_MSG_REQUEST primitives (which cannot change the configuration of the node or subnetwork) from any client.

Priority can take a value in the range 0-15. The node “does its best” to service high priority U_PDUs before lower priority U_PDUs which are queued in the system. This means that the node is not required to *guarantee* that the higher priority U_PDUs will overtake all queued lower priority U_PDUs (depending on the implementation of the node, it may not be possible for a higher priority U_PDU to overtake a queued lower priority U_PDU which has entered an advanced stage of processing).

Client rank and the priority of data submitted by that client are not necessarily dependent.

H.6 Implementation notes for Data Rate Change Procedure

Although many modems that support the STANAG 4285 waveform are implemented so that the transmit and receive data rates must be the same at any instant, this is not an absolute limitation. Some modems may be implemented so that they can use different transmit and receive data rates. Furthermore, the subnetwork, including the modem, may be implemented in such a way as to circumvent the problem if it exists, i.e., fast remote control of the modem (in a half-duplex system), or multiple modems (in a half or full duplex system). However, the DRC procedures defined in Annex C Section C.6.4.3 (Additional Scenarios) will support the case in which the system is limited by this constraint.

If a node uses D_PDU error statistics to make data rate adaptation decisions, the decision to change data rate should be made only based on a transmission containing DATA-ONLY or DATA_ACK D_PDUs; or after a transmission made up of only ACK_ONLY D_PDUs is received with errors. These conditions are imposed because it is difficult to make reliable DRC decisions based on analysis of short signals. While these conditions are not required for interoperability, they are required for reliable system operation. Experience with algorithms based on counting D_PDUs with errors shows that short transmissions are usually either received completely, with no errors, or not received at all (this is caused by the steep BER vs. E_b/N_o curves for the STANAG 4285 and MIL-110A modems). This can cause the data rate on an acknowledgement link, carrying only short transmissions, to increase steadily until, after one increase too many, the acknowledgements are no longer received. This is also the reason for transmitting ACK-ONLY D_PDUs multiple times when they are the only D_PDU type transmitted. This rule will generally lead to the ACK-ONLY D_PDUs being sent at a lower data rate, when data is flowing in (mainly) one direction on a link. In order to avoid the situation where the data transmissions are at 75 bps and the ACKs are at 2400 bps, the following rule should be followed: one end of the link shall transmit at not less than 1/8 the data rate of the other end.

If a node is only receiving acknowledgements, the transmit data rate for acks should be less than the transmit data rate for data but not less than 1/8 of the transmit data rate for data. The result of this rule will be that adaptive changes to data rate on one direction of the link will in some cases “pull” the data rate on the other direction of the link. This recommendation does not apply to a circuit that includes a shore answering frequency that must operate at a fixed data rate.

H.7 Optimum D_PDU Size

A number of studies have examined the impact of the size of D_PDUs (or HF frames) on the throughput of the ARQ protocol. Perhaps the most relevant of these studies to performance in the context of STANAG 5066 has been a study which combined on-the-air trials with OPNET modeling, done by DRA Portsmouth. Other studies have been done by SHAPE Technical Center (STC) and by Rohde and Schwartz.

The first work in this area was done by STC in 1992 and documented in “Laboratory and Field Tests of the High Frequency OSI Data Link Protocol”, STC TN-506, August 1993. Some of the major findings in this report are¹

1. the optimum frame size varies with the data rate and with channel conditions (fading as well as SNR)
2. throughput is not strongly sensitive to frame size
3. adaptive control of data rate with a compromise frame size of 200 bytes gives throughput very nearly identical to that realised with the “optimum” frame size for each data rate.

These results formed the basis of STC's, and later NC3A's, work with adaptive data rate HF ARQ protocols from 1992 until the present.

More recently, the DRA trials focused on the effect of varying error rate at 1200 bps. The DRA study has confirmed that throughput at a given BER is not strongly dependent on the frame size. At the higher BERs, there is a throughput maximum for frame sizes between 100 and 200 bytes. As the

¹ One result from this report which is not relevant to the subject at hand but is more relevant to adaptive data rate control, is the very small improvement in throughput gained from switching from 1200 to 600 bps (see figures 24 and 25 of NC3A TM-937).

BER decreases, the frame size for maximum throughput also increases but very slowly, and the curve becomes even flatter.

The DRA throughput data are also valid for non-ARQ use of STANAG 5066 D_PDUs. The DRA data show the error free data bytes received, rather than the overall throughput; the small amount of overhead due to the short ARQ transmissions on a half-duplex link is not included. Thus, the 200 byte frame size is also a “good compromise” for use with non-ARQ 5066 transmission.

The DRA trials have confirmed (at least for 1200 bps) the following:

1. that the throughput is not strongly sensitive to frame size;
2. that 200 bytes is a good a ‘compromise’ selection for frame size.

If there is a desire to vary frame size in some way, STANAG 5066 supports the use of variable frame sizes. While the data available to date suggest that the benefit may be marginal, it would be possible, for example, to associate a certain frame size with each data rate.

The variable frame size scheme mentioned in the preceding paragraph brings with it a number of implementation issues (as does any variable frame size). One of these issues is the fact that, when a data rate change is made, there will generally be some number of D_PDUs queued for transmission. It will, in general, be advisable to transmit these (at least up to the end of the next C_PDU) without changing the frame size and start using a new frame size when the next D_PDUs are created. This avoids some difficult synchronisation issues, and the data available suggest that the performance penalty will be negligible.

H.8 Application Note: Use of STANAG 5066 in Broadcast Transmission Modes

Classical use of HF has allocated a single circuit (i.e, a frequency or set of frequencies) to each broadcast-traffic source. Typically, multiple frequencies are allocated to the circuit to increase the HF coverage area. When broadcasts from multiple sources are supported, the techniques typically used to share the circuit is frequency-division multiplexing (FDM) or time-division multiplexing (TDM), with FDM typically preferred because of the low throughput (nominally 75 bps) supportable on long-haul HF circuits. Even when higher throughputs are available, and TDM is a viable approach for transmission, the fixed-capacity allocations of TDM and FDM techniques can lead to inefficient circuit utilization and poor flexibility.

With its capabilities for higher throughput and group-addressing of U_PDUs, STANAG 5066 can support a broadcast of traffic from multiple sources. The paragraphs below review some of the implementation and efficiency issues that arise in using these capabilities of STANAG 5066. Some of the differences between an approach based on 5066 broadcast modes and the traditional, physically multiplexed (either FDM or TDM) multi-channel approaches are described.

It is assumed in the discussions below that an HF node which is handling multiple broadcast clients will be dedicated to this task. It would seem in general inappropriate (although perhaps not inconceivable) to have this node abandon multiple broadcasts in order to do some other task.

In order to support multiple simultaneous clients of the same type and SAP ID attached to the subnetwork a multi-user subnetwork client is required. This client will sit between a number of clients and the subnetwork interface and perform a number of functions, including multiplexing and segmentation. The arrangement is shown in Figure H-5.

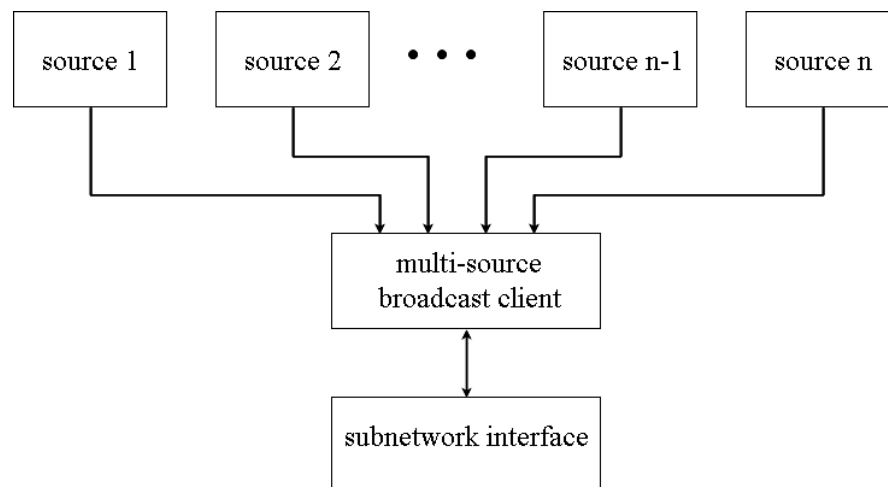


Figure H-5. Multi-source Broadcast (MSB) Client

Data will be accepted from the sources according to the interface spec of the source; the MSB client then forms a buffer between the standard interface to the 5066 subnetwork and the sources, which may be existing equipment.

Segmentation by the MSB client (or any other client) allows for more responsive and adaptive behaviour by the subnetwork. The client must segment large U_PDUs into segments (nominally 2 kb or smaller) and submit these segments in the appropriate primitive to the subnet interface sublayer. The reasoning behind the requirement for the client to segment large U_PDUs is provided below in conjunction with the summary of sublayer processing.

Insuring that each broadcast source gets appropriate access to the channel would seem to be a task appropriate to the channel access sublayer. However, since the MSB client is already responsible for queuing the data from the various sources, it knows the load that each source offers the subnetwork. The MSB client also knows the rank of the sources and the priority of the data. It therefore has all of the information required to allocate capacity to the various sources. Therefore, the MSB client should insure that U_PDUs or segments from multiple broadcast sources are passed down to the lower layers as is appropriate to

- the current (recent average) bandwidth of the source
- the rank of source
- the priority of the data from the source

At the receiving node, if a node is in more than one broadcast group, U_PDUs may (will!) arrive for various broadcast groups (perhaps composed by different broadcast compilers) and be delivered to the multi-source broadcast client. The MSB client will be able to identify the source of the arriving U_PDUs using the information in the S_UNIDATA_INDICATION primitives (sent to the client) which have the addressing information in them (S_PDUs do not). Exchanging this information between the Subnet Access Sublayer and Data Transfer Sublayer is an implementation issue (i.e., the definition of the internal primitives) which is not appropriate for a STANAG. Detailed guidance is available in NC3A TM-937.

Alternatively, the MSB client can use the Unreliable Datagram-Oriented Protocol (UDOP) defined in Annex F of this STANAG. UDOP supports source identification (through the connection identifier field of the UDOP PDU) and segmentation/re-assembly services (through the U_PDU segment identifier field of the UDOP). Yet another option would be the use of the Internet Protocol (IP) client type defined in Annex F of this STANAG in conjunction with the Internet User Datagram Protocol, since UDP/IP also supports source identification and segmentation and reassembly. Note that the subnetwork interface sublayer **does not segment** U_PDUs because re-assembly is not supported in the S_PDU format.

The channel access sublayer is simply a pipe for type 0 (data) S_PDUs, adding a few bits to convert them into type 0 (data) C_PDUs. The Data Transfer Sublayer will segment the arriving C_PDUs into D_PDUs and queue them for transmission. Note that, as noted in Annex H.7, the nominal optimum size of 200 bytes for HF transmission also applies to broadcast modes. So a C_PDU of 2 kb will be segmented into 10 D_PDUs, and queued for transmission (at 300 bps, these D_PDUs will take about 1 minute to transmit). The next C_PDU to arrive will be likewise segmented and queued.

With this summary of operation, the reason for limiting the size of U_PDUs, and thereby also S_PDUs, is clear. Very large U_PDUs, unsegmented by the user, can capture the channel. For example, if a 100 kb S_PDU (inside a C_PDU) were to arrive at the DT sublayer for ARQ transmission, then the DT sublayer would segment it into some 500 D_PDUs and queue them for transmission. If the transmit speed is 300 bps, it will be about 45 minutes before any other client has access to the channel. Depending on the implementation of the DT sublayer, this situation may or may not also arise if a large PDU arrives at the DT sublayer for non-ARQ service. Given that, even with data-rate adaptation, there is the potential for low-speed transmission over poor channels, the size of the maximum transmission unit (MTU) in the subnetwork should be made comparatively small to avoid channel-capture. MTU sizes on the order of 1500 bytes (comparable to Ethernet) and smaller should be used.

The addressing structure adopted for non-ARQ (type 7) D_PDU allows some 268 million different group addresses. This large address space would be reduced by adoption of hierarchical addressing schemes for the subnetwork, but even so remains quite large. There remains the possibility to have a destination node belong to a number of broadcast groups depending on its operational role. The large address space, combined with the ability to belong to more than one broadcast group, offers the potential for significant efficiencies in broadcasts, because a close match between a group and the intended audience of a message is more likely. When a node can belong to only one group, one is forced to trade off between very large groups, with members receiving large amounts of unwanted messages; or smaller groups, with messages being retransmitted to a number of different groups. This addressing scheme also allows a broadcast compiler to send a message to a single destination node, by simply using that node's individual address rather than a group address.

This approach is adaptive. If one of the broadcast sources is idle, or disconnects from the subnet, the capacity becomes available for other users (whereas in a TDMA or FDMA scheme it is more difficult or impossible to reallocate capacity). Because the allocation of capacity is done in software, it is flexible - it can take account of client rank, data priority, offered load, etc - and thereby use the HF channel more efficiently.

Annex H. Implementation Guide and Notes (information only)

This Annex contains rules and guidelines for adaptive speed control and other implementation issues based on NC3A experience with earlier systems. The best and most complete source of information on implementation topics is NC3A TM-937 "Open Systems for Radio Communications: A Subnet Architecture for Data Transmission over HF Radio".

H.1 Flow Control

Flow control imposed by a client on the Subnetwork could cause the receiving queues of the HF Node to be filled, which in turn could cause older queued data for other clients to be discarded or result in a temporary pause in accepting and acknowledging error free PDUs. This situation is not acceptable since a client, of even a low Rank, could, in principle, cause a deterioration of the service provided to the other clients connected to the Node. Implementers are encouraged to implement one of the following approaches:

- minimize or eliminate the use of flow control on the subnetwork-to-client (i.e., receive) interface;
- employ an implementation-dependent queuing strategy that allocates buffers separately for each client;
- allocate sufficient buffer space that overflow has a low-probability of occurrence.

H.2 Reasons for Data Transfer PDUs (D PDUs) with Different Rules

Annex C to this document defines a number of different D_PDUs which may be used to transfer data: normal data transfer (D_PDU types 0-3), expedited data (D_PDU types 4 and 5), management message (D_PDU type 6), and non-ARQ (types 7, 8, and 15). This section reviews the use of the different D_PDUs and the reasons for having them.

H.2.1 Normal Data Transfer

The normal data transfer D_PDUs are intended in most cases for use in the transfer of encapsulated U_PDUs. It is difficult to efficiently handle high priority data in the context of this type of D_PDU alone. When a PDU from a higher layer reaches the data transfer sublayer, it is segmented into a number of D_PDUs that are assigned sequence numbers. While it may be possible to "unqueue" or cancel parts of higher level PDUs, this will either result in the loss of data (cancellation) or delays in transmitting the high priority traffic. Thus, the additional types of D_PDUs have been introduced to efficiently accommodate high-priority traffic.

H.2.2 Expedited Data

Expedited data D_PDUs are intended for use to support subnetwork peer-to-peer communications (primarily making and breaking physical links), and exceptionally, to provide a path for U_PDUs of the highest priority which bypasses all existing queues. An example of the first would be if it were desired to immediately break a link that has long queues of traffic pending. If this "break" C_PDU was handled as a normal data PDU, it might (depending on the system implementation) have fairly long delays (minutes) before it could be transmitted. With the expedited data mechanism, the C_PDU bypasses all the normal data queues and is transmitted at the beginning of the next transmission interval.

Expedited data bypasses all existing normal data queues (which are maintained during the handling of the expedited data). A number of expedited data D_PDUs (corresponding to, and not more than, a single expedited data C_PDU) may be transmitted in a single transmission. A stop and wait protocol is used for the acknowledgement of (the group of) expedited data D_PDUs. This service is intended for occasional use to transmit small amounts of data. Frequent use, or use with large U_PDUs, will degrade system performance, and therefore the use of the expedited data modes for client traffic is strongly discouraged.

H.2.3 Management

A third type of service is provided exclusively for system management functions, for example, to coordinate the adaptive change of the HF modem data rate. This function, and others like it, requires a service with the smallest possible delay and with maximum robustness. This is provided by the management data D_PDU. Management D_PDUs bypass all pending data D_PDUs (when the system enters the management mode, both normal and expedited data queues are put on hold).

Transmission of the management D_PDU type follows a D_PDU-by-D_PDU stop-and-wait protocol, with D_PDUs repeated as necessary to fill the HF modem interleave buffers and provide maximum robustness and efficiency. Only a single Management (Type 6) D_PDU may be outstanding (unacknowledged) at any moment; the D_PDU is repeated until acknowledged (or the link fails because of a time out).

H.2.4 Non-ARQ

Non-ARQ, or unacknowledged, D_PDU types are provided in order to allow the transfer of data which does not require acknowledgement or cannot reasonably be acknowledged. This is useful for “broadcast” or “multicast” modes of operation, in which nodes are not allowed to transmit because of EMCON restrictions or where D_PDUs are addressed to multiple nodes. It is also required to support functions that occur when the node is not in an ARQ-processing state, such as the peer-to-peer communication by the Channel Access Sublayer when establishing a connection.

H.3 Other topics

Several short topics are addressed in the subsections below.

H.3.1 EOT Calculation

The definition of the End-of-Transmission (EOT) field in D_PDUs gives a maximum transmission interval of 127.5 seconds, or just over two minutes. When computing the value of the EOT, the results of the calculation must be rounded up to avoid collisions between node transmissions on half-duplex/single-frequency channels.

H.3.2 Error Detection for D_PDU Headers

Because the D_PDU header is generally shorter than the data, errors are more likely in the data part of a D_PDU than the header. Protecting the header with its own CRC allows the possibility to detect and use uncorrupted header information even if the data part of a D_PDU contains errors. For this reason, the added overhead of the CRC-on-header field was deemed warranted in the design of STANAG 5066.

H.3.3 DROP PDU Processing

If a D_PDU is to be sent with the DROP PDU bit set, it is inefficient to send the data portion of the D_PDU. However, all D_PDUs that make up the PDU to be dropped must be sent to maintain window synchronisation. A D_PDU that is received with the DROP PDU flag set must still be acknowledged.

H.3.4 Size of User Data Field

The 10-bit field SIZE OF USER DATA indicates the size of the information field in bytes. Its value does not include the size of the CRC-ON-SEGMENTED-C_PDU field. Note that full-size D_PDU transmissions can involve significant transmission time, as, for example, $2^{10}=1023$ bytes=8184 bits=109 seconds at 75 bps. A balance should be struck between protocol efficiency, which would encourage the use of large D_DPUs, and access and turnaround time, which would encourage small D_DPUs. An additional balancing factor in protocol efficiency must be weighed as well, since high-noise or fading environments will encourage the use of small (normally 100 to 400 byte) D_DPUs because of the increased likelihood of error (and therefore decreased throughput) with large D_PDUs. Subnetwork implementations that allow adaptation of the maximum D_PDU size to promote protocol efficiency in varying channel conditions are not necessarily precluded so long as they do not violate the mandatory requirements in this STANAG or prevent interoperability with subnetwork implementations that do not support such adaptation algorithms. This issue is addressed further in Section H.7

H.3.4 Forwarding of D_PDUs to the Channel Access Sublayer

To allow effective monitoring of channel activity by the Channel Access Sublayer, the Data Transfer Sublayer should deliver all identifiable *D_PDUs* (i.e. D_PDUs in which no errors were detected in the header) to the Channel Access Sublayer, even when they are *not addressed* to the receiving node. The Channel Access Sublayer filters the received C_PDUs according to address and type and, as appropriate, locally processes the C_PDU or passes the C_PDU to the Subnet Interface Sublayer.

In the non-ARQ modes of operation of the Data Transfer Sublayer, all identifiable *C_PDUs* (i.e. C_PDUs in which the header portion of one or more associated D_PDU's) was received without error are delivered to the Channel Access Sublayer. This procedure was adopted so that:

- i) the Channel Access Sublayer could effectively monitor the channel activity, and
- ii) a client could specify that only error free or, alternatively, that all identified S_PDUs be delivered.

The second option may be useful if the client handles printable text and/or implements additional error control functionality.

H.4 Synchronisation of the ARQ Machine

Establishment of a "connection" requires the initial synchronisation of the peer protocol ARQ machines in the Data Transfer Sublayer. This process can be viewed as automatic in the sense that the peer ARQ machines associated with a new connection (i.e. one for which an ARQ machine must be established) will be automatically reset, and valid numbers established for the upper and lower window edges at transmitter and receiver. The on-going assumption that the ARQ machines associated with a revived data state connection (i.e. one for which an existing ARQ machine is re-activated) remain in

synchronization over long periods of time may not be strong, however, because of node failure, undetected message error, or other anomaly. For these reasons, the procedures for synchronization of the ARQ machine on a reliable link were deemed sufficiently important for correct link operation to be defined as mandatory requirements for subnetwork operation. These synchronization procedures involve two steps:

- 1) on-going tests of ARQ machine operation through validation of the upper- and lower- window edge values in D_PDUs exchanged over the link (defined in Annex C section C.6.3) and
- 2) definition of the negotiated FULL-RESET ARQ resynchronization protocol using the RESET/WIN-RESYNC (Type 3) D_PDU (defined in Annex C section C.6.5).

Note that in addition to the negotiated FULL-RESET procedure of Annex C, the destination node can independently re-synchronise its receive LWE and UWE pointers to the indicated TX FRAME SEQ # UWE pointer of the originating node. This is equivalent to signaling a group ACK to all frames transmitted by the originating node. All synchronization options result in some loss of data although, in general, a negotiated re-synchronisation results in the loss of a smaller number of frames.

H.5 Use of Rank and Priority Arguments (Subnet Interface Sublayer)

The rank of subnet clients is used to determine the allocation of subnet resources. As an example, if a new client attempts to come on-line (bind to a node) but not enough resources are available, the Subnetwork Interface Sublayer is permitted to unilaterally declare a client with lower rank off-line in order to release resources for the higher-ranked client.

Rank is also used to determine management privileges. The node shall not accept command-type S_MANAGEMENT_MSG_REQUEST primitives from a client with rank less than 15. The node shall accept request-type S_MANAGEMENT_MSG_REQUEST primitives (which cannot change the configuration of the node or subnetwork) from any client.

Priority can take a value in the range 0-15. The node “does its best” to service high priority U_PDUs before lower priority U_PDUs which are queued in the system. This means that the node is not required to *guarantee* that the higher priority U_PDUs will overtake all queued lower priority U_PDUs (depending on the implementation of the node, it may not be possible for a higher priority U_PDU to overtake a queued lower priority U_PDU which has entered an advanced stage of processing).

Client rank and the priority of data submitted by that client are not necessarily dependent.

H.6 Implementation notes for Data Rate Change Procedure

Although many modems that support the STANAG 4285 waveform are implemented so that the transmit and receive data rates must be the same at any instant, this is not an absolute limitation. Some modems may be implemented so that they can use different transmit and receive data rates. Furthermore, the subnetwork, including the modem, may be implemented in such a way as to circumvent the problem if it exists, i.e., fast remote control of the modem (in a half-duplex system), or multiple modems (in a half or full duplex system). However, the DRC procedures defined in Annex C Section C.6.4.3 (Additional Scenarios) will support the case in which the system is limited by this constraint.

If a node uses D_PDU error statistics to make data rate adaptation decisions, the decision to change data rate should be made only based on a transmission containing DATA-ONLY or DATA_ACK D_PDUs; or after a transmission made up of only ACK_ONLY D_PDUs is received with errors. These conditions are imposed because it is difficult to make reliable DRC decisions based on analysis of short signals. While these conditions are not required for interoperability, they are required for reliable system operation. Experience with algorithms based on counting D_PDUs with errors shows that short transmissions are usually either received completely, with no errors, or not received at all (this is caused by the steep BER vs. E_b/N_o curves for the STANAG 4285 and MIL-110A modems). This can cause the data rate on an acknowledgement link, carrying only short transmissions, to increase steadily until, after one increase too many, the acknowledgements are no longer received. This is also the reason for transmitting ACK-ONLY D_PDUs multiple times when they are the only D_PDU type transmitted. This rule will generally lead to the ACK-ONLY D_PDUs being sent at a lower data rate, when data is flowing in (mainly) one direction on a link. In order to avoid the situation where the data transmissions are at 75 bps and the ACKs are at 2400 bps, the following rule should be followed: one end of the link shall transmit at not less than 1/8 the data rate of the other end.

If a node is only receiving acknowledgements, the transmit data rate for acks should be less than the transmit data rate for data but not less than 1/8 of the transmit data rate for data. The result of this rule will be that adaptive changes to data rate on one direction of the link will in some cases “pull” the data rate on the other direction of the link. This recommendation does not apply to a circuit that includes a shore answering frequency that must operate at a fixed data rate.

H.7 Optimum D_PDU Size

A number of studies have examined the impact of the size of D_PDUs (or HF frames) on the throughput of the ARQ protocol. Perhaps the most relevant of these studies to performance in the context of STANAG 5066 has been a study which combined on-the-air trials with OPNET modeling, done by DRA Portsmouth. Other studies have been done by SHAPE Technical Center (STC) and by Rohde and Schwartz.

The first work in this area was done by STC in 1992 and documented in “Laboratory and Field Tests of the High Frequency OSI Data Link Protocol”, STC TN-506, August 1993. Some of the major findings in this report are¹

1. the optimum frame size varies with the data rate and with channel conditions (fading as well as SNR)
2. throughput is not strongly sensitive to frame size
3. adaptive control of data rate with a compromise frame size of 200 bytes gives throughput very nearly identical to that realised with the “optimum” frame size for each data rate.

These results formed the basis of STC's, and later NC3A's, work with adaptive data rate HF ARQ protocols from 1992 until the present.

More recently, the DRA trials focused on the effect of varying error rate at 1200 bps. The DRA study has confirmed that throughput at a given BER is not strongly dependent on the frame size. At the higher BERs, there is a throughput maximum for frame sizes between 100 and 200 bytes. As the

¹ One result from this report which is not relevant to the subject at hand but is more relevant to adaptive data rate control, is the very small improvement in throughput gained from switching from 1200 to 600 bps (see figures 24 and 25 of NC3A TM-937).

BER decreases, the frame size for maximum throughput also increases but very slowly, and the curve becomes even flatter.

The DRA throughput data are also valid for non-ARQ use of STANAG 5066 D_PDUs. The DRA data show the error free data bytes received, rather than the overall throughput; the small amount of overhead due to the short ARQ transmissions on a half-duplex link is not included. Thus, the 200 byte frame size is also a “good compromise” for use with non-ARQ 5066 transmission.

The DRA trials have confirmed (at least for 1200 bps) the following:

1. that the throughput is not strongly sensitive to frame size;
2. that 200 bytes is a good a ‘compromise’ selection for frame size.

If there is a desire to vary frame size in some way, STANAG 5066 supports the use of variable frame sizes. While the data available to date suggest that the benefit may be marginal, it would be possible, for example, to associate a certain frame size with each data rate.

The variable frame size scheme mentioned in the preceding paragraph brings with it a number of implementation issues (as does any variable frame size). One of these issues is the fact that, when a data rate change is made, there will generally be some number of D_PDUs queued for transmission. It will, in general, be advisable to transmit these (at least up to the end of the next C_PDU) without changing the frame size and start using a new frame size when the next D_PDUs are created. This avoids some difficult synchronisation issues, and the data available suggest that the performance penalty will be negligible.

H.8 Application Note: Use of STANAG 5066 in Broadcast Transmission Modes

Classical use of HF has allocated a single circuit (i.e, a frequency or set of frequencies) to each broadcast-traffic source. Typically, multiple frequencies are allocated to the circuit to increase the HF coverage area. When broadcasts from multiple sources are supported, the techniques typically used to share the circuit is frequency-division multiplexing (FDM) or time-division multiplexing (TDM), with FDM typically preferred because of the low throughput (nominally 75 bps) supportable on long-haul HF circuits. Even when higher throughputs are available, and TDM is a viable approach for transmission, the fixed-capacity allocations of TDM and FDM techniques can lead to inefficient circuit utilization and poor flexibility.

With its capabilities for higher throughput and group-addressing of U_PDUs, STANAG 5066 can support a broadcast of traffic from multiple sources. The paragraphs below review some of the implementation and efficiency issues that arise in using these capabilities of STANAG 5066. Some of the differences between an approach based on 5066 broadcast modes and the traditional, physically multiplexed (either FDM or TDM) multi-channel approaches are described.

It is assumed in the discussions below that an HF node which is handling multiple broadcast clients will be dedicated to this task. It would seem in general inappropriate (although perhaps not inconceivable) to have this node abandon multiple broadcasts in order to do some other task.

In order to support multiple simultaneous clients of the same type and SAP ID attached to the subnetwork a multi-user subnetwork client is required. This client will sit between a number of clients and the subnetwork interface and perform a number of functions, including multiplexing and segmentation. The arrangement is shown in Figure H-5.

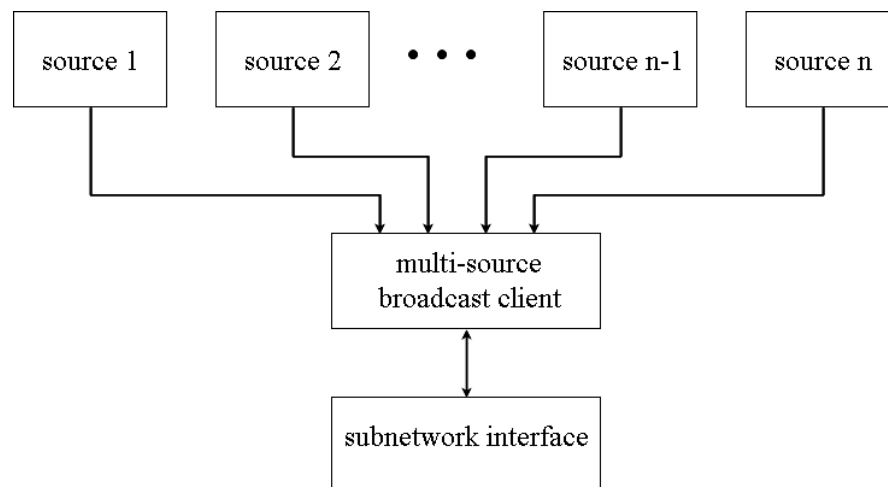


Figure H-5. Multi-source Broadcast (MSB) Client

Data will be accepted from the sources according to the interface spec of the source; the MSB client then forms a buffer between the standard interface to the 5066 subnetwork and the sources, which may be existing equipment.

Segmentation by the MSB client (or any other client) allows for more responsive and adaptive behaviour by the subnetwork. The client must segment large U_PDUs into segments (nominally 2 kb or smaller) and submit these segments in the appropriate primitive to the subnet interface sublayer. The reasoning behind the requirement for the client to segment large U_PDUs is provided below in conjunction with the summary of sublayer processing.

Insuring that each broadcast source gets appropriate access to the channel would seem to be a task appropriate to the channel access sublayer. However, since the MSB client is already responsible for queuing the data from the various sources, it knows the load that each source offers the subnetwork. The MSB client also knows the rank of the sources and the priority of the data. It therefore has all of the information required to allocate capacity to the various sources. Therefore, the MSB client should insure that U_PDUs or segments from multiple broadcast sources are passed down to the lower layers as is appropriate to

- the current (recent average) bandwidth of the source
- the rank of source
- the priority of the data from the source

At the receiving node, if a node is in more than one broadcast group, U_PDUs may (will!) arrive for various broadcast groups (perhaps composed by different broadcast compilers) and be delivered to the multi-source broadcast client. The MSB client will be able to identify the source of the arriving U_PDUs using the information in the S_UNIDATA_INDICATION primitives (sent to the client) which have the addressing information in them (S_PDUs do not). Exchanging this information between the Subnet Access Sublayer and Data Transfer Sublayer is an implementation issue (i.e., the definition of the internal primitives) which is not appropriate for a STANAG. Detailed guidance is available in NC3A TM-937.

Alternatively, the MSB client can use the Unreliable Datagram-Oriented Protocol (UDOP) defined in Annex F of this STANAG. UDOP supports source identification (through the connection identifier field of the UDOP PDU) and segmentation/re-assembly services (through the U_PDU segment identifier field of the UDOP). Yet another option would be the use of the Internet Protocol (IP) client type defined in Annex F of this STANAG in conjunction with the Internet User Datagram Protocol, since UDP/IP also supports source identification and segmentation and reassembly. Note that the subnetwork interface sublayer **does not segment** U_PDUs because re-assembly is not supported in the S_PDU format.

The channel access sublayer is simply a pipe for type 0 (data) S_PDUs, adding a few bits to convert them into type 0 (data) C_PDUs. The Data Transfer Sublayer will segment the arriving C_PDUs into D_PDUs and queue them for transmission. Note that, as noted in Annex H.7, the nominal optimum size of 200 bytes for HF transmission also applies to broadcast modes. So a C_PDU of 2 kb will be segmented into 10 D_PDUs, and queued for transmission (at 300 bps, these D_PDUs will take about 1 minute to transmit). The next C_PDU to arrive will be likewise segmented and queued.

With this summary of operation, the reason for limiting the size of U_PDUs, and thereby also S_PDUs, is clear. Very large U_PDUs, unsegmented by the user, can capture the channel. For example, if a 100 kb S_PDU (inside a C_PDU) were to arrive at the DT sublayer for ARQ transmission, then the DT sublayer would segment it into some 500 D_PDUs and queue them for transmission. If the transmit speed is 300 bps, it will be about 45 minutes before any other client has access to the channel. Depending on the implementation of the DT sublayer, this situation may or may not also arise if a large PDU arrives at the DT sublayer for non-ARQ service. Given that, even with data-rate adaptation, there is the potential for low-speed transmission over poor channels, the size of the maximum transmission unit (MTU) in the subnetwork should be made comparatively small to avoid channel-capture. MTU sizes on the order of 1500 bytes (comparable to Ethernet) and smaller should be used.

The addressing structure adopted for non-ARQ (type 7) D_PDU allows some 268 million different group addresses. This large address space would be reduced by adoption of hierarchical addressing schemes for the subnetwork, but even so remains quite large. There remains the possibility to have a destination node belong to a number of broadcast groups depending on its operational role. The large address space, combined with the ability to belong to more than one broadcast group, offers the potential for significant efficiencies in broadcasts, because a close match between a group and the intended audience of a message is more likely. When a node can belong to only one group, one is forced to trade off between very large groups, with members receiving large amounts of unwanted messages; or smaller groups, with messages being retransmitted to a number of different groups. This addressing scheme also allows a broadcast compiler to send a message to a single destination node, by simply using that node's individual address rather than a group address.

This approach is adaptive. If one of the broadcast sources is idle, or disconnects from the subnet, the capacity becomes available for other users (whereas in a TDMA or FDMA scheme it is more difficult or impossible to reallocate capacity). Because the allocation of capacity is done in software, it is flexible - it can take account of client rank, data priority, offered load, etc - and thereby use the HF channel more efficiently.

Annex I: Messages and Procedures for Frequency Change (information only)

This annex defines protocol data units (PDUs) and procedures to allow the use of automatic link establishment (ALE) systems to select a new operating frequency. It could also be used to support manual frequency change; however the required coordination procedures are not defined here.

For some systems it may be desirable to adapt the data rate along with the frequency change. Three cases may be considered:

1. Leave data rate at current settings
2. Return to default data rate
3. Set to new values

Initially, a simple procedure that implemented case 1 or 2 was envisioned. However, it turned out that the “simple procedure” was so similar to the DRC procedure defined in Annex C that it seemed overall a good idea to extend the DRC PDUs and procedures to the ALM procedure. This would cover all three cases above and is defined as option 1 below; the “simple procedure” is defined in option 2 below. Only one will be present in the final version of the STANAG.

I.1 Option 1: Combined Frequency and Data Rate Change (Automatic Link Maintenance - ALM)

This section defines PDUs and procedures for automatic link maintenance that combines data rate and operating frequency changes. This is an extension to the DRC PDUs and procedures defined in Annex C. Table I-1 defines the additional MANAGEMENT message types that are used to implement this function.

Table I-1. ALM MANAGEMENT Messages

| Message Type | Function | Contents |
|---------------------|--|---|
| 5 | Automatic Link Maintenance Request (ALM_Req) | New HF modem transmit data rate and interleaving setting for ALM master |
| 6 | Automatic Link Maintenance Response (ALM_Resp) | Positive or negative response (including reason if negative) |

The format and contents of the type 5 message shall be as defined for the type 1 message in C.3.8. The format and contents of the type 6 message will be as defined for the type 2 message in C.3.8, with a new reason as shown in table I-2.

Since the procedures defined in this Annex are extensions of the Annex C DRC procedures, they allow either

1. the ALM master tx parameters may be changed and the rx parameters left unchanged
2. the tx and rx parameters can be changed to identical new values

As the procedures stand now, there is no capability to specify during ALM new tx and rx parameters that are asymmetric. Another change that should be considered is to add a data rate parameter

(extension of table C-7) which explicitly indicates that the parameters on the new frequency will be determined externally (i.e., policy, or an ALE system).

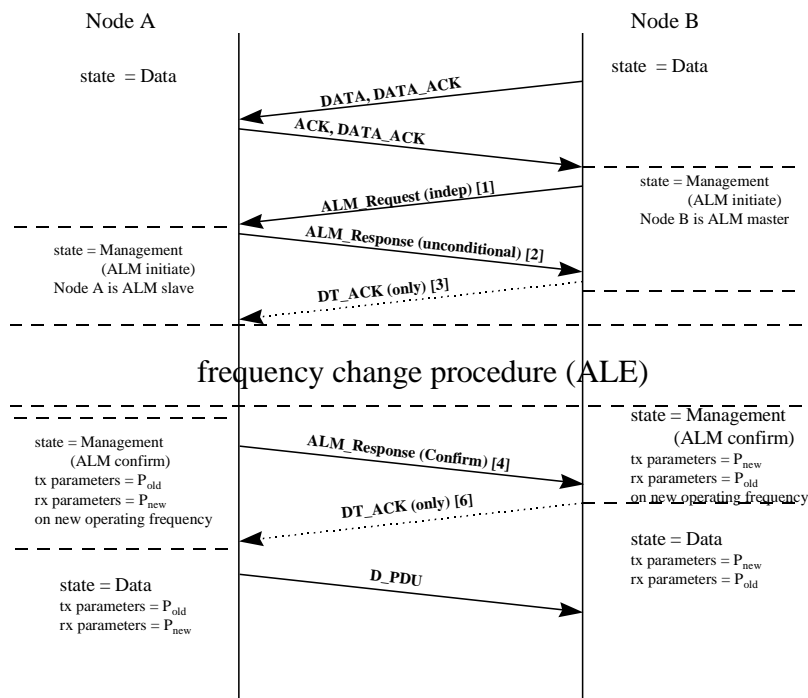
Table I-2. Contents for Type 6 Message (Reason)

| MSB - LSB | Interpretation |
|-----------|--|
| 0 0 0 0 0 | no reason (used to indicate unconditional acceptance of ALM_Request) |
| 0 0 0 0 1 | Tx and Rx parameters must be the same (conditionally accept) |
| 0 0 0 1 0 | Not possible to change modem data rate |
| 0 0 0 1 1 | Not possible to change modem interleaving |
| 0 0 1 0 0 | Not possible to change modem data rate or interleaving |
| 0 0 1 0 1 | Not consistent with local conditions |
| 0 0 1 1 0 | Not possible to change frequency |

Procedures

Following a decision to change frequency, a node shall use type 6 D_PDUs containing type 5 and type 6 MANAGEMENT messages to coordinate the change. The node initiating the frequency change is referred to as the ALM master for this ALM procedure. The data rate and interleaving fields shall carry the data rate and interleaving for the ALM Confirm phase of the procedure (see Figure I-1). The data rate and interleave parameters to be used on the new frequency may be selected based on some external information (i.e., a sounding or ALE system) or by policy (i.e., existing or default parameters shall be used on the new frequency).

An advisory EOW message has not been defined which requests a frequency change, because the need to change frequencies will generally be caused by link conditions which have deteriorated below some threshold, and active measures are more appropriate than advisory measures in that situation.



**Figure I-1: Automatic Link Maintenance Procedure
(Data Rate and Frequency Change)**

The node initiating the ALM procedure by sending a ALM Request (type 5) management message (shown at [1] in figure I-1) will be referred to as an “ALM master” (node B in figure I-1). When a node recognizes a MANAGEMENT D_PDU addressed to it, containing a ALM_Request message, the node (referred to as the ALM slave, node A in Figure I-1) shall transition to the management state. The ALM slave shall respond to the ALM_Request D_PDU with a ALM_Response (type 6) message (shown at [2] in Figure I-1). The ALM Response message shall indicate either “accept” or “refuse”, in accordance with Table I-2. If the ALM slave accepts the ALM_Request, the “reason” field shall indicate either “unconditional acceptance” or “Tx and Rx parameters must be the same”. If the ALM slave refuses the request, the reason field shall indicate the reason for the refusal. Only the five reasons defined in the table are valid reasons for refusing a ALM_Request.

The figure shows an example in which the modem at the ALM slave also has independent transmit and receive data rate.

In order to increase reliability, the ALM message should be repeated. Table I-3 gives a minimum suggested number of times that a message should be transmitted, based on minimizing the use of stuff bits in the interleaver. Other considerations could make a larger number of repetitions desirable.

Table I-3. Suggested minimum number of ALM messages to be transmitted at various data rates using STANAG 4285 modem

| Data rate | repetitions (short interleave) | repetitions (long interleave) |
|-----------|--------------------------------------|----------------------------------|
| 75 | 1 | 9 |
| 150 | 1 | 18 |
| 300 | 1 | 37 |
| 600 | 3 | 75 |
| 1200 | 7 | 150 |
| 2400 | 15 | 300 |

The number of retransmissions is selected to (nearly) fill the modem interleave buffer. For waveforms and interleaver settings not shown, the number of repetitions should be selected as required to minimize the use of “stuff bits” to fill the modem interleave buffer.

After receiving the ALM_Response message the ALM master shall review its contents and determine the appropriate response [4]. The various ALM_Response messages, and the allowed responses from the ALM master, are shown in table I-4.

Table I-4. Possible ALM_Response and Allowed ALM Master Actions

| ALM_Response | ALM_Response reason | allowed from ALM master |
|--------------|--|--|
| accept | unconditional | DT_ACK only |
| accept | transmit and receive parameters must be the same | DT_ACK only, or ALM_Response (cancel, or ALM Request) ^{note 1} |
| refuse | not possible to change modem data rate | ALM_Response (cancel) ^{note 2} or ALM_Request ^{note 3} (with DT_ACK) |
| refuse | not possible to change modem interleave | ALM_Response (cancel) ^{note 2} or ALM_Request ^{note 4} (with DT_ACK) |
| refuse | not possible to change modem data rate or interleave | ALM_Response (cancel) ^{note 2} (with DT_ACK) |
| refuse | not consistent with local conditions (see note 5) | ALM_Response (cancel) ^{note 2} or ALM_Request ^{note 6} (with DT_ACK) |

Notes to Table:

1. If EOW messages have been sent before the ALM procedure is initiated, the ALM master should already know that the ALM slave's transmit and receive parameters must be the same. Therefore, the ALM master should generally reply with a DT_ACK, accepting that the new parameters will apply to both transmit and receive.
2. ALM Slave shall acknowledge the cancel message with DT_ACK only; then the ALM procedure is discontinued. Note that this situation may frequently lead to failure of the link.
3. ALM_Request may be sent by master to request a different interleave setting at the same data rate.

4. ALM_Request may be sent by master to request a different data rate setting at the same interleave.
5. This reply shall only be sent in response to a request for a less robust set of parameters, i.e., higher data rate and/or shorter interleave than currently in use. It is expected that this will
6. ALM_Request may be sent by master to request different modem parameters that may be consistent with the local conditions.

In the table above, the DT_ACK refers to a data transfer sublayer acknowledgement of the preceding MANAGEMENT message (shown at [3] in Figure I-1). The DT_ACK reply indicates that the node has nothing further to communicate. If the DT_ACK (with no further management message) is sent in reply to a ALM_Response “accept” (as shown in Figure I-1), the nodes initiate the frequency change procedure, which may be controlled by an ALE system. Following the completion of the frequency change procedure, the nodes proceed to the “confirmation” phase. The ALM slave shall NOT initiate the frequency change procedure until it has received the DT_ACK (with no further management message) from the ALM master. If the DT_ACK (with no further management message) is sent by the ALM slave in reply to a ALM_Response “cancel”, both nodes abandon the procedure and return to the prior state. If node A (formerly the ALM slave) has no queued data or acknowledgements to send to node B, it shall send a data D_PDU, expedited data D_PDU, or non-ARQ D_PDU, with zero data attached.

In the figure, the slave’s ALM_Response with an “accept/unconditional” message generates the allowed DT_ACK from the ALM master.

After sending the DT_ACK [3], the ALM master initiates the frequency change procedure. On completion of the frequency change procedure, the ALM master sets its modem parameters and waits to receive a ALM Confirm message (type 6 MANAGEMENT message with response set to “confirm” and reason set to “none”) from node A (“confirmation phase”).

After receiving the DT_ACK [3], the ALM slave changes its modem parameters and transmits a ALM Confirm message [4] to the master. On receiving the ALM Confirm message, the master shall respond with a DT_ACK and then return to the previous state. After sending the ALM Confirm message [4] to the master and receiving the DT_ACK from the master, the slave shall return to the previous state and send any queued D_PDUs to node B. If node A (formerly the ALM slave) has no queued data to send to node B, it shall send a data D_PDU or expedited data D_PDU with zero data attached.

I.2 Option 2: Frequency Change without Data Rate Change

This Annex defines a data transfer sublayer procedure that will support a pause in data transfer for purposes of link management involving a change of frequency. The procedures for selecting the new frequency are not defined here; this is included for the benefit of systems that make use of some form of ALE for that purpose.

The procedures defined here make use of additional management messages (type 6 D_PDUs).

Table I-1. MANAGEMENT Message Types

| Message Type | Function | Contents |
|--------------|-------------------------------------|--|
| 5 | Frequency Change Request (FC_Req) | |
| 6 | Frequency Change Response (FC_Resp) | Positive or negative response (including reason if negative) |



Figure I-1 (a). Message Type 5 Format



Figure I-1 (b). Message Type 6 Format

Table I-2. Contents for Type 6 Message (Response)

| MSB - LSB | Interpretation |
|-----------|----------------|
| 0 0 0 | accept |
| 0 0 1 | refuse |
| 0 1 0 | cancel |
| 0 1 1 | confirm |

Procedures

Following a decision to change frequency, a node shall use type 6 D_PDUs containing type 5 and type 6 MANAGEMENT messages to coordinate the change. The node initiating the frequency change is referred to as the frequency change master (FC master) for this FC procedure.

An advisory EOW message has not been defined to which requests a frequency change, because the need to change frequencies will generally be caused by link conditions which have deteriorated below some threshold, and active measures are more appropriate than advisory measures in that situation.

Table I-2. Possible FC_Responses and Allowed FC Master Actions

| FC_Response | allowed from FC master |
|-------------|--|
| accept | DT_ACK only |
| refuse | ALM_Response (cancel) ^{note1} |

Notes to Table:

1. FC Slave shall acknowledge the cancel message with DT_ACK only; then the FC procedure is discontinued (as shown in Figure I-3).

In the table above, the DT_ACK refers to a data transfer sublayer acknowledgement of the preceding MANAGEMENT message (shown at [3] in Figure I-2). The DT_ACK reply indicates that the node has nothing further to communicate. If the DT_ACK (with no further management message) is sent in reply to a FC_Response “accept” (as shown in Figure I-2), the nodes switch to ALE mode in order to select a new frequency for the connection. When ALE is completed, the nodes enter the “confirmation” phase to complete the procedure.

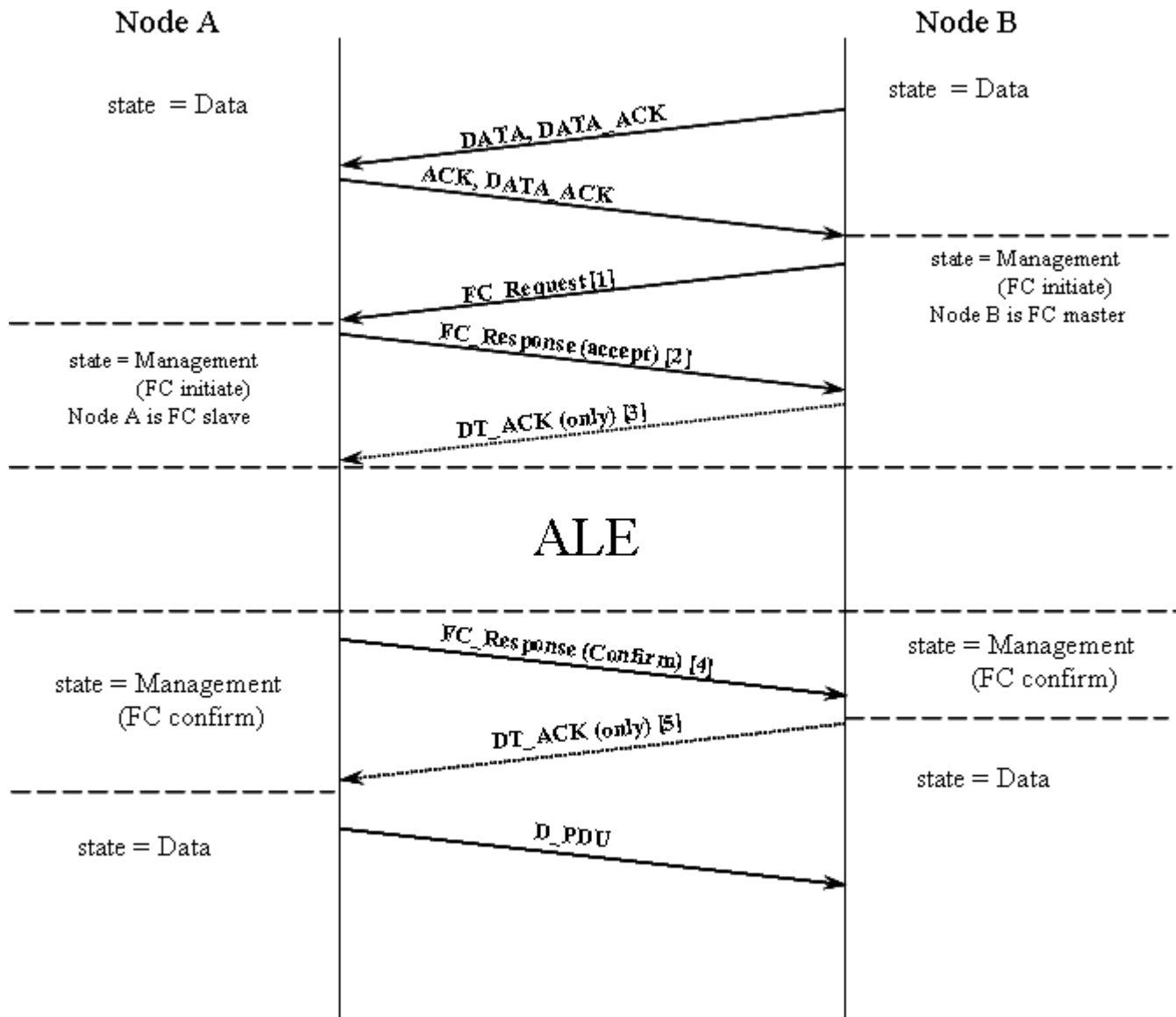


Figure I-2. Frequency Change Procedure (Example 1)

Figure I-2 presents a “normal”, successful frequency change procedure. In this diagram, node B determines that a frequency change is needed. Node B then enters the management state and sends an *FC_Request* message to node A (shown at [1] in Figure I-2). On receiving this message, node A enters the management state and sends an *FC_Response(accept)* message (shown at [2]). Node B responds with a *DT_ACK* (shown at [3]) and enters the ALE state as the active party. Node A enters the ALE state as the passive party, i.e., expecting to receive from node B. On completion of the ALE procedure, node A sends a *FC_Response(confirm)* message (shown at [4]). Node B responds with a *DT_ACK* (shown at [5]), unless node B wishes to initiate a ALM procedure.

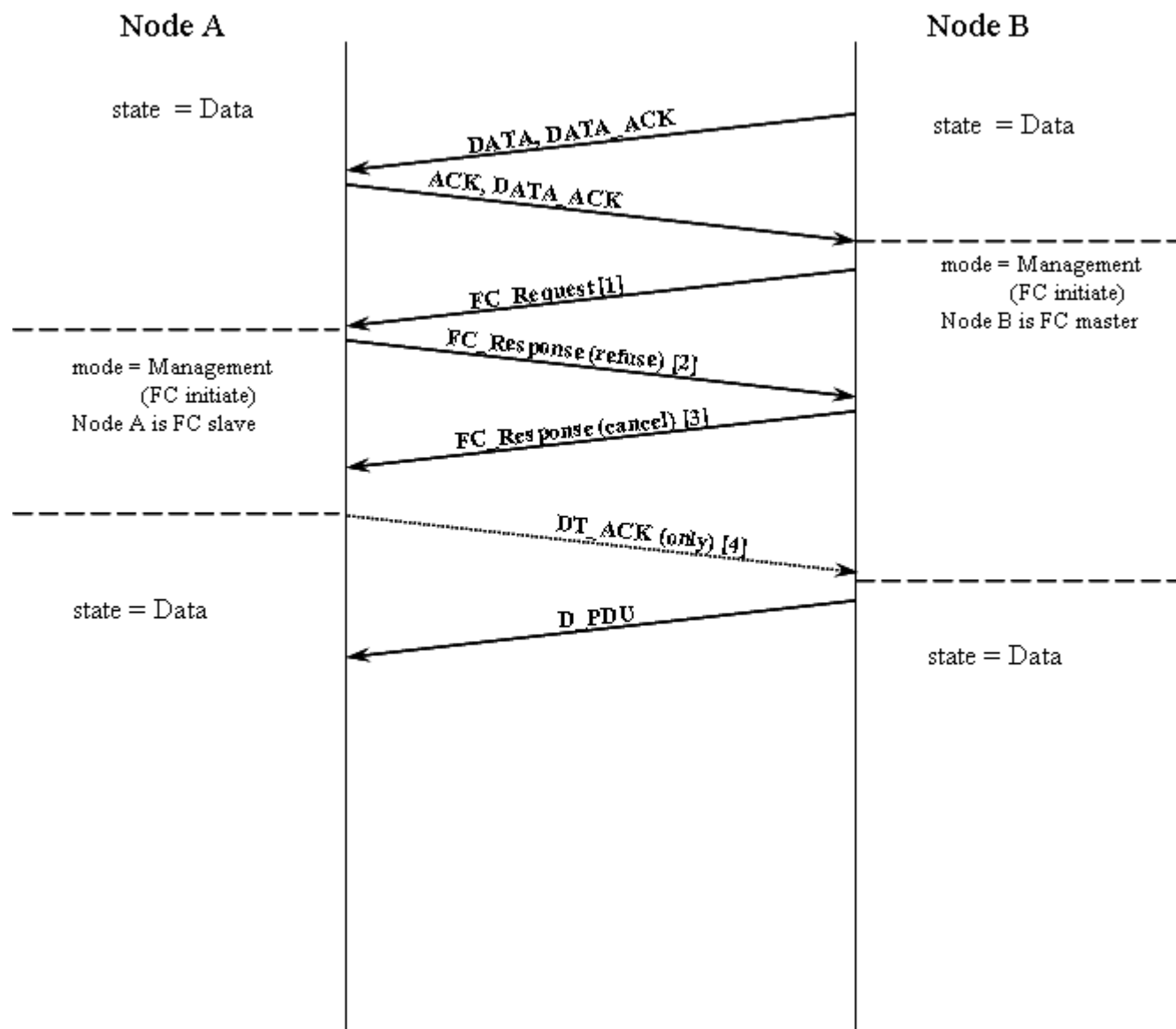


Figure I-3. Frequency Change Procedure (Example 2)

Left blank intentionally

ANNEX J - GENERAL REQUIREMENTS FOR ENHANCED MEDIA-ACCESS-CONTROL (MAC) CAPABILITIES IN MULTI-NODE STANAG 5066 NETWORKS (INFORMATIVE)

J.1 INTRODUCTION

The channel-access control capability of STANAG 5066 Annex B Edition 1 provides mechanisms (i.e., the CAS-1 linking protocol) to establish a point-to-point link (or links) for data communication. The CAS-1 protocol belongs to the class of link request/accept protocols that are effective at resolving the hidden-terminal problem in wireless point-to-point scenarios.

This annex introduces optional modes for enhanced media-access control capability for HF data communication in multi-node networks, and the prescribed method in which they may be used with other STANAG 5066 capabilities. These optional channel-access modes extend or modify, but do not replace, the channel-access and link-control mechanisms defined in Annex B of this STANAG.

J.2 MEDIA-ACCESS-CONTROL SUBLAYER (MACS)

Enhanced Media-Access Control capabilities are defined in the context of an augmented model of the HF Subnetwork's protocol stack shown in Figure J-1.

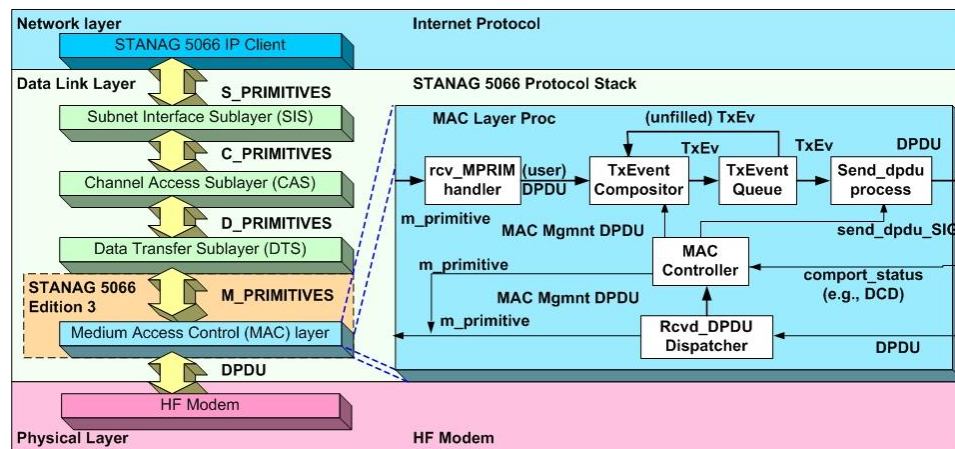


Figure J-1 — Augmented Model of the HF Subnetwork Protocol-Stack

The augmented model adds a Media-Access Control Sublayer (MACS) and inserts it below the Data-Transfer Sublayer of Annex C. Additional functionality implementing enhanced media-access control may be contained in the MACS, as shown in the figure.

The added media-access-control functionality:

- **shall** be based on the original DPDU message types defined in STANAG 5066 Annex C Edition 1, and
- **may** use in addition new DPDU types implemented for media-access control that **shall** conform to the message-definition rules and the Generic DPDU Frame Structure and DPDU field-element requirements of S'5066 Annex C Sections C.3.1 and C.3.2. New functionality to implement enhanced media-access-control **shall** be confined to the DPDU Type-Specific Header element and, if present, DPDU Payload element of any new DPDU type.

To minimize impact on pre-existing functionality defined in Annexes A, B, and C for other layers of the HF subnetwork, new DPDU types defined for media-access control **shall** only support peer-to-peer MACS-layer communication between nodes.

Node-to-node communication between HF subnet clients and peer-layer-to-peer-layer communication supporting the functionality of Annexes A, B, and C **shall** continue to use the DPDU types defined in Annex C. The SPDU and CPDU specifications of Annex A and Annex B **shall** be unchanged by the MACs functionality.

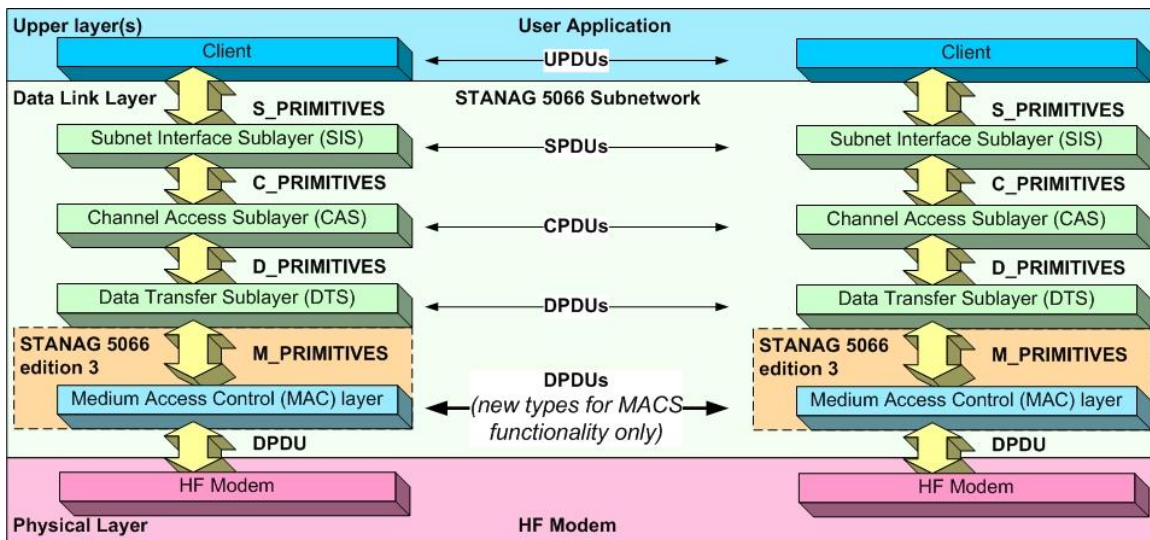


Figure J-2 — Peer-to-Peer Communication for Enhanced Media-Access Control

MACS functionality may be tailored as outlined in section J.3, which defines options to implement a range of functionality for multi-node networks. Detailed functional and performance specifications from other Annexes are cross-referenced for each option.

J.3 SUMMARY OF MEDIA-ACCESS-CONTROL MODES

Four Media-Access Control Modes are defined:

- Point-to-Point Mode (P2P)
- Carrier-Sense Media Access (CSMA)
- Wireless Token-Ring Media Access (WTRMA)
- Adaptive Time-Division Media Access (ATDMA)

These modes are summarized below. Implementation and performance requirements for these are standardized in the cross-referenced annexes.

J.3.1 (Default) Point-to-Point (P2P) Media-Access Mode

The default point-to-point (P2P) media-access mode enables two nodes to reserve the channel for their use in point-to-point communication. The default point-to-point channel-access control **shall** be implemented in accordance with STANAG 5066 Annex B. For this mode, the MACS provides no additional functionality.

J.3.2 Carrier-Sense Multiple Access (CSMA) Media-Access Mode

The Carrier-Sense Media Access (CSMA) mode enables a set of nodes to share the channel in a multi-node network. CSMA is a form of media-access control based on a node's ability to listen to the channel (i.e., radio-frequency carrier) and use its local knowledge that the channel is clear to control its transmissions to avoid interference — also called collisions — with the transmissions of other nodes.

A node **may** — indeed, a node **should** — implement the CSMA mode¹. If a CSMA mode is implemented, it **shall** be implemented in accordance with STANAG 5066 Annex K.

The CSMA mode conforms to the requirements of this Annex: it is implemented using only the message catalogue defined by STANAG 5066 Edition 1, and the basic capabilities of Annexes A, B, and C, with augmented requirements for Annex D to implement a Listen-Before-Transmit (LBT) mechanism with collision avoidance.

J.3.3 Wireless Token-Ring Media-Access (WTRMA) Mode

The Wireless Token-Ring media-access (WTRMA) mode enables a set of nodes to share the channel in a multi-node network. Token-ring or token-bus multiple

¹ Nodes should implement and use CSMA even when the intent is point-to-point communication, because it helps to resolve channel contention during the Physical-Link establishment protocol defined in Annex B.

access is a form of media-access control based on ownership of a 'token' that grants the right to transmit (RTT) on the channel. A token holder transmits until it no longer has data to send, or until its right-to-transmit timer expires, and then it passes the RTT token to its successor. Adaptation of each node's allocated channel capacity to the offered traffic load occurs automatically with the WTR mode.

The Wireless Token-Ring Media-Access mode **shall** be implemented in accordance with STANAG 5066 Annex L. The Wireless Token-Ring Protocol (WTRP) defined in that Annex is in two parts:

- a DPDU message design for the management tokens exchanged by network nodes, and
- the algorithms used to create, maintain, and repair the ring (i.e., transmission sequence) of nodes in the network.

WTRP's DPDU message design conforms to the requirements of this Annex: it is a new DPDU message type that conforms fully to the requirements of Annex C, whose DPDU-specific part provides the requisite data fields to satisfy the information exchange requirements of WTRP.

J.3.4 Adaptive Time-Division Media-Access (ATDMA) Mode

The Time-Division Media-Access (TDMA) mode enables a set of nodes to share the channel in a multi-node network. TDMA divides the channel access into timeslots that are allocated to nodes in the network (or, even more generally, allocated to network services).

Fixed TDMA protocols are known to be inefficient in channel utilization and service times when the traffic offered by the timeslot owner (i.e., the node or service to which the time slot is allocated) is mismatched to the channel capacity provided by the time slot. This is particularly true when the node or service has insufficient offered traffic to fill the timeslot(s) it has been allocated.

Adaptive TDMA (ATDMA) permits variations in the timeslot allocation or length that allow the network to adapt to variations in the traffic offered by nodes and in their service requirements.

In general, ATDMA's adaptivity requires new DPDU management message types with which nodes coordinate timeslot usage and length information. These DPDU management messages conform to the requirements of this Annex through their conformance with the structural requirements of Annex C.

Adaptive Time-Division Media-Access control shall be implemented in accordance with STANAG 5066 Annex M.

J.2 COMPATIBILITY AND INTEROPERABILITY ISSUES

Through their conformance with the generic DPDU-message structure defined in Annex C the enhanced media access protocols defined and cross-referenced herein are compatible in the limited sense that their common message elements (e.g., Maury-Styles synchronization preamble, DPDU type field, address fields, etc.) are recognizable regardless of which MACS mode is in operation or has been implemented by the node. Thus, all STANAG-5066-compliant implementations will be capable of decoding the field elements common to all the DPDU messages. And, in particular, all compliant implementations will be capable of determining the source address of the node that sent the DPDU and the type number of the DPDU that was sent, even if the node does not implement the MACs protocol for which the DPDU is used.

But the enhanced media access modes defined herein are not interoperable. There is no intent or expectation that a node implementing one of these enhanced media-access modes should be interoperable with a node implementing a different protocol. Rather, the intent and expectation is that standard operating procedures for network establishment will ensure that only nodes using the same media-access control protocol are network members. As this is a naïve view of what can happen in an operational network, this Annex defines provisions for nodes to discover the MAC-mode in use by other nodes, to, at the very least, recognize when they are using incompatible protocols and to take appropriate action to avoid mutual interference.

J.2.1 MAC-Mode Discovery

The processes by which nodes discover the media-access-control modes in use within a STANAG 5066 network are called MAC-Mode Discovery.

Nodes **should** implement MAC-Mode Discovery, which consists of the process elements defined here:

- Use of DPDU Type 15 Warning Messages – the (mandatory) provisions of Annex C Section C.3.12 **shall** apply to the implementation of MAC-Mode Discovery, as further amplified below.
- Channel Analysis – nodes implementing a given media-access mode shall analyze the channel usage by other nodes to detect violations of the protocol that it implements, and take actions as noted further below. These actions include the use of T15 Warning messages to notify neighbouring nodes of the protocol incompatibility.

J.2.1.1 *Use of T15 Warning Messages*

A STANAG 5066-compliant node that receives a DPDU message that is “unexpected or unknown” to it is required under the provisions of Annex C Section

C.3.12 to send a Type 15 DPDU (Warning) Message to the node that generated the unexpected or unknown DPDU. This requirement is unchanged by the enhanced-media-access modes defined herein.

In particular, and with cross reference to Annex C Section C.3.12, a node that receives a DPDU that is unexpected for the MACS protocol it is using (this may be an indication that the sending node is using a different MACS protocol) or that unknown to it (also a potential indication of an incompatible MACS protocol) will determine the source address of the node that sent the unexpected or unknown DPDU and send a Type 15 DPDU (Warning) Message to that node:

- As required in Annex C, the Type 15 DPDU (Warning) Message RECEIVED FRAME TYPE field **shall** indicate the Type number of the unexpected or unknown DPDU.
- If the node does not recognize the received DPDU, the Type 15 DPDU (Warning) Message REASON WARNING SENT field **shall** be set to the value assigned to the reason “Unrecognised D_PDU type Received”.
 - N.B.: This case applies to DPDU types with subtypes, i.e., the Type 6 DPDU (Management) that uses the Engineering Orderwire (EOW) Type field (as defined in Annex C Section C.3.2.2) as a subtype indicator to distinguish variants of the Extended Management Message types: the receiving node may recognize the main type (i.e., Type 6 Management Message) but not the subtype (i.e., the particular Extended Management Message subtype designated by the EOW-type field type). In this case, the node **shall** treat the DPDU as an unrecognized DPDU, and send a Type 15 DPDU (Warning) Message as above.
- If the node recognizes the received DPDU, but it was unexpected for the protocol that it was using, the Type 15 DPDU (Warning) Message REASON WARNING SENT field **shall** be set to the value assigned to the reason “Invalid D_PDU Received for Current State”.
 - N.B.: This case also applies to DPDU types with subtypes as described in the preceding case. If the node receives a DPDU of the proper type but if the subtype is unexpected, e.g., it designates an Extended Management Message defined for a MACS protocol that the node recognizes but is not currently executing, then in this case also, the node **shall** treat the DPDU as an unexpected DPDU, and send a Type 15 DPDU (Warning) Message as above.
- In particular, a node executing the P2P or CSMA mode that receives any of the new DPDU types defined in Annex L (for WTRMA) or Annex M (for ATDMA) (whether or not they are recognized to the receiving node) **shall** send a Type 15 DPDU as specified above.

Use of T15 DPDU (Warning) Messages of course is possible only when the node is configured in a compatible transmission mode; nodes configured for transmit-only operation (e.g., to providing exclusive support to a Broadcast Data Exchange Session as defined in Annex A) will not receive any DPDUs that could trigger the warning condition, and nodes configured for receive-only operation (e.g., to receive the Broadcast Data) would not be capable of sending the warning messages and therefore disable the protocol-error-detection logic and warning-message-generation functionality.

J.2.1.2 Channel Analysis

Transmissions by nodes executing one MAC-mode will likely be uncoordinated with the transmissions of nodes executing another (i.e., they don't participate in the protocol), and mutual interference may occur.

Nodes executing a given MAC-mode **shall** analyze the DPDU messages received from other nodes to detect violations of the protocol it is executing. Message analysis includes the following:

- processing of received Type 15 DPDU (Warning) Messages in accordance with Annex C and as amplified by Section J.2.1.1.
- analysis and identification of the MAC mode in use by other nodes:
 - transmissions that contain none of the new DPDU types (recognizable to the receiving node) defined in Annex L (for WTRMA) or Annex M (for ATDMA) **should** be assumed to originate from a node that is currently operating in P2P or CSMA modes, as these modes (i.e., P2P and CSMA) do not use any new DPDUs for their operation. This indication is one of the current operating mode, and not of capability.
 - Transmissions that contain Type 15 DPDU (Warning) Messages that warn ignorance of any of the new DPDU types defined in Annex L (for WTRMA) or Annex M (for ATDMA), i.e., that have a RECEIVED FRAME TYPE field defined for one of the new DPDU types and that denotes a REASON WARNING SENT field as unrecognized, **should** be assumed to originate from a node that is capable of operating only in P2P or CSMA modes. The presumption **should** be that node is a legacy (STANAG 5066 Edition 1 compliant) implementation incapable of executing either WTRMA or ATDMA.
 - Transmissions that contain Type 15 DPDU (Warning) Messages that warn of unexpected use of any of the new DPDU types defined in Annex L (for WTRMA) or Annex M (for ATDMA), i.e., that have a RECEIVED FRAME TYPE field defined for one of the new DPDU types and that denotes a REASON WARNING SENT field as unrecognized, **should** be assumed to originate from a node that is operating in a different MACS mode than the node to which the Type 15 DPDU (Warning) Message is addressed.

J.2.2 Embedding Broadcast / Multicast Traffic

Receive-only nodes (e.g., nodes in an emission-control [EMCOM] status) make no transmissions that can interfere with the operation of multi-node network, whatever MACS mode it uses. This may be exploited by nodes that do transmit in a multi-node network to embed broadcast or multicast traffic, a capability that is fully consistent with the intent of Annex A Section A.1.1 (from Edition 2 and later of this STANAG).

ANNEX J - GENERAL REQUIREMENTS FOR ENHANCED MEDIA-ACCESS-CONTROL (MAC) CAPABILITIES IN MULTI-NODE STANAG 5066 NETWORKS (INFORMATIVE)

J.1 INTRODUCTION

The channel-access control capability of STANAG 5066 Annex B Edition 1 provides mechanisms (i.e., the CAS-1 linking protocol) to establish a point-to-point link (or links) for data communication. The CAS-1 protocol belongs to the class of link request/accept protocols that are effective at resolving the hidden-terminal problem in wireless point-to-point scenarios.

This annex introduces optional modes for enhanced media-access control capability for HF data communication in multi-node networks, and the prescribed method in which they may be used with other STANAG 5066 capabilities. These optional channel-access modes extend or modify, but do not replace, the channel-access and link-control mechanisms defined in Annex B of this STANAG.

J.2 MEDIA-ACCESS-CONTROL SUBLAYER (MACS)

Enhanced Media-Access Control capabilities are defined in the context of an augmented model of the HF Subnetwork's protocol stack shown in Figure J-1.

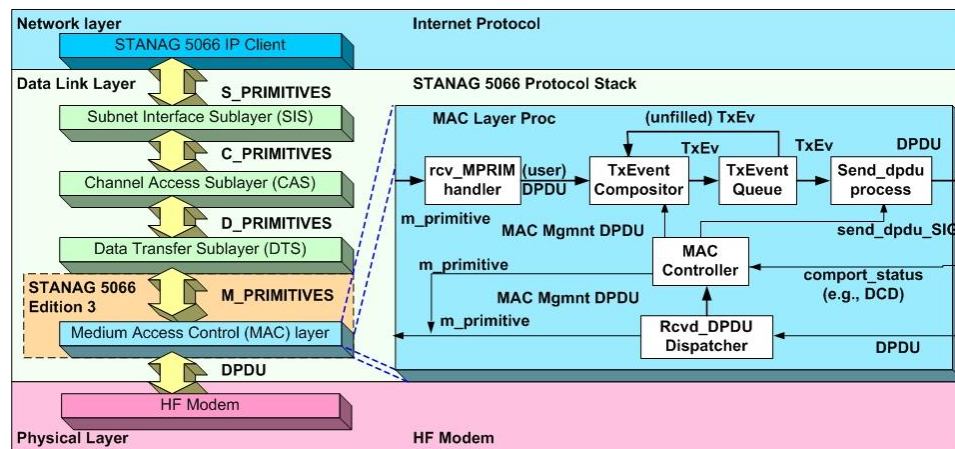


Figure J-1 — Augmented Model of the HF Subnetwork Protocol-Stack

The augmented model adds a Media-Access Control Sublayer (MACS) and inserts it below the Data-Transfer Sublayer of Annex C. Additional functionality implementing enhanced media-access control may be contained in the MACS, as shown in the figure.

The added media-access-control functionality:

- **shall** be based on the original DPDU message types defined in STANAG 5066 Annex C Edition 1, and
- **may** use in addition new DPDU types implemented for media-access control that **shall** conform to the message-definition rules and the Generic DPDU Frame Structure and DPDU field-element requirements of S'5066 Annex C Sections C.3.1 and C.3.2. New functionality to implement enhanced media-access-control **shall** be confined to the DPDU Type-Specific Header element and, if present, DPDU Payload element of any new DPDU type.

To minimize impact on pre-existing functionality defined in Annexes A, B, and C for other layers of the HF subnetwork, new DPDU types defined for media-access control **shall** only support peer-to-peer MACS-layer communication between nodes.

Node-to-node communication between HF subnet clients and peer-layer-to-peer-layer communication supporting the functionality of Annexes A, B, and C **shall** continue to use the DPDU types defined in Annex C. The SPDU and CPDU specifications of Annex A and Annex B **shall** be unchanged by the MACs functionality.

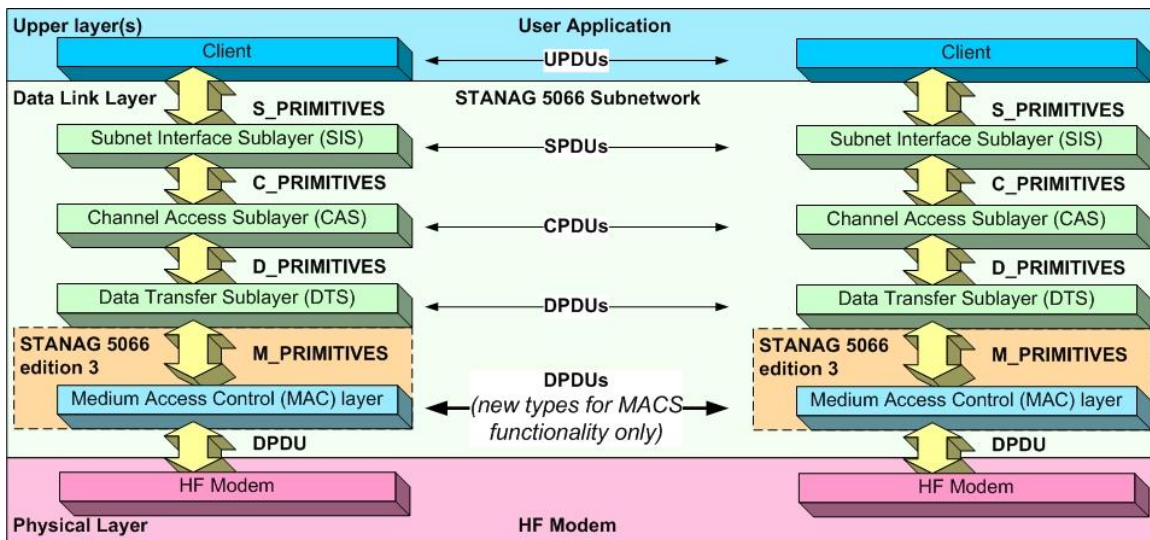


Figure J-2 — Peer-to-Peer Communication for Enhanced Media-Access Control

MACS functionality may be tailored as outlined in section J.3, which defines options to implement a range of functionality for multi-node networks. Detailed functional and performance specifications from other Annexes are cross-referenced for each option.

J.3 SUMMARY OF MEDIA-ACCESS-CONTROL MODES

Four Media-Access Control Modes are defined:

- Point-to-Point Mode (P2P)
- Carrier-Sense Media Access (CSMA)
- Wireless Token-Ring Media Access (WTRMA)
- Adaptive Time-Division Media Access (ATDMA)

These modes are summarized below. Implementation and performance requirements for these are standardized in the cross-referenced annexes.

J.3.1 (Default) Point-to-Point (P2P) Media-Access Mode

The default point-to-point (P2P) media-access mode enables two nodes to reserve the channel for their use in point-to-point communication. The default point-to-point channel-access control **shall** be implemented in accordance with STANAG 5066 Annex B. For this mode, the MACS provides no additional functionality.

J.3.2 Carrier-Sense Multiple Access (CSMA) Media-Access Mode

The Carrier-Sense Media Access (CSMA) mode enables a set of nodes to share the channel in a multi-node network. CSMA is a form of media-access control based on a node's ability to listen to the channel (i.e., radio-frequency carrier) and use its local knowledge that the channel is clear to control its transmissions to avoid interference — also called collisions — with the transmissions of other nodes.

A node **may** — indeed, a node **should** — implement the CSMA mode¹. If a CSMA mode is implemented, it **shall** be implemented in accordance with STANAG 5066 Annex K.

The CSMA mode conforms to the requirements of this Annex: it is implemented using only the message catalogue defined by STANAG 5066 Edition 1, and the basic capabilities of Annexes A, B, and C, with augmented requirements for Annex D to implement a Listen-Before-Transmit (LBT) mechanism with collision avoidance.

J.3.3 Wireless Token-Ring Media-Access (WTRMA) Mode

The Wireless Token-Ring media-access (WTRMA) mode enables a set of nodes to share the channel in a multi-node network. Token-ring or token-bus multiple

¹ Nodes should implement and use CSMA even when the intent is point-to-point communication, because it helps to resolve channel contention during the Physical-Link establishment protocol defined in Annex B.

access is a form of media-access control based on ownership of a 'token' that grants the right to transmit (RTT) on the channel. A token holder transmits until it no longer has data to send, or until its right-to-transmit timer expires, and then it passes the RTT token to its successor. Adaptation of each node's allocated channel capacity to the offered traffic load occurs automatically with the WTR mode.

The Wireless Token-Ring Media-Access mode **shall** be implemented in accordance with STANAG 5066 Annex L. The Wireless Token-Ring Protocol (WTRP) defined in that Annex is in two parts:

- a DPDU message design for the management tokens exchanged by network nodes, and
- the algorithms used to create, maintain, and repair the ring (i.e., transmission sequence) of nodes in the network.

WTRP's DPDU message design conforms to the requirements of this Annex: it is a new DPDU message type that conforms fully to the requirements of Annex C, whose DPDU-specific part provides the requisite data fields to satisfy the information exchange requirements of WTRP.

J.3.4 Adaptive Time-Division Media-Access (ATDMA) Mode

The Time-Division Media-Access (TDMA) mode enables a set of nodes to share the channel in a multi-node network. TDMA divides the channel access into timeslots that are allocated to nodes in the network (or, even more generally, allocated to network services).

Fixed TDMA protocols are known to be inefficient in channel utilization and service times when the traffic offered by the timeslot owner (i.e., the node or service to which the time slot is allocated) is mismatched to the channel capacity provided by the time slot. This is particularly true when the node or service has insufficient offered traffic to fill the timeslot(s) it has been allocated.

Adaptive TDMA (ATDMA) permits variations in the timeslot allocation or length that allow the network to adapt to variations in the traffic offered by nodes and in their service requirements.

In general, ATDMA's adaptivity requires new DPDU management message types with which nodes coordinate timeslot usage and length information. These DPDU management messages conform to the requirements of this Annex through their conformance with the structural requirements of Annex C.

Adaptive Time-Division Media-Access control shall be implemented in accordance with STANAG 5066 Annex M.

J.2 COMPATIBILITY AND INTEROPERABILITY ISSUES

Through their conformance with the generic DPDU-message structure defined in Annex C the enhanced media access protocols defined and cross-referenced herein are compatible in the limited sense that their common message elements (e.g., Maury-Styles synchronization preamble, DPDU type field, address fields, etc.) are recognizable regardless of which MACS mode is in operation or has been implemented by the node. Thus, all STANAG-5066-compliant implementations will be capable of decoding the field elements common to all the DPDU messages. And, in particular, all compliant implementations will be capable of determining the source address of the node that sent the DPDU and the type number of the DPDU that was sent, even if the node does not implement the MACs protocol for which the DPDU is used.

But the enhanced media access modes defined herein are not interoperable. There is no intent or expectation that a node implementing one of these enhanced media-access modes should be interoperable with a node implementing a different protocol. Rather, the intent and expectation is that standard operating procedures for network establishment will ensure that only nodes using the same media-access control protocol are network members. As this is a naïve view of what can happen in an operational network, this Annex defines provisions for nodes to discover the MAC-mode in use by other nodes, to, at the very least, recognize when they are using incompatible protocols and to take appropriate action to avoid mutual interference.

J.2.1 MAC-Mode Discovery

The processes by which nodes discover the media-access-control modes in use within a STANAG 5066 network are called MAC-Mode Discovery.

Nodes **should** implement MAC-Mode Discovery, which consists of the process elements defined here:

- Use of DPDU Type 15 Warning Messages – the (mandatory) provisions of Annex C Section C.3.12 **shall** apply to the implementation of MAC-Mode Discovery, as further amplified below.
- Channel Analysis – nodes implementing a given media-access mode shall analyze the channel usage by other nodes to detect violations of the protocol that it implements, and take actions as noted further below. These actions include the use of T15 Warning messages to notify neighbouring nodes of the protocol incompatibility.

J.2.1.1 *Use of T15 Warning Messages*

A STANAG 5066-compliant node that receives a DPDU message that is “unexpected or unknown” to it is required under the provisions of Annex C Section

C.3.12 to send a Type 15 DPDU (Warning) Message to the node that generated the unexpected or unknown DPDU. This requirement is unchanged by the enhanced-media-access modes defined herein.

In particular, and with cross reference to Annex C Section C.3.12, a node that receives a DPDU that is unexpected for the MACS protocol it is using (this may be an indication that the sending node is using a different MACS protocol) or that unknown to it (also a potential indication of an incompatible MACS protocol) will determine the source address of the node that sent the unexpected or unknown DPDU and send a Type 15 DPDU (Warning) Message to that node:

- As required in Annex C, the Type 15 DPDU (Warning) Message RECEIVED FRAME TYPE field **shall** indicate the Type number of the unexpected or unknown DPDU.
- If the node does not recognize the received DPDU, the Type 15 DPDU (Warning) Message REASON WARNING SENT field **shall** be set to the value assigned to the reason “Unrecognised D_PDU type Received”.
 - N.B.: This case applies to DPDU types with subtypes, i.e., the Type 6 DPDU (Management) that uses the Engineering Orderwire (EOW) Type field (as defined in Annex C Section C.3.2.2) as a subtype indicator to distinguish variants of the Extended Management Message types: the receiving node may recognize the main type (i.e., Type 6 Management Message) but not the subtype (i.e., the particular Extended Management Message subtype designated by the EOW-type field type). In this case, the node **shall** treat the DPDU as an unrecognized DPDU, and send a Type 15 DPDU (Warning) Message as above.
- If the node recognizes the received DPDU, but it was unexpected for the protocol that it was using, the Type 15 DPDU (Warning) Message REASON WARNING SENT field **shall** be set to the value assigned to the reason “Invalid D_PDU Received for Current State”.
 - N.B.: This case also applies to DPDU types with subtypes as described in the preceding case. If the node receives a DPDU of the proper type but if the subtype is unexpected, e.g., it designates an Extended Management Message defined for a MACS protocol that the node recognizes but is not currently executing, then in this case also, the node **shall** treat the DPDU as an unexpected DPDU, and send a Type 15 DPDU (Warning) Message as above.
- In particular, a node executing the P2P or CSMA mode that receives any of the new DPDU types defined in Annex L (for WTRMA) or Annex M (for ATDMA) (whether or not they are recognized to the receiving node) **shall** send a Type 15 DPDU as specified above.

Use of T15 DPDU (Warning) Messages of course is possible only when the node is configured in a compatible transmission mode; nodes configured for transmit-only operation (e.g., to providing exclusive support to a Broadcast Data Exchange Session as defined in Annex A) will not receive any DPDUs that could trigger the warning condition, and nodes configured for receive-only operation (e.g., to receive the Broadcast Data) would not be capable of sending the warning messages and therefore disable the protocol-error-detection logic and warning-message-generation functionality.

J.2.1.2 Channel Analysis

Transmissions by nodes executing one MAC-mode will likely be uncoordinated with the transmissions of nodes executing another (i.e., they don't participate in the protocol), and mutual interference may occur.

Nodes executing a given MAC-mode **shall** analyze the DPDU messages received from other nodes to detect violations of the protocol it is executing. Message analysis includes the following:

- processing of received Type 15 DPDU (Warning) Messages in accordance with Annex C and as amplified by Section J.2.1.1.
- analysis and identification of the MAC mode in use by other nodes:
 - transmissions that contain none of the new DPDU types (recognizable to the receiving node) defined in Annex L (for WTRMA) or Annex M (for ATDMA) **should** be assumed to originate from a node that is currently operating in P2P or CSMA modes, as these modes (i.e., P2P and CSMA) do not use any new DPDUs for their operation. This indication is one of the current operating mode, and not of capability.
 - Transmissions that contain Type 15 DPDU (Warning) Messages that warn ignorance of any of the new DPDU types defined in Annex L (for WTRMA) or Annex M (for ATDMA), i.e., that have a RECEIVED FRAME TYPE field defined for one of the new DPDU types and that denotes a REASON WARNING SENT field as unrecognized, **should** be assumed to originate from a node that is capable of operating only in P2P or CSMA modes. The presumption **should** be that node is a legacy (STANAG 5066 Edition 1 compliant) implementation incapable of executing either WTRMA or ATDMA.
 - Transmissions that contain Type 15 DPDU (Warning) Messages that warn of unexpected use of any of the new DPDU types defined in Annex L (for WTRMA) or Annex M (for ATDMA), i.e., that have a RECEIVED FRAME TYPE field defined for one of the new DPDU types and that denotes a REASON WARNING SENT field as unrecognized, **should** be assumed to originate from a node that is operating in a different MACS mode than the node to which the Type 15 DPDU (Warning) Message is addressed.

J.2.2 Embedding Broadcast / Multicast Traffic

Receive-only nodes (e.g., nodes in an emission-control [EMCOM] status) make no transmissions that can interfere with the operation of multi-node network, whatever MACS mode it uses. This may be exploited by nodes that do transmit in a multi-node network to embed broadcast or multicast traffic, a capability that is fully consistent with the intent of Annex A Section A.1.1 (from Edition 2 and later of this STANAG).

ANNEX K – HIGH-FREQUENCY CARRIER-SENSE MULTIPLE-ACCESS PROTOCOLS (INFORMATIVE)

K.1 INTRODUCTION

This Annex specifies a High-Frequency Carrier-Sense Multiple Access (CSMA) with Collision Avoidance protocol [1] for STANAG 5066 in multi-node single-frequency networks.

The protocol uses the basic capabilities of Annexes A, B, and C, with augmented requirements for Annex D (a requirement to provide a Data-Carrier-Detect signal) to implement a Listen-Before-Transmit mechanism.

The HF CSMA protocol introduces no new DPDU types to the STANAG 5066 catalogue. There is no explicit peer-to-peer communication required by the CSMA protocol. CSMA media access control relies on a node's local information of HF channel activity and inferences of the activity — or lack thereof — of other nodes.

Section K.2 of this Annex presents an overview of the protocol and provides a definition of terms. Details of the protocol, its state diagram, and parameter values are specified in section K.3

K.2 OVERVIEW: CARRIER-SENSE MULTIPLE-ACCESS PROTOCOLS

Definitions and management concepts are introduced below prior to detailed specification of the protocol in later sections.

K.2.1 Definitions

The following terms are used in the specification of the High-Frequency Carrier-Sense Multiple Access (CSMA) protocol.

K.2.1.1 Stations and Nodes

The terms “*station*” and “*node*” are used interchangeably to describe the communication entities on the shared HF channel.

¹ The protocol described here is based on a presentation to the High-Frequency Industry Association by Robert McFarland of Rockwell Collins., “Collision Avoidance using STANAG 5066 in a Network Environment”, with additional input from Michael Stringer of Harris Corporation.

K.2.1.2 Collisions

Channel collisions — or, simply, *collisions* — are simultaneous or overlapping transmissions by two or more nodes that interfere with each other, preventing reception by another node.

K.2.1.3 Listen-Before-Transmit (LBT)

Listen-before-transmit is the action a node takes to ensure that the channel is unoccupied and free of activity before it attempts to transmit.

K.2.1.4 DCD

DCD is the *Data-Carrier-Detect* signal provided by the communications equipment interface. A node senses that an HF radio carrier is present when *DCD* is *true* (or — using the nomenclature of some interfaces — asserted). Communications equipment that senses an idle channel will provide set *DCD* = *false* (noted in this annex as the !*DCD* signal).

K.2.1.5 VDCD

VDCD is a *Virtual Data-Carrier-Detect* signal, derived from reception of valid DPDU. Observation of DPDU header's End-of-Transmission (EOT) field (defined in Annex C Section C.3.2.3) may be used to extrapolate into the future and predict the time at which a channel will be idle.

K.2.1.6 Contention Interval

The *contention interval* is the period of time during which nodes attempt to access the shared HF channel.

K.2.1.7 Collision Avoidance

Collision Avoidance is a strategy that nodes use during the contention interval to increase the probability that some contending node successfully accesses the channel.

K.2.1.8 Node States

A node may be in one of the following states as it executes the CSMA protocol:

- *Offline* state (OFFLINE) — the *offline* state is a state in which a station acts as if it were physically offline, i.e., it can neither transmit nor receive;

- *Sensing* State (SENSE) — the *sensing* state is a state in which a station monitors channel activity (using the DCD or VDCD signals), waiting for it to become idle.
- *Listen-before-Transmit-Wait* State (LBT_WAIT) — the *Listen-before-Transmit-Wait* state is a state in which a station waits to determine if another node has transmitted on the idle channel.
- *Contention-Wait* State (CONT_WAIT) — the *contention-wait* state is a state in which the station waits a random time before it may transmit. Nodes wait a random time during this state while continuing to sense channel activity to avoid collisions.
- *CAS Linking* State (LINKING) — the *CAS Linking* state is a state in which the station attempts to establish a Physical Link with a destination node.

K.2.1.9 Timers

The following timers control operation of the CSMA protocol:

- LBT_WAIT_TIMER – the LBT_WAIT_TIMER controls the maximum length of time a node will wait on an idle channel before transitioning to the contention-wait state.
- CONT_WAIT_TIMER – the CONT_WAIT_TIMER controls the length of time a node (i.e., a contending node) waits to access the channel before it starts to transmit.

K.2.1.10 Scalar Control Parameters

The following scalar parameters control operation of the CSMA protocol:

- LBT_WAIT_TIMER_VALUE – the waiting time set for the LBT_WAIT_TIMER;
- CONT_WAIT_TIMER_VALUE – the waiting time set for the CONT_WAIT_TIMER; this is a computed value that depends on the number and width of time slots defined for the contention interval;
- NUM_CONT_SLOTS – the number of contention slots defined for the contention interval;
- CONT_SLOT_WIDTH – the duration of each contention slot.

K.2.2 Concept of Operations

A likely occurrence in multi-node HF subnetworks is that nodes need to communicate at the same time and may attempt to do so. Uncontrolled attempts to access and transmit on the same channel can lead to channel collisions when these

transmissions occur at the same time. Mitigating the effects of collisions requires retransmissions and delays that lower channel throughput, but, when retransmissions remain uncontrolled, can further decrease throughput. Repeated collisions on an uncontrolled channel can degrade the network severely. Carrier-Sense Multiple Access (CSMA) concepts that include collision-avoidance (CA) can provide simple yet effective mechanisms for a (small) number of nodes to share a single-frequency HF subnetwork when they each have traffic to send.

The concept of operations for the CSMA protocol follows.

From an offline, non-operating state a node starts the protocol by entering a carrier-sensing state, where it listens on the channel for a radio-frequency carrier. The mechanisms a node uses for carrier-sense may vary but this Annex assumes that the carrier is sensed using either the Data-Carrier-Detect (DCD) signal available from the communications equipment, or from a Virtual Data-Carrier-Detect signal generated by tracking the EOT-field of received DPDUs contained in previous transmissions.

A node with no data queued for transmission remains in the carrier-sensing state.

A node with queued data to send that senses an idle channel enters the Listen-Before-Transmit state, and sets a timer — the LBT_WAIT_TIMER — that controls its exit from the state. The timer is restarted whenever the carrier is detected. If the LBT_WAIT_TIMER expires and the node still has queued data, the node enters a contention state. At any time in the Listen-Before-Transmit state, if the node no longer has any queued data, the node returns to the carrier-sensing state. Note that the LBT_WAIT_TIMER timer value specifies the maximum time a node will wait on an idle channel. The node will wait for a longer time — potentially a much longer time — on a busy channel because the LBT_WAIT_TIMER is restarted whenever carrier is sensed.

Nodes wait before contending for the channel and transmitting because of the time delays associated with the carrier-sense mechanism. A node waits to determine that the channel truly is idle. Detection of a busy channel — i.e., declaration of the DCD or VDCD signals — is not instantaneous. A node's transmissions have a propagation delay between itself and any receiver. Then, DCD is declared by a receive modem on declaration of valid data and it takes the modem's decoder some number of interleaver frames to make this declaration, introducing additional delay. Additionally, for secure systems that use line crypto inserted between the HF modem and the STANAG 5066 system, the modem DCD is relayed through the cryptographic equipment, which introduces additional delay. These delays influence the length of the listen-before-transmit interval that enables nodes to avoid — but not completely prevent — collisions on the channel.

On completion of the Listen-before-Transmit interval i.e., when the LBT_WAIT_TIMER expires, a node with queued data to transmit enters a contention state, where it selects a random interval to wait before it enters the transmit state.

The contention-state's random waiting interval is quantized. It consists of a number of slots, each of a fixed width. The slot width is specified to insure that all nodes in the network can detect a node's transmission before the beginning of the next slot. A node picks a random time to wait by picking a random slot to start its transmission.

The random waiting period during the contention interval is intended to avoid collisions — the carrier-sense and listen-before-transmit states tend to synchronize the access attempts of multiple nodes and the random waiting time during the contention interval forces, with some probability of success, the attempts to occur at different times.

When the slot width is greater than the channel-sensing delays — i.e., the sum of the modem-, crypto-, propagation- and other delays — the probability of collision can be reduced to the order of the probability that two or more contending nodes pick the same contention slot.

Quantizing the contention-state's waiting time and constraining nodes to attempt transmissions only at slot boundaries actually reduces its length. Unconstrained transmission-attempt times during the contention interval leaves open the possibility of collisions between contending nodes.

K.3 SPECIFICATION AND PROTOCOL

The CSMA protocol is specified below in the following sections:

- K.3.1 Overall State Diagram , which presents the protocol as a state graph with directed transitions and transition events that may be conditional or unconditional;
- K.3.2 State Specifications, which presents detail of each state and outbound-transition tables that govern the actions that take place when the node is in the state, and the events that trigger transitions to the next state.

K.3.1 Overall State Diagram

Figure K-1 below shows the simplified CSMA state diagram in which only the states and transition events are shown. Each state is specified in detail in the following section.

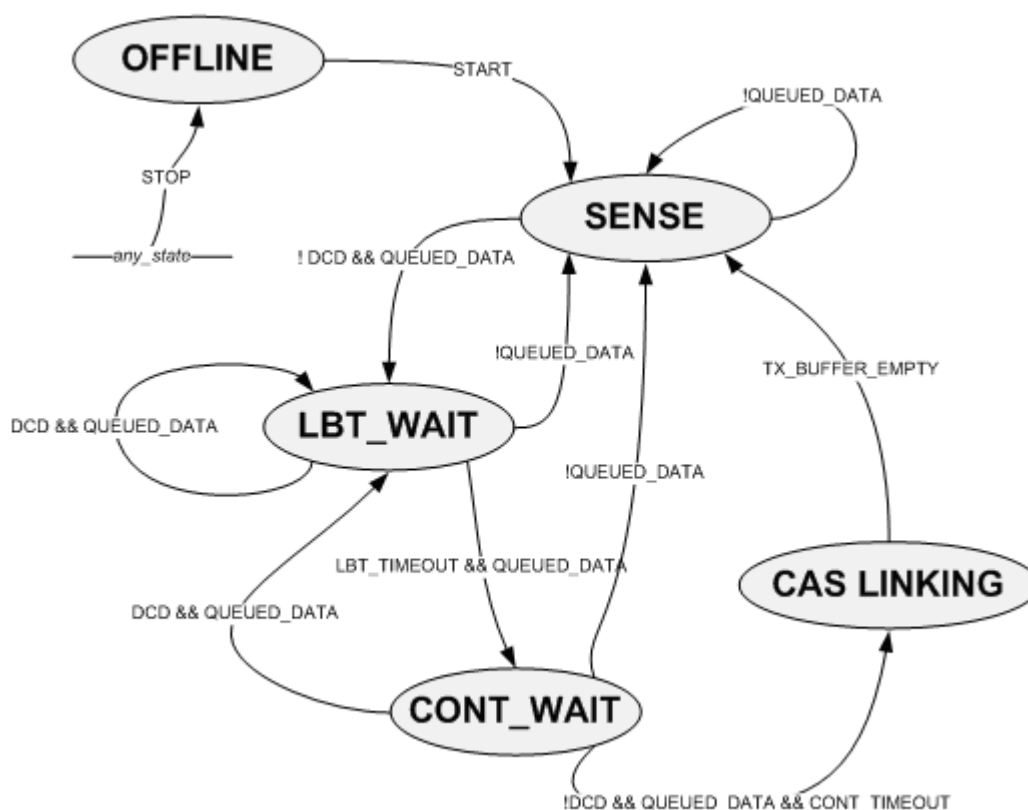


Figure K-1 — CSMA state diagram.

K.3.2 State Specifications

Specifications for CSMA operation for each state are provided below. For specification-purposes, transitions from each state to the next state shall be controlled by the accompanying outbound-transition tables, which specify:

- The current state,
- The event that triggers the transition,
- The action that shall be taken as a result of the transition,
- The next state to which the protocol transitions,
- The timer (if any) that is started.

K.3.2.1 Offline State (OFFLINE)

A node in the OFFLINE state **shall** neither send nor receive data.

Outbound transitions from the OFFLINE state **shall** conform to the table below.

Table 1 — OFFLINE-State Outbound-Transition Table

| state | event | Condition | Action | next state | start timer |
|---------|-------|--|--------------------------|------------|-------------|
| OFFLINE | START | start event received from the subnetwork management function | Neither send nor receive | SENSE | <i>none</i> |

A node **shall** transition to the SENSE state when it receives a START signal from the subnetwork management function (i.e., when the nodes starts operating).

K.3.2.1 Carrier-Sensing State (SENSE)

A node in the SENSE state **shall** listen for an idle channel, which is declared whenever the DCD signal is false (i.e, a !DCD signal).

Outbound transitions from the SENSE state **shall** conform to the table below.

Table 2 — SENSE-State Outbound-Transition Table

| state | event | Condition | action | next state | start timer |
|-------|------------------------|------------------------------|--------|------------|----------------|
| SENSE | !DCD (idle channel) | data QUEUED for transmission | | LBT_WAIT | LBT_WAIT_TIMER |
| SENSE | | no data for transmission | | SENSE | <i>none</i> |

A node with no data queued for transmission **shall** remain in the SENSE state regardless of channel activity.

A node with data to send **shall** wait in the SENSE state until it detects an idle channel, and then **shall** transition to the LBT_WAIT state.

K.3.2.1 Listen-Before-Transmit-Wait State (LBT_WAIT)

A node in the LBT_WAIT state **shall** wait while continuing to sense channel activity, to ensure that the channel is truly idle.

Outbound transitions from the LBT_WAIT state **shall** conform to the table below.

Table 3 — LBT_WAIT-State Outbound-Transition Table

| state | event | condition | action | Next state | start timer |
|----------|---------------------------------|------------------------------------|----------------------------------|------------|------------------------------------|
| LBT_WAIT | DCD (busy channel) | data QUEUED for transmit | | LBT_WAIT | LBT_WAIT_TIMER (restarts timer) |
| LBT_WAIT | LBT_WAIT_TIMER timeout event | data QUEUED for transmission | compute CONT_WAIT_TIMER_VALUE | CONT_WAIT | CONT_WAIT_TIMER |
| LBT_WAIT | | no data for transmission | | SENSE | <i>none</i> |

At any time while in the LBT_WAIT state, detection of a busy channel (i.e., receipt of a DCD or VDCD signal) **shall** restart the LBT_WAIT_TIME; the node **shall** remain in the LBT_WAIT state in this case.

At any time while in the LBT_WAIT state, detection of an empty data queue (e.g., it may have been emptied by a management function) **shall** force a transition to the SENSE state.

If the LBT_WAIT_TIME expires, the node **shall** compute a random value of CONT_WAIT_TIMER_VALUE, start the CONT_WAIT_TIMER, and transition to the CONT_WAIT state.

K.3.2.1 Contention-Wait State (CONT_WAIT)

Outbound transitions from the CONT_WAIT state **shall** conform to the table below.

Table 4 — CONT_WAIT-State Outbound-Transition Table

| state | event | condition | action | next state | start timer |
|-----------|---|---------------------------------|--------|------------|------------------------------------|
| CONT_WAIT | DCD (busy channel) | data QUEUED for transmission | | LBT_WAIT | LBT_WAIT_TIMER (restarts timer) |
| CONT_WAIT | !DCD && CONT_WAIT_TIMER timeout event | data QUEUED for transmission | | LINKING | <i>none</i> |
| CONT_WAIT | | no data for transmission | | SENSE | <i>none</i> |

Detection of DCD (i.e., detection of a busy channel) while in the CONT_WAIT state is taken as an indication that the node has 'lost' the contention round

to another, and thus, if the node still has data to transmit, it **shall** restart the LBT_WAIT_TIMER and transition to the LBT_WAIT state to wait once again for an idle channel.

Expiration of the CONT_WAIT_TIMER while the channel remains unoccupied (i.e. without any detection of DCD) is taken as an indication that the node has 'won' the contention round and thus, in this case and if the node still has queued data for transmission, the node **shall** transition to the LINKING state.

At any time while in the CONT_WAIT state, detection of an empty data queue (e.g., it may have been emptied by a management function) **shall** force a transition to the SENSE state.

K.3.2.1 CAS Linking State (LINKING)

Outbound transitions from the LINKING state shall conform to the table below.

Table 5 — LINKING-State Outbound-Transition Table

| state | event | condition | action | next state | start timer |
|---------|--|-----------|--------|------------|-------------|
| LINKING | (TX_BUFFER_EMPTY) or (TX_TIME > MAX_TRANSMIT_INTERVAL) | | | SENSE | <i>none</i> |

A node in the LINKING state transmit until its transmit-buffer is empty (i.e., TX_BUFFER_EMPTY == TRUE) or until the time it has transmitted exceeds the maximum transmit interval allowed to it (N.B.: a node's maximum transmit interval can be limited by various considerations, whether it is the 127.5 second limitation imposed by the EOT field size, or the time it takes to transmit the maximum number of unacknowledged DPDUs allowed by the ARQ protocol, or some other lesser time imposed by the subnetwork management function).

K.4 TIMERS AND TIMER VALUES

Operation of the CSMA protocol is controlled by the timers and timer values listed in the table below.

Table 6 —Default values for CSMA Timers

| Timer Name | Default Value | Units | Com- puted | Default-Value Name; Comments |
|-----------------|--|-------|---------------|--|
| LBT_WAIT_TIMER | 30 | secs | No | |
| CONT_WAIT_TIMER | RAND[0 ... NUM_CONT_SLOTS – 1]] * CONT_SLOT_WIDTH | secs | Yes | computed as a function of the contention-slot width and the randomly selected contention-slot number for the access attempt. |

K.5 SCALAR CONTROL PARAMETERS

Operation of the CSMA protocol is controlled by the scalar parameters listed in the table below.

Table 7 —Default values for CSMA Scalar Parameters

| Parameter Name | Default Value | Units | Com- puted | Default-Value Name; Comments |
|----------------------|---------------|---------|---------------|--|
| CONT_SLOT_WIDTH H | 3 | secs | no | DEFAULT_ CONT_SLOT_WIDTHH; optimization of this value requires that it be a function of the modem preamble duration, data-rate, interleaver duration. |
| NUM_CONT_SLOTS | 16 | integer | No | DEFAULT_NUM_CONT_SLOTS; the value selected is a balance between probability of one and only one node selecting the winning slot and acceptable delay (eg. 3 nodes, 90%probability, 16 slots) |

ANNEX L - HIGH-FREQUENCY WIRELESS-TOKEN-RING-PROTOCOL (WTRP) REQUIREMENTS

L.1 INTRODUCTION

This Annex specifies a High-Frequency Wireless Token Ring Protocol (WTRP) for enhanced media access control within single-frequency multi-node radio networks using STANAG 5066. The protocol is in two parts: a message design for the management tokens exchanged by nodes in the radio network, and the algorithms used to create, maintain, and repair the radio-transmission sequence (i.e., the virtual ring) of nodes in the network.

The token messages are sub-types of the Type-6 Management messages defined in STANAG 5066 Annex C message catalogue and added to STANAG 5066 as part of the effort to specify an efficient multiple access protocol for multi-node radio networks within that STANAG.

The High Frequency Wireless Token Ring Protocol (WTRP) is a self-organizing Medium Access Control (MAC) protocol for HF wireless networks. The MAC protocol by which mobile stations can share a broadcast channel is crucial in wireless networks, especially for HF wireless networks where bandwidth is considerably lower than other types of wireless networks. WTRP is a MAC protocol derived from Ergen et al [1] and tailored for low-speed wireless networks. WTRP provides QoS in terms of bounded latency and reserved bandwidth. WTRP is efficient in reducing the number of retransmissions due to collisions and is fair in that each station uses the channel for a bounded maximum amount of time.

WTRP requires that stations in a ring take turns to transmit for a specified maximum amount of time. WTRP is robust against single node failure. WTRP is different from its parent protocol [1] in that it provides the notion of self-rings, that it allows relaying of right-to-transmit tokens in connected networks with arbitrary connectivity, and that it requires nodes to back off from joining a ring if the contention among nodes that wish to join is too severe.

This Annex of STANAG 5066 is organized as follows.

- Section L.2 gives an overview of WTRP;
- section L.3 specifies the STANAG 5066 message design for WTRP management tokens;
- section L.4 gives the complete WTRP state diagram and description for each state in WTRP;
- section L.5 specifies parameters for ring-management and operation,
- section L.6 provides specification of the manner in which timer values are calculated and interrelated,
- section L.7 provides additional detail on token-ring management,
- section L.8 provides an analysis and overview of fault-recovery mechanisms during ring operation, and
- section L.9 specifies requirements for token transmission and the placement of tokens in a transmissions.
- section L.10 specifies requirements for support by WTRP to IP-interface autoconfiguration and IP-address resolution.

L.2 OVERVIEW: WIRELESS TOKEN RING PROTOCOL

Token-Ring definitions and management concepts are introduced below prior to detailed specification of the protocol in later sections.

L.2.1 Definitions

The following terms are used in the specification of the High-Frequency Wireless Token Ring Protocol (WTRP).

L.2.1.1 Stations and Nodes

The terms “station” and “node” are used interchangeably to describe the communication entities on the shared transmission medium.

L.2.1.2 Tokens, Rings and Ring Members

The WTRP protocol is a Medium Access Control (MAC) protocol. The task of this protocol is to schedule the access of two or more stations connected to the same physical medium, nominally an HF wireless network. Messages exchanged between stations for WTRP control are called *tokens*.

The WTRP protocol organizes stations in such a manner that they rotate a *right-to-transmit token* among stations connected to the same physical medium. Only a *station* that receives the *right-to-transmit token* has the right to transmit user data. This *station* is then the *token-holder*. In normal operation there should only be one *token-holder*.

A set of stations sharing the same *right-to-transmit token* is defined as a *ring*, or *virtual ring*. A station participating in a ring is called a *ring member*.

When a station starts initially it is not a member of any ring. It will only be able to become a *ring member* if there is at least already one station connected to the same physical medium. If the station connected to the medium finds out there is no existing ring, it will attempt to set up a new ring with itself as its only member. Such a ring is called a *self ring*. If the station trying to establish a ring finds another station willing to join the ring, the *self ring* becomes a *ring*. The station shall then become the *ring owner*.

The *right-to-transmit* will be passed in rotation among the ring members. Every time the *ring owner* receives the *right-to-transmit* a *ring cycle* has completed.

L.2.1.3 Successors and Predecessors

If station S_A is passing the right to transmit to station S_B , then station S_B shall be called the *successor* of station S_A . The *predecessor* of station S_B shall be station S_A .

In other words, the *successor* of station X is the station to which X sends the *right-to-transmit token* to and the *predecessor* is the station from which X received the *right-to-transmit token*. Note that these stations are member of the same ring.

L.2.1.4 Ring Transmit Order

The order in which the Right-To-Transmit (RTT) token is passed around in the virtual ring is called the *transmit order*, which also defines the *successor* and *predecessor* relationship among *ring members*. For example, if the RTT token is passed from station S_A to station S_B , from station S_B to station S_C , and back to station S_A , then the transmit order is $\{S_A > S_B > S_C > S_A\}$. In a ring whose transmit order equals $\{S_A > S_B > S_C > S_A\}$, station S_B is the *successor* of station S_A , station S_C is the *successor* of station S_B , and station S_A is the *successor* of station S_C .

The ring transmit order is sent to current and potential ring members in tokens, notably the right-to-transmit, solicit, and set tokens, as a Transmit-Order-List (TOL). Stations may modify the ring transmit order to accommodate connectivity changes in the network, loss or addition of stations in the network, or to define a more efficient ring transmit order.

L.2.1.5 Node States

The following WTRP states are defined in the protocol:

- Floating State (FLT) - the initial (starting) state where a station is not part of a ring and assumes there is an existing ring that it can join. The FLT state is fully documented in L.4.11.2.
- Self Ring State (SFR) - in this state a station assumes there is **no** existing ring to join, and will therefore try to setup a new ring and declare itself the ring owner. The new ring will start with this station as the only member and is therefore called a self-ring.
- Seeking State (SEK) - in this state a station considers itself to be in a self-ring and has broadcast an SLS (i.e., solicit) token as an invitation for other stations to join its ring.
- Solicit Reply State (SRP) - in this state a station tries to join another ring; it intends to respond to a received SLS token but is waiting for a timeout to avoid congestion before sending its reply.
- Joining State (JON) - In this state a station has replied to the invitation to join (i.e., to the received SLS token) by sending a SET token to the node inviting it to join the net, and proposing the joining node's position in the ring's transmit order list.
- Have-Token State (HVT) - in this state a station is part of a ring. It has received a RTT (right-to-transmit) token from its predecessor and with this token, the right to transmit. After the transmission it is responsible for passing the RTT token to its successor;
- Monitoring State (MON) - in this state a station is part of a ring. It passed the RTT token to its successor, but is unsure if the successor has received the right to transmit (i.e., the station is MONitoring the channel for an implicit or explicit acknowledgement the RTT was successfully passed);
- Idle State (IDL) - In this state a station is part of a ring and knows it has successfully passed the RTT token its successor; it will process any DPDU messages received from other stations;
- Soliciting State (SLT) - in this state a station is part of a ring, currently has the right to transmit, and is inviting new stations to join the ring by broadcasting an SLS token and listening for replies;

- Relaying State (RLY) - the relaying state is a state in which a station relays a REL token for a member which can not reach its successor directly;

L.2.1.6 Timers

Each state has associated with it one or more timers, and some states are associated with identical timers.

Timers defined within the WTRP state machine are:

- Claim Token Timer (TCLT) - Timer used in the *floating-state* (FLT). Controls the time a station waits while in the *floating state* to claim a token before transiting to another state; a station restarts its TCLT timer when it transits to the FLT state.
- Solicit Successor Timer (TSLS) - This timer is used in the *self-ring-state* (SFR). If it times out the station shall transit to the *seeking-state* (SEK). The timer shall be set with a random timeout to reduce the probability of collisions by transmissions from stations attempting to establish different rings at the same time.
- Solicit Reply Timer (TSRP) - Timer used in the *solicit-reply-state* (SRP). A station starts its *solicit-reply timer* when it transits to the SRP state. If it expires a station will transit to the JON state where it replies to one of the received SLS tokens, if any, with a SET token.
- Contention Timer (TCON) - Timer used in the *joining-state* (JON). It controls the time a station waits for a response from another station following an attempt to join the network, so-named because failure to receive a response is attributed to contention with other stations attempting to join the network at the same time; a station restarts its *contention timer* when it goes to JON state.
- Idle Timer (TIDL) - Timer used in the *idle-state* (IDL). It controls the time a station waits for its *right-to-transmit* before transiting to the *floating-state* (FLT).
- Solicit Wait Timer (TSLW) - This timer is used in the *soliciting state* (SLT) and the *seeking-state* (SEK) by stations inviting new ring members. When it expires, the inviting station it will randomly selecting a station of all those that solicited to join, and add it to the ring and the Transmit Order List.
- Token Pass Timer (TPST) - Used in the *monitoring-state* (MON) and the *relaying-state* (RLY). It controls the time a station waits after passing an RTT (or other) token to another *station* and failing to hear an implicit or explicit acknowledgement before considering the *right-to-transmit* as lost.
- End of Transmission Timer (TEOT). This timer is used in the *have-token-state* (HVT) state by a station with the right-to-transmit; it controls the length of the station waits to transit from the HVT, and is calculated as a function of the length of data transmitted and the transmission rate; it is closely related to the value encoded in the DPDU EOT field.

L.2.1.7 Token Contents

Table L-1 shows a list of common fields of the WTRP tokens, using the RTT as example.

Table L-1 - RTT Token Field Values

| Location | Field | Comment |
|---------------|-------|--|
| Token Header | DA | Destination Address: the node to which this token is sent. (In one case this is a broadcast address) |
| | SA | Source Address: shall always be the address of the station sending the token. |
| | RO | Ring Owner/ |
| | SN | Sequence Number: incremented by one every time the <i>right-to-transmit</i> is passed to a <i>successor</i> . |
| | GSN | Generation Sequence Number: Incremented by one only by the <i>ring owner</i> every time the <i>right-to-transmit</i> is passed to its <i>successor</i> (i.e., a generation represents one cycle of RTT ownership in the ring transmit order. |
| | NS | New Successor: could be used to define the successor of the destination station; For a REL-token this is the final destination. |
| | NON | Number Of Nodes: Number of nodes in the ring. |
| Token Payload | TOL | Transmit Order List: the list of nodes specifying the sequence of ownership of the RTT token |
| | DM | Distance Matrix: the distance, measured in hops, between each pair of nodes in the network. |

There are eight fields required in the header of any token for WTRP operation: FC, RO, DA, SA, SN, GSN, NS, and NON; two additional fields are included in the token payload, when present, of a token: TOL and DM. The fields are described further in Section L.3.2.2

L.2.1.8 Network Graph

A graph $G(N, L)$ is a representation of a network of nodes in a set N , with pairs of nodes connected by one-way (i.e., unidirectional) links from a set L . Operations using the graph of a WTRP network, (e.g., computation of path lengths between nodes) are performed to optimize the ring transmit order and token-passing performance during network operation.

L.2.1.9 Promiscuous reception

Promiscuous reception is a receive mode in which a station performs limited processing and information collection on all DPDUs it receives, whether or not they are not addressed to it (i.e., the station takes in all traffic).

Promiscuous reception is the means by which a station discovers and confirms the existence or loss of links in the HF radio network (or that may be used to construct a network), compiles local node-adjacency information, and ultimately computes node-to-node distance information usable to optimize the performance of the token-ring media-access protocol.

L.2.1.10 Adjacency Matrix

An adjacency matrix is a square ($N \times N$) matrix (two-dimensional array) whose elements connote the existence of one-way radio links between nodes, taken pairwise, in a network of N nodes.

The adjacency matrix is denoted as A , with elements $a_{i,j}$ such that:

$$A = [a_{i,j} = 1 \text{ if a link exists from node } i \text{ to node } j, 0 \text{ otherwise}]$$

When all links in the network are two-way (i.e., bi-directional), $a_{j,i} = a_{i,j}$ for all i and j , and the adjacency matrix is symmetric.

L.2.1.11 Distance Matrix

A distance matrix is a square ($N \times N$) matrix (two-dimensional array) containing the minimum distances measured as the number of links (also called hops) between nodes, taken pairwise, in a network of N nodes.

The distance matrix is denoted as DM , with elements $dist_{i,j}$ such that:

$$DM = [dist_{i,j} = \text{the minimum distance in hops from node } i \text{ to node } j]$$

There are a number of algorithms that may be used to compute the distance matrix D given the adjacency matrix A . This document makes no recommendation on the choice of algorithm, only that the station be capable of computing D from A .

Note that given DM , the corresponding adjacency matrix A can easily be recovered, as

$$A = [a_{i,j} = \begin{cases} 1, & \text{if } (dist_{i,j} = 1); \\ 0, & \text{otherwise.} \end{cases}]$$

L.2.1.12 Tour

A *tour* of a graph $G(N,L)$ is a sequence of nodes from the graph such that each node appears at least once and two nodes are adjacent in the sequence only if they are adjacent in the graph.

An unconstrained, ordinary tour allows revisits to network nodes.

In the context of the wireless token-ring protocol, the sequence of ownership of the right-to-transmit (RTT) token, i.e., the Transmit-Order-List should be a *closed tour* of the network that starts and ends with nodes that are adjacent in the network, i.e., every node gets an opportunity to transmit, and passing the RTT token along the closed tour is feasible because nodes that are adjacent in the transmission sequence are adjacent in the network. A network in which it is impossible to find closed tours without revisiting some nodes is a network in which the RTT token must be relayed.

L.2.1.13 Hamiltonian Tour

A *Hamiltonian tour* is a tour in which each node appears only once.

A *closed Hamiltonian tour*, i.e., a Hamiltonian tour in which the starting and ending nodes are adjacent, is an optimal path over which to pass the RTT token, as it requires (and uses) no relay nodes to visit all nodes in the network.

L.2.1.14 Ring-Cycle Length (RCL)

The Ring-Cycle Length (RCL) is the length in hops of the tour through the WTRP network taken by the RTT token as it follows the ring transmit order list (TOL).

L.2.1.15 Modified Nearest Insertion Method

The Modified Nearest Insertion Method (MNIM) is an algorithm to find a minimum-cost tour (e.g., a closed Hamiltonian tour) in a complete network of weighted links. MNIM is recommended as the method for re-computation of the Ring-Cycle-Length when stations join the network, or when the ring's TOL is recomputed to find a better transmit order.

(N.B. see ISBN 0-07-027893-8, Data Structures, Algorithms and Object-Oriented Programming, by Gregory L. Heileman, University of New Mexico, section 12.4)

L.2.1.16 Tok(TOL, DM)

The nomenclature Tok(TOL, DM) connotes a WTRP token – of any type, e.g., RTT, REL, SLT, or SET) – containing a Transmit Order List (TOL) and Distance Matrix (DM).

The RTT, REL, and SLT tokens must contain a TOL and DM; the other tokens may contain a TOL and DM. The SET token may contain the TOL and DM. This enables the soliciting-station sending the SET token to propose where it should be inserted in the transmit order if it is not the inviting station's immediate successor.

L.2.2 Concept of Operation

Operational concepts for the Wireless Token Ring are outlined below.

L.2.2.1 Ring Operation

The basic concept of the Wireless Token Ring Protocol is to provide a mechanism that allows two or more stations operating on the same single-frequency radio channel to share it in such a way that only one station will transmit at a time. Only the station that has the *right-to-transmit (token-holder)* is allowed to transmit on the shared channel. The *right-to-transmit* is passed onward to all stations that joined the ring. Figure L-1 shows how the *right-to-transmit* is circulated among the *ring-members*.

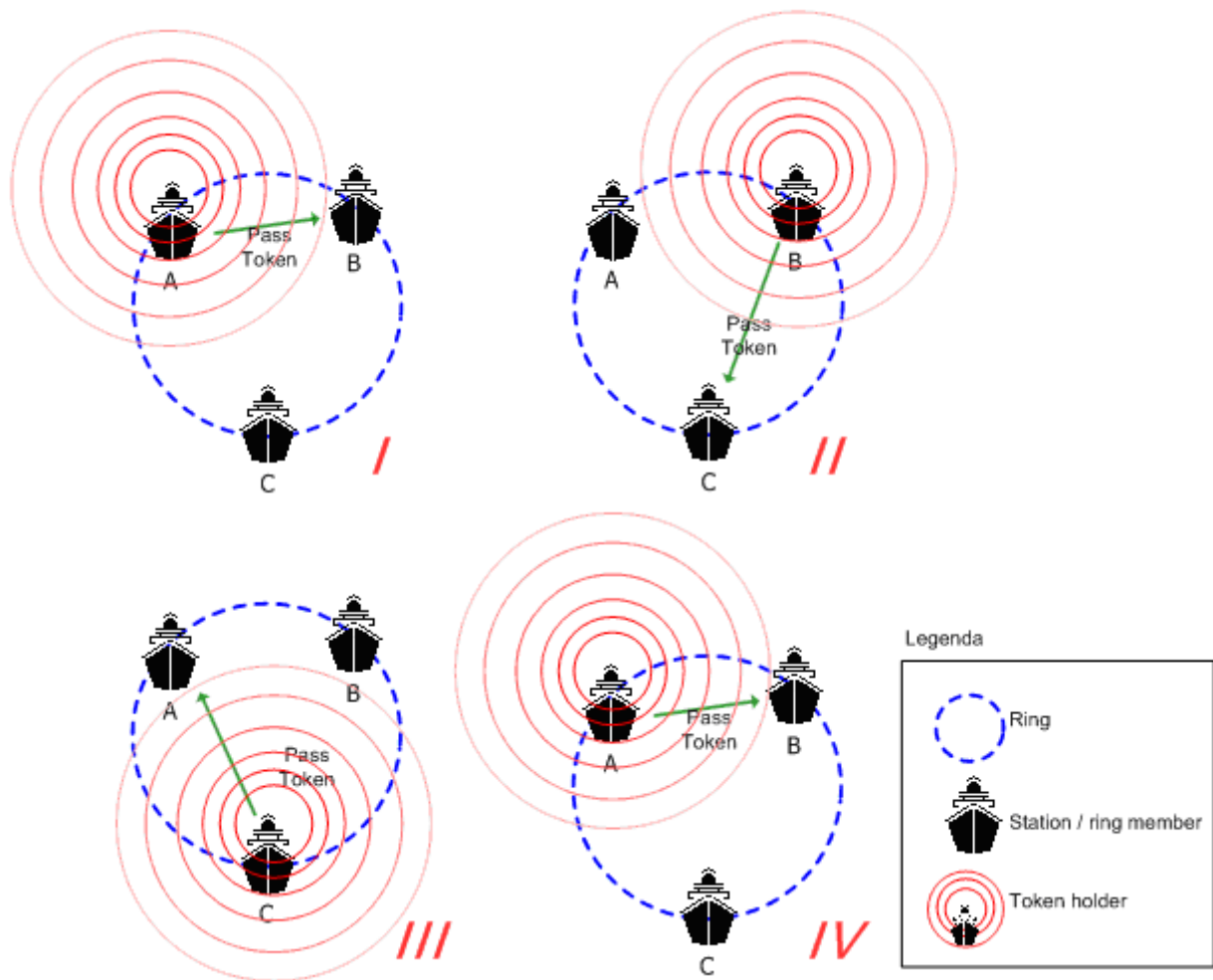


Figure L-1 - Normal Token-Ring Operation

The ring is a closed cycle of stations that transmit each in turn in a prescribed sequence, which will be adapted if new stations join the network or changes in network connectivity force adaptation. When a station receives the *right-to-transmit* from its *predecessor* it will take control of the channel. The *right-to-transmit token* (like many of the WTRP token messages) contains the *transmit-order-list (TOL)* and the node-to-node *distance matrix (DM)*, which the transmitting node may update based on its own local information before passing the *right-to-transmit token* to the next node in the *transmit-order-list (TOL)*. A node will close its transmission with a RTT-token or REL-token that passes the *right-to-transmit* to its *successor*. If the station has no data to transmit, then it shall pass the *right-to-transmit* immediately.

The data sent by a station are packed in a set of one or more D_PDU. The D_PDU **shall** count down the *end of transmit time (EOT)* in accordance with the requirements of Annex C Section C.3.2.3, with the *right-to-transmit*, the last D_PDU, having the smallest EOT value of the sequence. A station **shall** not exceed the maximum transmission time allowed.

Note that though the transmission sequence and ownership of the RTT token in the ring is prescribed, a station with the *right-to-transmit* **may** send data to any other station in the network that is within radio range, not just its successor or predecessor in the virtual ring. Any station in the virtual ring may therefore engage in multiple concurrent traffic exchanges with any

other station in range. This includes the capability to support concurrent soft-link, physical-link, and ARQ connections in accordance STANAG 5066 Annex A, B, and C, albeit with retransmission timers and other timing parameters modified to allow for the network size.

Possession of the *right-to-transmit* controls access to the radio channel; it does not prescribe the destination(s) to which traffic can be sent. The token-ring protocol controls node access to the transmission medium to preclude collisions and self-interference in the network and thereby increase the efficiency and throughput in the network. To do so, in general, there is no requirement that all nodes in the network be in communications range, only that the network have a closed tour, i.e., that a cyclic sequence of nodes exist where every node is in communications range of its predecessor and successor in the sequence. For incomplete networks, whether or not a closed Hamiltonian tour can be constructed for the RTT token, i.e., whether or not relay of the right-to-transmit is required, a higher-level process, not defined in this annex, may be required to relay traffic-bearing DPDUs. This higher-level process could, for example, be an IP-routing protocol (if the traffic-bearing DPDUs encapsulate IP datagrams) or a mail-forwarding protocol (if the traffic-bearing DPDUs encapsulate e-mail or S4406 Tactical Military Messages).

Each station will retransmit tokens passing the *right-to-transmit* to its *successor* until this has been acknowledged by its *successor* (or intermediate relay). Various attempts to forward the *right-to-transmit* may occur, but data will only be sent by a node after it has received a token passing it the *right-to-transmit*.

L.2.3 Ring formation

Before operating as a ring, a ring must be formed by stations on the same radio channel. The ring is a self-organizing and self-repairing mechanism, subject to a number of requirements:

- As soon as a station starts to operate it will listen to the radio channel in a floating state, unaffiliated with any ring.
- A station may transmit data (i.e., D_PDU other than ring-management tokens) only if it is part of a ring that is not a self ring.
- A station shall first listen for an existing ring on the radio-channel; if it hears a ring it will try to join that ring, otherwise it will form a new ring.
- If a station receives the right-to-transmit from another node, it has become a part of the ring.
- If a station is unable to pass the right-to-transmit to its successor, it shall exclude its successor from the ring, restructure and re-optimize the transmit order list using the remaining ring members, and attempt to pass the right-to-transmit to its new successor; this process may be called iteratively until the token is successfully passed or the node reverts to a self ring.
- A station dropped from a ring must rejoin the ring to become again part of it and obtain transmission rights.

Note that there are two possible ways to form a ring:

- Join an existing one.
- Start a new ring yourself.

The process of Ring Formation is illustrated below in Figure L-2 - Ring Formation.

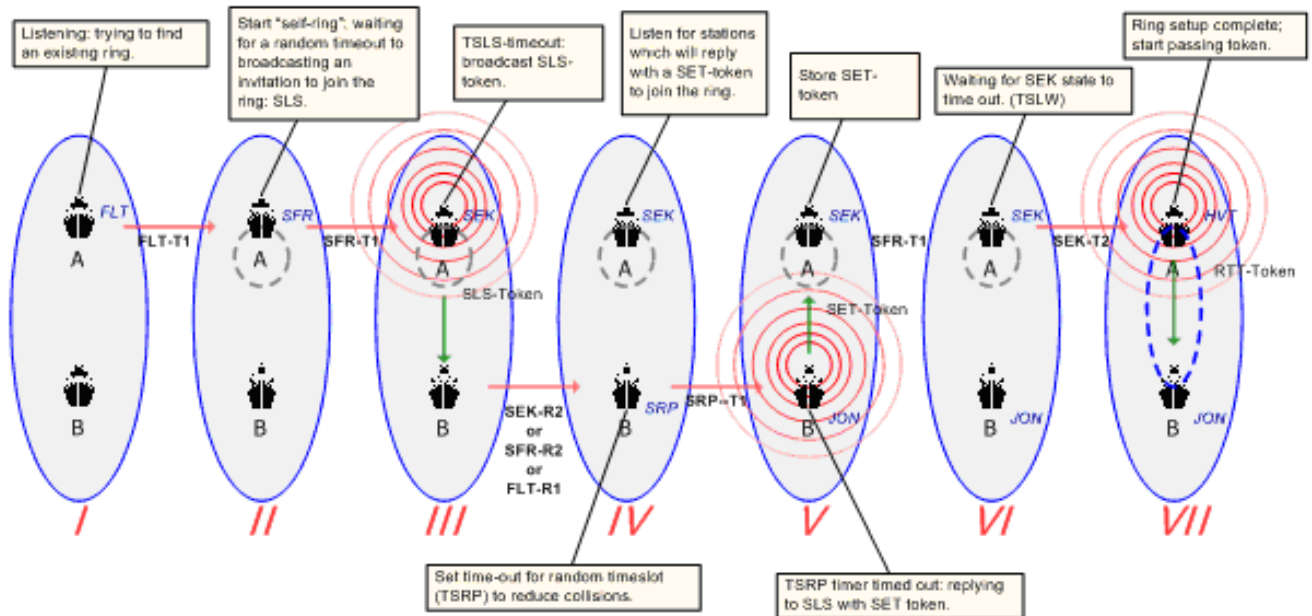


Figure L-2 - Ring Formation

Figure L-2 shows a time-sequence evolution of two stations, station A and station B, as they form a two-node ring.

In situation **I** station A is listening for an existing ring. After its TSLT timer times out, station A assumes there is no existing ring (otherwise it would have heard it) and it transits from a FLT state to the SFR state illustrated in situation **II**.

The transition moment from the SFR state in situation **II** to the SEK state in situation **III** is controlled by a random timeout that avoids collision between two stations in the same state. The value of the timeout is calculated based on picking an asynchronous timeslot (See TSLT timer details in section L.6).

In the SEK state of situation **III** a station invites new members by sending an SLS-token. Each station that is soliciting to join will reply to an SLS-token. There could be more than one station waiting to reply to an SLS-token. Therefore, to reduce the probability of collision between replies from multiple joining nodes, the joining node's reply shall be transmitted in a randomly chosen time slot among a set of slots available for replies. The station that announced the SLS will pick one random joiner and send it the right-to-transmit. The ring is created at this time.

Once the ring is created, each node periodically creates an opportunity to invite new stations to join the ring by broadcasting an SLS token and waiting for SET-token replies from soliciting joiners.

L.2.4 Ring Entry

The invitation to join shall be made periodically by a station receiving the right-to-transmit, with invitation frequency about once every SLS_INTERVAL seconds. The opportunity to invite new members is passed through the TOL from predecessor to a node and then to its successor, so the opportunity to invite new members to join will be equally divided over all stations, in time and in space. A station uses the token sequence number (SN) to determine when is its turn to create a solicit opportunity. Ring entry may be inhibited to preclude the ring from growing too large. The equation used to determine this from the SN is defined in L.4.8.

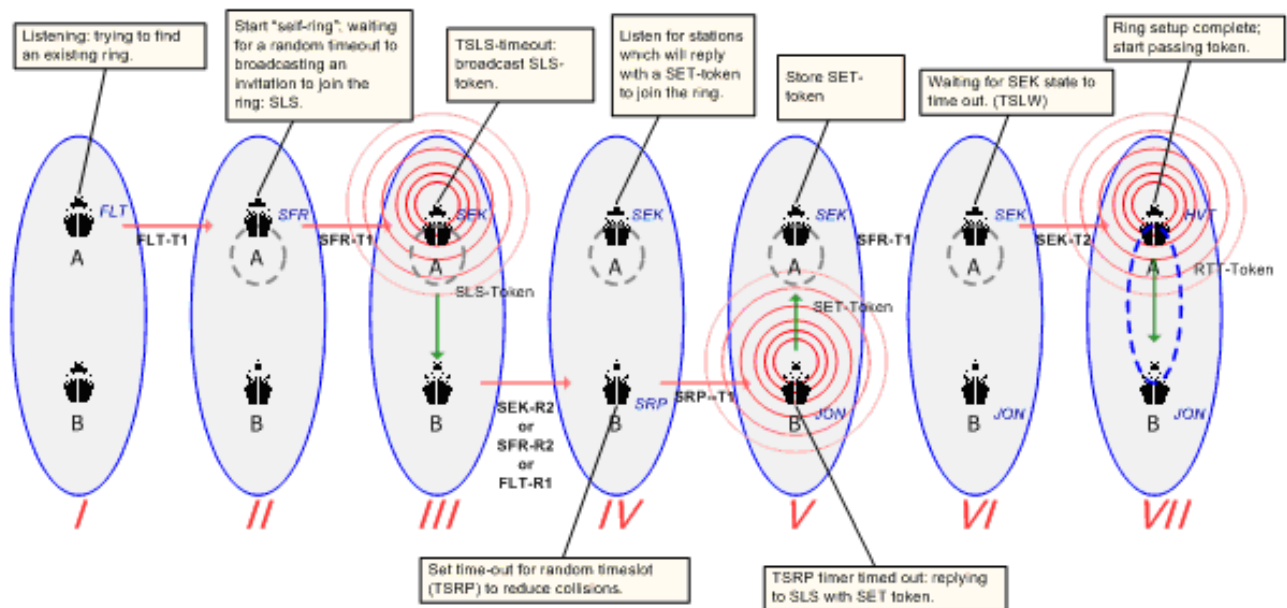


Figure L-3 - Ring Entry

Figure L-3 illustrates the ring created by station A and station B in the earlier example (see section L.2.3, Figure L-2), and picks up the time-sequence evolution starting when station C joins the ring.

In the first situation **I** station C hears an invitation to join (announced with the SLS token transmitted by station A). The SLS token contains the TOL and DM, which a joining node will use with its own local knowledge of stations it has heard to determine its position in the transmit order. The joining node will update the TOL and DM, placing itself in the TOL to minimize the increase in the resultant *ring-cycle-length* (RCL). Station C then sets a timer, which will expire at the start of a randomly chosen timeslot following the invitation. In situation **II** station C waits for this chosen timeslot. In situation **III**, the timer has expired and it is time to reply to the SLS-token with a SET-token soliciting to join. In situation **V** station the set of timeslots have finished and Station A picks a random solicitor of all those heard and promotes the solicitor to its successor. Station A will now pass the right-to-transmit to Station C, which has now successfully joined the ring.

L.2.5 Connectivity Update

A node's locally known changes to the Distance Matrix (DM) are inserted into DM and disseminated whenever the node forwards a Tok(TOL, DM). These changes are computed based on the node's observed changes in its local connectivity (i.e., on changes in its local Adjacency matrix), and how they affect changes in the DM received from its predecessor in the TOL.

Each node, whether in a ring or attempting to join a ring **must** keep an up-to-date list (or matrix) of its adjacencies¹. This serves as a fundamental building block of the protocol. A node **shall** use this adjacency data together with a received distance matrix (which implicitly contains an adjacency matrix) to compute the distance between itself and all other nodes in the ring. It also has the distance between all other nodes in the ring as reported by the other nodes via the DM. If this node is in the ring it **shall** update the row in the DM corresponding to itself with these distances. If this node joins the ring it **must** optimally insert itself into the TOL and augment the DM to include itself. This augmented payload is carried in the SET token.

L.2.6 Local-Adjacency Matrix Determination

Nodes **shall** maintain an Adjacency matrix $A = \{ a_{ij} \}$ to assist in the computation of the Distance Matrix. The value $a_{ij} = 1$, if node j can receive from node i , and 0 otherwise.

Nodes **shall** update the entries in the Adjacency Matrix using knowledge obtained from DPDU's received from any other nodes as follows:

- Node j **shall** set $a_{ij} = 1$, whenever a DPDU is received from node i ;
- Node j **may** set $a_{ij} = 1$, and set $a_{ji} = 1$, whenever an ACK DPDU is received from node i [N.B. an ACK connotes that the link is directional].
- Node j **may** set $a_{ik} = 1$, and set $a_{ki} = 1$, whenever an ACK DPDU is received from node k addressed to node i [N.B. an ACK connotes that the link is directional].

L.2.7 Ring-Cycle (i.e., TOL) Reconfiguration

There are three scenarios or cases in which the ring-cycle, i.e., the TOL, **must** or **may** be reconfigured:

- Scenario 1 (Joining Scenario): A node joins the network and thereby **must** be inserted into both the TOL and the DM;
- Scenario 2 (Transient-Topology Scenario): Changes in the network topology could result in changes in the distance matrix and **may** force a change in the TOL, e.g., when a successor node becomes unreachable (even with relay);
- Scenario 3 (TOL-Optimization Scenario): Sub-optimal TOL (e.g., TOL that use more token relays than necessary) could evolve in a network during Joining or Transient Topology

¹It is not specified how this adjacency list is maintained. At a minimum a passive node (one not participating in an active ring) must build this list by adding to it nodes in which it has received any tokens or data DPDU's. Likewise an active node must remove from its adjacency list a node in which it has failed to pass a token.

scenarios, and reconfiguration of the TOL to obtain a shorter ring-cycle length (RCL) **may** be performed when the network topology has stabilized. Any node **may** change the transmit order to create a shorter RCL. It does so by changing the TOL and by making a corresponding permutation of the DM.

These three scenarios are considered in more detail below.

All three scenarios are based upon the information contained in the ensemble of currently maintained lists of adjacencies by stations in the ring, information that is shared implicitly in the DM.

L.2.7.1 Ring-Reconfiguration in the Joining Scenario

A new node joining the network results in a new network topology, through determination and recognition of the new links between the joining node and other network members. The new node **must** be added to the TOL, rows and columns in the DM **must** be added in the correct positions and populated with values representing the distance between the new node and other nodes in the network.

The joining node **should** respond to invitations from a network node and **may** join the network as the inviter's successor if it yields a minimal increase in the resulting RCL. On obtaining the SET token from the new member, the inviting node **shall** make any further updates required to the TOL and DM, and these are passed on to the next node when the inviting node passes the token.

Alternatively, a joining node **may** respond to the invitation by inserting itself in the TOL in a position other than the direct successor to the inviting node. Selection of an alternate location in the TOL for the joining node **shall** be made if and only if the resulting TOL has a shorter length than would result if the node inserted itself as a direct successor to the solicitor.

Support for this last capability requires that a joining node be responsible for augmenting the token payload (TOL and DM) rather than having the inviting node do this. Consequently, the SET token sent by the node attempting to join **shall** contain this augmented payload including the placement of the joining node in the TOL and an additional row with distances and column in the DM.

L.2.7.2 Ring-Reconfiguration in the Transient-Topology Scenario

A node **should** leave the network gracefully by waiting until its transmit opportunity and then forwarding the right-to-transmit to its successor having deleted itself from both the TOL and the DM. The node will be denied subsequent opportunity to transmit until it completes a joining procedure as any other node.

Instances will occur with topology change where a node holding the RTT token cannot find a path to the next node in the TOL, e.g., when the successor station has suffered an

equipment failure or otherwise left the ring without notification. These changes **must** be reflected in both the TOL and DM².

L.2.7.3 Ring-Reconfiguration in the TOL-Optimization Scenario – Ring Transmit-Order Recomputation

Rings **may** be reconfigured with the generation and promulgation of a new TOL only after they have been stable for one or more ring cycles. A node **may** declare the TOL as candidate for reconfiguration – and attempt to optimize it – only when $(TOL, DM)_k = (TOL, DM)_{(k-1)}$, i.e., when neither the transmit-order nor the network topology has changed during the last ring-cycle.

A node **may** choose any algorithm when recomputing the transmit order list, though good, near-optimum algorithms are closely related to those which solve the so-called Travelling-Salesman Problem (TSP) for a weighted network. Any node that computes a transmit order list with an RCL that is less than the one currently in use and for which the above conditions apply **may** replace the current transmit order with the recomputed one.

L.2.7.4 Ring-Cycle-Length (i.e., Tour-Length) Computation

The length of the ring-cycle *RCL* includes any relays required to support the token pass between successive pairs. It is computed directly from the entries in the distance matrix as follows, and is the sum of the distances between successive nodes in the transmit-order list, i.e.,

$$RCL = \text{dist}_{0,1} + \text{dist}_{1,2} + \text{dist}_{2,3} + \text{dist}_{N-2,N-1} + \text{dist}_{N-1,0}$$

Within the DM, these distances are seen to lie on the diagonal above the main diagonal, and include the first element in the last row, as shown in general in Figure L-4 and as a four-node example is shown in Figure L-5, using the Augmented Distance Matrix data-structure defined in section L.4.4.

² An example may be most useful here, where the node cannot pass the token to the first-hop relay in route to the destination node. In this case the node removes the adjacency from its adjacency list, recomputes the distances to all other nodes and updates its row in the DM. If a path still exists to the original destination then the node sends the token in the direction of this (minimal length) path. If no such path exists then the network is disconnected. In this latter case all nodes that have become disconnected from the remaining connected component containing the token holder are removed from the token ring and the token holder begins to send the token to the next node in the transmit order that is still in the ring. All ring reformation is really done by this mechanism. This procedure applies (almost) equally to an intermediate relay node experiencing this same scenario.

| Row Index | Column Index | 0 | 1 | 2 | 3 | N+1 | N+2 |
|-----------|--------------|----------------------|----------------------|-----------------------|-----------------------|-------------------------|-------------------------|
| 0 | 0 | 0 | 0 | TOL _{0,1} | TOL _{1,1} | TOL _{N-2,1} | TOL _{N-1,1} |
| 1 | 0 | 0 | 0 | TOL _{0,2} | TOL _{1,2} | TOL _{N-2,2} | TOL _{N-1,2} |
| 2 | | TOL _{0,1} | TOL _{0,2} | dist _{0,0} | dist _{0,1} | dist _{0,N-2} | dist _{0,N-1} |
| 3 | | TOL _{1,1} | TOL _{1,2} | dist _{1,0} | dist _{1,1} | dist _{1,N-2} | dist _{1,N-1} |
| N | | TOL _{N,1} | TOL _{N,2} | dist _{N-3,0} | dist _{N-3,1} | dist _{N-3,N-2} | dist _{N-3,N-1} |
| N+1 | | TOL _{N-2,1} | TOL _{N-2,2} | dist _{N-2,0} | dist _{N-2,1} | dist _{N-2,N-2} | dist _{N-2,N-1} |
| N+2 | | TOL _{N-1,1} | TOL _{N-1,2} | dist _{N-1,0} | dist _{N-1,1} | dist _{N-1,N-2} | dist _{N-1,N-1} |

Figure L-4 – DM-elements used to compute the Ring-Cycle Length (general case)

| Row Index | Column Index | 0 | 1 | 2 | 3 | 4 | 5 |
|-----------|--------------|------|------|--------|--------|--------|--------|
| 0 | 0 | 0 | 0 | S(1) | S(2) | S(3) | S(4) |
| 1 | 0 | 0 | 0 | I(1) | I(2) | I(3) | I(4) |
| 2 | | S(1) | I(1) | 0 | D(1,2) | D(1,3) | D(1,4) |
| 3 | | S(2) | I(2) | D(2,1) | 0 | D(2,3) | D(2,4) |
| 4 | | S(3) | I(3) | D(3,1) | D(3,2) | 0 | D(3,4) |
| 5 | | S(4) | I(4) | D(4,1) | D(4,2) | D(4,3) | 0 |

Figure L-5 – DM-elements used to compute the Ring-Cycle Length (example, 4-node network)

The optimization goal for any node joining the ring **shall** be to join in a TOL position that minimizes the resulting RCL.

As the shortest distance between two nodes in the network is 1, it follows that the minimum possible value for RCL in a network is the number of nodes N. Thus, ring reconfiguration for networks with RCL equal to the number of nodes is not required and **shall not** be performed.

L.3 WTRP TOKEN SPECIFICATION

The requirements on WTRP token messages follow.

L.3.1 WTRP Information Exchange Requirements (IER)

The WTRP token-message definitions are based on the Information Exchange Requirements (IERs) defined by Mustafa Ergen, Duke Lee et. al., “Wireless Token Ring Protocol”, University of California, Berkeley, CA 94720, USA. In this reference, access to the wireless channel is controlled through exchange of a token, described and composed as in the following Table:

Table L-2 - Token Field descriptions for Ergen et al's *original* WTRP specification (historically informative only).

| FC | RO | DA | SA | SN | GSN | NS | NON | |
|----|----|----|----|----|-----|----|-----|-----------|
| 1 | 6 | 6 | 6 | 4 | 4 | 4 | 4 | bytes (*) |

* NOTE: the field sizes in bytes given here are based on the original reference's use of the IEEE 802.x MAC-layer addresses and do not apply to WTRP as defined herein. This is a historical comparison only!

where,

- FC (frame control) - identifies the type of packet.
- RO (ring owner) - identifies the current ring owner.
- DA (destination address) - specifies the destination node address of this token.
- SA (source address) - specifies the source node address or the sender node address of this token.
- SN (sequence number) - specifies the age of current RTT token. It is initialized to zero at ring initialization and then incremented by one by every node upon passing the RTT token.
- GSN (generation sequence number) - specifies the current ring age. It is initialized to zero upon ring creation and then incremented by one as the ring owner passes the token.
- NS (new successor) - in an SLS token specifies the new successor for the token receiver. However, this field specifies different information for other token types (as detailed in section L.3.3);
- NON (Number Of Nodes) - for all token types specifies current ring size or current number of nodes in the ring.

In addition to the IER inherited from Ergen et al [1], WTRP support for token-relay within networks with incomplete connectivity requires the following additional IER:

- TOL (transmit order list) – specifies the order of right-to-transmit (RTT) token ownership in the network;
- DM (distance matrix) – specifies the minimum distance between pairs of nodes, measured in hops along the shortest path between them.

L.3.2 WTRP Management Messages and Signalling

The WTRP management messages and signalling techniques consist of an Extended EOW Message embeddable in the Type 6 Management DPDU with optional payload part in accordance with STANAG 5066 Annex C and the sections that follow. The general structure of the WTRP management messages (i.e., tokens) **shall** conform to Table L-3.

Table L-3 – EOW-Token Message (Extended Form) –
Type 6 Management DPDU w/ ID-Management EOW

| Byte/ Bit Num. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Field encoding per S5066 Annex C, as amplified below: |
|---|--|---|---|---|---|---|---|---|--|
| <i>The two-byte Maury-Styles message preamble is not shown;</i> | | | | | | | | | |
| 0 — (Header Length -1) | DPDU Header Type-6 Management DPDU; Sub-Type 15 | | | | | | | | DPDU (Token) Header encoded per Annex L.3.2.1, Table L-4. |
| (Header Length) — Header Length + Payload Length -1 | DPDU Payload (present depending on the token type and usage) | | | | | | | | DPDU (Token) Payload encoded per Annex L.3.2.2, Table L-2. |

L.3.2.1 Type 6.15 (WTRP-Token) Management DPDU Message Header Format

All data fields required of the WTRP header **shall** be encoded as shown in Table L-4 below. The message structure is conformant with all requirements in Annex C of the STANAG 5066 Type 6 Management message in the format of an Extended Engineering Orderwire Message. The EOW field in this Type-6 message format is an integral part of the Type-6 message, rather than a 'piggy-backed' short EOW message that is unrelated to it, and its extended-field elements continue in the DPDU_HEADER_SPECIFIC_PART of the Type 6 message.

Table L-4 - EOW-WTRP-Token Message (Extended Form) -
Type 6.15 Management DPDU w/ ID-Management EOW

| Byte/ Bit Num. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Field encoding per S5066 Annex C, as amplified below: |
|-------------------|---|-------------------------------|---|--|---|-------------------|---------------------|-------|---|
| | The two-byte Maury-Styles message preamble is not shown; | | | | | | | | |
| 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | DPDU_TYPE = 6, per S5066 Annex C; EOW_TYPE = 15 |
| 1 | MSB | | | | FC field ⁽¹⁾ ∈ LSB {Token, Solicit Successor, Set Successor, , ... } | | | | EOW_DATA = WTRP Frame-Control |
| 2 | END_OF_TRANSMISSION (EOT) | | | | | | | | encoded per S5066 Annex C |
| 3 | SIZE_OF_ADDRESS (m ∈ {1 ... 7}) | | | SIZE_OF_HEADER ⁽²⁾ (k = 28) | | | | | m, k in bytes, encoded per S5066 Annex C |
| 3 + m | SOURCE_AND_DESTINATION_ADDRESS | | | | | | | | Field-length = m bytes; encoded perS5066 Annex C; These fields correspond to the WTRP DA and SA fields |
| 4+m | NOT USED | NOT_USED_1 | | | HAS BODY = 0 ⁽⁴⁾ | EXT MSG = 1 | VALID MSG = 1 | ACK | This is the extended form of the ID Mgmt EOW message; encoded per S5066 Annex C; the HAS_BODY field designates if a payload is present. |
| 5+m | MSB - | -- MANAGEMENT FRAME ID NUMBER | | | | | | - LSB | encoded per S5066 Annex C |
| 6+m | PS - PAYLOAD_SIZE (2-bytes, designate the length of any management-message payload) | | | | | | | | WTRP-required field |
| 8+m | RO - RING_ADDRESS (4-bytes, in the address format of STANAG 5066 Annex A) | | | | | | | | WTRP-required field ⁽³⁾ |
| 12+m | SN - SEQUENCE NUM (4-bytes, per the WTRP requirement) | | | | | | | | WTRP-required field |
| 16+m | GSN - GENERATION SEQUENCE NUM (4-bytes, per the WTRP requirement) | | | | | | | | WTRP-required field |
| 20+m | NS - NEW SUCCESSOR ID (4-byte, context-dependent format, per the WTRP requirement) | | | | | | | | WTRP-required field |
| 24+m | NON - NUMBER OF NODES (2-bytes, per the WTRP requirement) | | | | | | | | WTRP-required field |
| CRC_H_1 | CRC_ON_HEADER (16 bits) | | | | | | | MSB | encoded per S5066 Annex C |
| CRC_H_2 | LSB | | | | | | | | |
| | TOKEN PAYLOAD (PAYLOAD_SIZE + 4 bytes) | | | | | | | | WTRP-conditional field (encoded per L.3.2.2) |

Notes:

- (1) field-values corresponding to the enumerated frame-control functions as defined herein
- (2) the given value are based on the use of 4-byte fields are required for SEQUENCE and GENERATION_SEQUENCE, but see the text for further discussion.
- (3) to reduce complexity in message parsing, these fields are encoded as a full fixed-length address fields following the STANAG 5066 rules, regardless of the encoding of the SA and DA fields
- (4) the HAS_BODY field is a new field created in the Type 6 header; see the text for usage.

In the WTRP-token-message definition of Table 1, all fields that are part of the basic Type 6 DPDU message are shown in light or dark grey. Fields required by the Type 6 DPDU

message that meet the WTRP information exchange requirements are shown in white-text-on-dark-grey; new fields for WTRP information exchange requirements are shown in black-on-white.

The design requires that field-sizes related to MAC-layer addresses in the token-ring protocol conform to the MAC-layer over-which the protocol operates. Hence, the RO-, SA, and DA- and NS- field sizes in WTRP **shall** be consistent with STANAG 5066 addressing, and do not contain 6 bytes as in the WTRP protocol from which it was derived. The source and destination address field size **shall** be allowed to be of variable length in accordance with STANAG 5066 Annex C. The RO ring-address field encoding **shall** use a full-length 4-byte address, encoded per STANAG 5066 Annex A, to simplify construction and parsing of the token.

The sequence-number related fields (SN, GSN) **shall** be encoded as four-byte fields as in the WTRP source design³.

The Number of Nodes (NON) field **shall** be encoded in a 2-byte field (N.B. despite the fact that 2^{16} nodes in the network is a prohibitively high number for efficient HF operation, or any wireless network).

The HAS_BODY field is a change to the basic Type 6 header design of Annex C. It **shall** be created by reducing the size of the NOT_USED_1 field specified in previous editions of the STANAG by one bit, and designating that bit as the HAS_BODY field. The HAS_BODY field **shall** be set equal to 0 if the management message has no payload (i.e., body) part, and **shall** be set equal to 1 if it does have a payload.

The two-byte field following the MANAGEMENT FRAME ID field **shall** encode the size-of-payload field, in bytes, to allow station-to-station exchange of Multi-Hop Token-management information in the DPDU-payload body that **may** be included with the Type 6 Management message. The DPDU payload, if present, shall conform to the requirements of section L.3.2.2 below.

L.3.2.2 Type 6.15 (WTRP-Token) Management DPDU Message Payload Format

The payload contents (also called the body-part) of the token, when present, **shall** consist of two segments for the WTRP information exchange requirements plus a 32-bit cyclic-redundancy-check segment for error detection. These segments are:

- a Transmit-Order List (TOL) segment that **shall** have one or more ordered pairs of data elements, each element in the pair encoded in a 4-byte field, as follows:
 - The first data-element **shall** be a STANAG 5066 address, encoded per Annex A, section A.2.2.28.1 (Node Address Encoding for all Primitives), but omitting the size-of-address and group-address fields;

³ With respect to design requirements, it's an open issue whether or not the sequence-number related fields need to be 4-bytes. The four-byte requirement allows WTRP to distinguish 2^{31} unique generations of the token-ring rotation (allowing for field-count rollover), and to support rings with 2^{32} nodes. These seem to be highly unrealistic requirements for a wireless token-ring network based on STANAG 5066 and operation at HF, VHF, or UHF. They've been retained from earlier test implementations in which the overhead represented by these fields was found to be minor compared to other HF signal elements, e.g., interleaver buffers.

- The second data-element **shall** be encoded to support higher-level signalling (e.g., address resolution or dynamic address assignment) when used in conjunction with the Internet Protocol (IP) per the requirements of STANAG 5066 Annex O, and containing the IPv4 address corresponding to the station's STANAG 5066 address.
- a Distance Matrix (DM) segment that **shall** have 4-bit fields that encode the minimum distance (measured as the minimum number of radio links) between pairs of nodes in the network.
- Following the TOL and DM segments **shall** be a 4-byte CRC segment, computed in accordance with the algorithm of Annex C, Section C.3.2.11 with the token payload's TOL and DM segments concatenated together and replacing the C_PDU segment as the operand.

Table L-5 – EOW-Token Message (Extended Form) –
Subtype-15/WTRP Token-Management DPDU Payload format.

| Byte/ Bit Num. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Field encoding per S5066 Annex C, as amplified below: |
|---------------------------------|--|---|---|---|---|---|---|---|--|
| | The two-byte Maury-Styles message preamble is not shown; | | | | | | | | |
| 0 — (Header Length -1) | DPDU Header Type-6 Management DPDU; Sub-Type 15 (RTT Token), Number of Nodes = NON = N Body Length Field = 8*N + Ceil(N²/2) | | | | | | | | DPDU (Token) Header encoded per Annex L.3.2.1, Table L-2. |
| | Ring Transmit-Order List (TOL) (encoded per section L.3.2.2.1) | | | | | | | | Token-Payload Contents for Multi-Hop Token-Relay Algorithm Operation |
| | Node Distance Matrix (DM) (encoded per section L.3.2.2.2) | | | | | | | | |
| CRC_B_1 | MSB | | | | | | | | CRC on Body encoded per Annex C, C.3.2.11 |
| CRC_B_2 | CRC_32 bits ON_PAYLOAD | | | | | | | | |
| CRC_B_3 | | | | | | | | | |
| CRC_B_4 | LSB | | | | | | | | |

L.3.2.2.1 Transmit-Order List (TOL)

The Transmit-Order List (TOL) **shall** be encoded as shown in Table L-6 below.

Table L-6 – Structure of the Transmit-Order List

| Byte/ Bit Num. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Field encoding per S5066 Annex C, as amplified below: |
|----------------|---------------|---|---|---|-----|---|---|-----|--|
| 0 | SOL = {0 1} | 0 | 0 | 0 | MSB | | | | First Node-Address-Pair entry (in network-byte order) in the TOL |
| 1 | | | | | | | | | |
| 2 | | STANAG 5066 Node-Address 1 | | | | | | | |
| 3 | | | | | | | | LSB | |
| 4 | MSB | | | | | | | | |
| 5 | | IP-protocol usage (e.g., IPv4 Node-Address 1) | | | | | | | |
| 6 | | | | | | | | | |
| 7 | | | | | | | | LSB | |
| 8 (N-1) + 0 | SOL = {0 1} | 0 | 0 | 0 | MSB | | | | N-th Node-Address-Pair entry (in network-byte order) in the TOL |
| 8 (N-1) + 1 | | | | | | | | | |
| 8 (N-1) + 2 | | STANAG 5066 Node-Address N | | | | | | | |
| 8 (N-1) + 3 | | | | | | | | LSB | |
| 8 (N-1) + 4 | MSB | | | | | | | | |
| 8 (N-1) + 5 | | IP-protocol usage (e.g., IPv4 Node-Address N) | | | | | | | |
| 8 (N-1) + 6 | | | | | | | | | |
| 8 (N-1) + 7 | | | | | | | | LSB | |

The SOL field **shall** specify whether the designated node is scheduled to invite nodes to join the network. When the SOL = 1, the designated node **may** execute the invitation protocol when its turn to transmit occurs, in accordance with the algorithm of section L.4.5. (N.B. A node designated as the inviter sends the SLS token as a function of the SLS_INTERVAL and other parameters as defined in section L.4.5, and thus it might not send the SLS token every time it has the right to transmit.) The inviting node shall remain so designated until the invitation conditions of L.4.5 are satisfied and it completes the invitation protocol, at which time it sets SOL=0 for its own TOL entry, and sets SOL=1 for its successor's entry in the TOL. The SOL = 1 **shall** be set in accordance with the algorithm designated under Action #2 in Section L.4.8, Receiving the Right to Transmit.

STANAG 5066 addresses in the TOL **shall** be encoded in accordance with STANAG 5066 Annex A, section A.2.2.28.1 as full length (28-bit) addresses; the size-of-address and group-address fields defined in Annex A being replaced by the SOL field and the three padding bits as indicated in Table L-6.

L.3.2.2.2 Distance Matrix

The Distance Matrix (DM) is an $N \times N$ matrix containing values $dist_{i,j}$ such that:

- $DM = [dist_{i,j}]$ = the matrix element in the i -th row and j -th, equal to the minimum-path distance in the network *from* node i *to* node j].

Distance values between a pair of nodes **shall** be encoded as the minimum number of links for all paths in the network connecting the node pair.

A distance value $dist_{i,j}$ is encoded within a 4-bit field, as a decimal value $\{0 \dots 15\}$. A distance value $dist_{i,i}$ on the main diagonal of the matrix is by definition 0 (i.e., the distance from a node to itself is zero).

The DM **shall** be transmitted as a densely packed octet sequence as shown in Table L-7 (for N even) or in Table L-8 (for N odd).

Table L-7 – Distance-Matrix Encoding (when the number of network nodes is even)

| Byte/ Bit Num. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Field encoding as amplified below: |
|-----------------------|-----|--|---|-----|-----|--|---|-----|------------------------------------|
| 0 | msb | $dist_{0,0}$ | | lsb | msb | $dist_{0,1}$ | | lsb | First Row of the Distance Matrix |
| 1 | msb | $dist_{0,2}$ | | lsb | msb | $dist_{0,3}$ | | lsb | |
| | ... | ... | | ... | ... | ... | | ... | |
| $Ceil(N/2)-1$ | msb | $dist_{0,(N-2)}$ | | lsb | msb | $dist_{0,(N-1)}$ | | lsb | |
| $Ceil(N/2)$ | msb | $dist_{1,0}$ | | lsb | msb | $dist_{1,1}$ | | lsb | Second Row of the Distance Matrix |
| $Ceil(N/2)+1$ | msb | $dist_{1,2}$ | | lsb | msb | $dist_{1,3}$ | | lsb | |
| | | | | | | | | | |
| $2 * Ceil(N/2)-1$ | msb | $dist_{1,(N-2)}$ | | lsb | msb | $dist_{1,(N-1)}$ | | lsb | |
| $(k-1) * Ceil(N/2)$ | msb | $dist_{(k-1),0}$ | | lsb | msb | $dist_{(k-1),1}$ | | lsb | k-th Row of the Distance Matrix |
| $(k-1) * Ceil(N/2)+1$ | msb | $dist_{(k-1),2}$ | | lsb | msb | $dist_{(k-1),3}$ | | lsb | |
| | | | | | | | | | |
| $(k) * Ceil(N/2)-1$ | msb | $dist_{(k-1),(N-2)}$ | | lsb | msb | $dist_{(k-1),(N-1)}$ | | lsb | |
| $(N-1) * Ceil(N/2)$ | msb | $dist_{(N-1),0}$ | | lsb | msb | $dist_{(N-1),1}$ | | lsb | Last Row of the Distance Matrix |
| $(N-1) * Ceil(N/2)+1$ | msb | $dist_{(N-1),2}$ | | lsb | msb | $dist_{(N-1),3}$ | | lsb | |
| | | | | | | | | | |
| $N * Ceil(N/2)-1$ | msb | $dist_{(N-1),(N-2)}$ | | lsb | msb | $dist_{(N-1),(N-1)}$ | | lsb | |

N.B.: $Ceil(x)$ is the smallest integer greater than or equal to x

Table L-8 – Distance-Matrix Encoding (when the number of network nodes is odd)

| Byte/ Bit Num. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | Field encoding as amplified below: |
|---------------------------|-----|-----------------------------------|---|-----|-----|---------------------------------|---|-----|-------------------------------------|
| 0 | msb | dist_{0,0} | | lsb | msb | dist_{0,1} | | lsb | First Row of the Distance Matrix |
| 1 | msb | dist_{0,2} | | lsb | msb | dist_{0,3} | | lsb | |
| | ... | ... | | ... | ... | ... | | ... | |
| Ceil(N/2)-1 | msb | dist_{0,(N-1)} | | lsb | msb | dist_{1,0} | | lsb | Second Row of the Distance Matrix |
| Ceil(N/2) | msb | dist_{1,1} | | lsb | msb | dist_{1,2} | | lsb | |
| | | | | | | | | | |
| N - 1 | msb | dist_{1,(N-2)} | | lsb | msb | dist_{1,(N-1)} | | lsb | k-th Row of the Distance Matrix |
| | | | | | | | | | |
| | msb | dist_{(k-1),0} | | lsb | msb | dist_{(k-1),1} | | lsb | |
| | msb | dist_{(k-1),2} | | lsb | msb | dist_{(k-1),3} | | lsb | (k+1)-th Row of the Distance Matrix |
| | | | | | | | | | |
| | msb | dist_{(k-1),(N-1)} | | lsb | msb | dist_{(k),0} | | lsb | |
| | msb | dist_{(k),1} | | lsb | msb | dist_{(k),2} | | lsb | |
| | | | | | | | | | |
| | msb | dist_{(k),(N-2)} | | lsb | msb | dist_{(k),(N-1)} | | lsb | |
| | | | | | | | | | Last Row of the Distance Matrix |
| | msb | dist_{(N-1),0} | | lsb | msb | dist_{(N-1),1} | | lsb | |
| | msb | dist_{(N-1),2} | | lsb | msb | dist_{(N-1),3} | | lsb | |
| Ceil(N ² /2)-1 | msb | dist_{(N-1),(N-1)} | | lsb | msb | 0 = unused | | lsb | |

N.B.: Ceil (x) is the smallest integer greater than or equal to x

Two distance values **shall** be encoded in each octet (except in the last octet when the number of nodes in the network is odd). It follows that:

- the size of the transmitted distance matrix is $Ceil(N^2 / 2)$ octets, where the 'ceiling' function $Ceil(X)$ is the smallest integer greater than or equal to X.

Note the algorithm for octet packing when transmitting the DM: distance-matrix elements are packed first into the high-order bits of an octet, followed by the low-order bits (i.e., the bits in the octet table are filled from left to right and top to bottom as the DM is packed and transmitted). (N.B. The packing algorithm is dense, and may encode in the same octet the distance to the Nth-node of one row of the matrix and the distance to the 1st-node of the next row of the matrix. Any padding bits in the DM are included only as the four lowest-valued bit positions of the last byte of the DM segment.)

L.3.2.2.3 Token Payload Size

The size of the token payload, encoded in the token header, **shall** be the sum of the sizes of the transmit-order-list (TOL) and the distance-matrix (DM) and excludes the size of the CRC_32_ON_PAYLOAD field (in conformance with the practices of Annex C for DPDUs that may have a payload field).

Token-payload size as a function of network size **shall** be as given in Table L-9 below.

Table L-9 – Token Payload Size in bytes as a function of network size

| Network Size = (NON) | Payload Size = | TOL Size | + DM Size |
|-------------------------|----------------------------|----------|----------------------|
| 2 | 18 | 16 | 2 |
| 3 | 29 | 24 | 5 |
| 4 | 40 | 32 | 8 |
| 5 | 53 | 40 | 13 |
| 6 | 66 | 48 | 18 |
| 7 | 81 | 56 | 25 |
| 8 | 96 | 64 | 32 |
| N | $8*N + \text{Ceil}(N^2/2)$ | $8*N$ | $\text{Ceil}(N^2/2)$ |

N.B.: Ceil (x) is the smallest integer greater than or equal to x

L.3.2.2.4 Mapping between TOL Node Addresses and DM Indices

There **shall** be a straight-forward and direct correspondence between STANAG 5066 node addresses in the Transmit-Order List and the row/column indices for distance values in the Distance Matrix, i.e.,:

- The *i*-th STANAG 5066 address in the TOL **shall** be the address of the ‘*from*’ node *i* for distance values $\text{dist}_{i,j}$ and the ‘*to*’ node *i* for distance values $\text{dist}_{j,i}$.

This direct correspondence **shall** be maintained through any manipulation or modification of either the TOL or DM, as, for example:

- insertion of a newly-joined network node into the TOL **shall** result in the insertion of corresponding row and column elements in the DM (with resultant change in its size and format);
- deletion of a node from the network **shall** result in the deletion of the corresponding row and column elements in the DM (with resultant change in its size and format);
- re-ordering of the TOL (e.g., to implement a more efficient transmit sequence) **shall** result in a re-ordering of the corresponding row and column elements of the DM (while preserving its size and format).

L.3.3 Token-Type Definitions

WTRP tokens **shall** be encoded as a Type-6 Management DPDU with an extended EOW-type of 15 (0x0F); thus a DPDU with first-byte (following the Maury-Styles synchronization preamble) value equal to 0x6F denotes a WTRP token.

The token-type **shall** be specified by the Frame-Control field values given in the Table below.

Table L-10 - Frame-Control-Field Values

| | |
|---------------------------------------|----------|
| RTT - right-to-transmit token | 0x0 1 |
| ACK - acknowledgement token | 0x0 2 |
| SLS - solicit successor token | 0x0 3 |
| SET - set successor token | 0x0 4 |
| REL - relayed right-to-transmit token | 0x0 5 |
| DEL - delete token | 0x0 6 |

The FC value **shall** be encoded as the 8-bit EOW-TYPE-specific data part (i.e., the EOW_DATA part) of the DPDU header in accordance with STANAG 5066 Annex C.3.2.2.

Usage of the token types is described in the subsections below.

L.3.3.1 Token: Right-to-Transmit (RTT)

A *right-to-transmit* (RTT) token shall be used to pass the right to transmit along the ring. There are two different types of right-to-transmit tokens:

- *Direct right-to-transmit* (RTT) token (defined in this paragraph), which passes the right to transmit (RTT) to a direct successor without relay, and
- *Relayed right-to-transmit* (REL) token (defined in paragraph L.3.3.5), which passes the right-to-transmit indirectly to a successor through one or more intermediate relay nodes.

A node **shall** transmit data from its Data Transfer Sublayer (DTS) if and only if it has received a right-to-transmit (RTT) token with the destination address set to its own address; i.e., a node relaying the right-to-transmit (REL) token **shall** not transmit its own data, but **shall** only forward the REL token on the path to the designated successor.

RTT-token field values **shall** be encoded in the Type-6/EOW-type-15 Management message in accordance with the requirements of Table L-11.

Table L-11 - RTT Token Field Values

| Field | Length [bit] | Format | Comment |
|-------|----------------------------------|-------------------------|--|
| FC | 8 | unsigned integer | Defined constant for the RTT type: 0x01. |
| DA | 32 | S5066 Address | Destination Address: the node to which to pass the <i>right-to-transmit</i> . |
| SA | 32 | S5066 Address | Source Address: address of station sending this token. |
| RO | 32 | S5066 Address | Ring Owner address. |
| SN | 32 | unsigned integer | Sequence Number: incremented by one every time the <i>right-to-transmit</i> is passed to a <i>successor</i> . |
| GSN | 32 | unsigned integer | Generation Sequence Number: Incremented by every time the <i>right-to-transmit</i> is passed to a <i>successor</i> . |
| NS | 32 | S5066 Address | Not-used |
| NON | 32 | unsigned integer | Number of nodes in the ring. |
| TOL | length in bytes per Table L-9 | format per L.3.2.2.1 | Transmit-Order-List |
| DM | length in bytes per Table L-9 | format per L.3.2.2.2 | Distance Matrix |

L.3.3.2 Token: Acknowledge (ACK)

An *acknowledgement* token **may** be used to confirm the successful delivery of another token, such as an RTT token.

ACK-token field values **shall** be encoded in the Type-6/EOW-type-15 Management message in accordance with the requirements of Table L-12. Note that the ACK token **shall** omit the TOL and DM fields of any token it acknowledges that may contain them.

Table L-12 - ACK Token Field Values

| Field | Length [bit] | Format | Comment |
|-------|-----------------|------------------|--|
| FC | 8 | unsigned integer | Defined constant for the ACK type: 0x02. |
| DA | 32 | S5066 Address | (destination) address of the node to which the ACK is being sent |
| SA | 32 | S5066 Address | Source Address: address of station sending this token. |
| RO | 32 | S5066 Address | Equal to RO-value of the acknowledged token |
| SN | 32 | unsigned integer | Equal to SN-value of the acknowledged token |
| GSN | 32 | unsigned integer | Equal to GSN-value of the acknowledged token |
| NS | 32 | unsigned integer | Not-used |
| NON | 32 | unsigned integer | Equal to NON-value of the acknowledged token |

L.3.3.3 Token: Solicit Successor (SLS)

An SLS token **shall** be used to invite non-members to join the ring. Upon receiving an SLS token, a non-member node **may** solicit the inviting node to join the network and respond with a SET token, in accordance with the WTRP state-machine specification of section L.4.11.

SLS-token field values **shall** be encoded in the Type-6/EOW-type-15 Management message in accordance with the requirements of Table L-13.

The SLS token contains network topology information as known to the inviting node.

Table L-13 - SLS Token Field Values

| Field | Length [bit] | Format | Comment |
|-------|-------------------------------|----------------------|--|
| FC | 8 | unsigned integer | Fixed value defining the SLS type: 0x03. |
| DA | 32 | S5066 Address | Broadcast address: 15.255.255.255 |
| SA | 32 | S5066 Address | Source Address: address of station sending this token. |
| RO | 32 | S5066 Address | Address of the ring owner. |
| SN | 32 | unsigned integer | Not-used |
| GSN | 32 | unsigned integer | Not-used |
| NS | 32 | unsigned integer | the tentative successor specified for the responder of this SLS token. (N.B. this is nominally the successor of the SLS-token originator, as denoted in the TOL, with which this field must agree) |
| NON | 32 | unsigned integer | Number of nodes in the token-originator's ring |
| TOL | length in bytes per Table L-9 | format per L.3.2.2.1 | Transmit-Order-List, as known locally by the inviting node |
| DM | length in bytes per Table L-9 | format per L.3.2.2.2 | Distance Matrix, as known locally by the inviting node |

L.3.3.4 Token: Set Successor (SET)

A SET token **shall** be used to solicit membership in the network. The SET-token shall be used only in reply to a SLS-token inviting solicitors to join.

SET-token field values **shall** be encoded in the Type-6/EOW-type-15 Management message in accordance with the requirements of Table L-14.

The SET token **shall** contain updates to the TOL and DM sent by the inviting node; the soliciting node **shall** insert itself in the TOL and update the DM with its local adjacency information. The soliciting node **may** place itself in the TOL in some position other than as direct successor of the inviting node.

Table L-14 - SET Token Field Values

| Field | Length [bit] | Format | Comment |
|-------|-------------------------------|----------------------|--|
| FC | 8 | unsigned integer | Fixed value defining the SET type: 4. |
| DA | 32 | S5066 Address | Destination address to which the SET-token is sent: (source) address of the SLS token initiator of the solicitation procedure. |
| SA | 32 | S5066 Address | Source Address: address of station sending this token. |
| RO | 32 | S5066 Address | address of the ring owner: Equal to RO-value of the SLS token received. |
| SN | 32 | unsigned integer | Not-used |
| GSN | 32 | unsigned integer | Not-used |
| NS | 32 | unsigned integer | Specifies the new successor Equal to NS-value of the SLS token received. |
| NON | 32 | unsigned integer | unused |
| TOL | length in bytes per Table L-9 | format per L.3.2.2.1 | Transmit-Order-List |
| DM | length in bytes per Table L-9 | format per L.3.2.2.2 | Distance Matrix |

L.3.3.5 Token: Relay Token (REL)

A *relay* (REL) token **shall** be used to relay the right to transmit from the current RTT holder to its successor. A REL token is sent by a station, the *requestor*, to relay the right to transmit data through a third party, the *relayer*, to its unreachable successor, the *target*. The requester and relayer **must** be adjacent in the network topology, i.e., the set of candidate relay nodes for requester node i is the set of all nodes j with adjacency matrix entries $a_{ij} = 1$.

A REL token **shall** be used to pass the right-to-transmit to a successor when the station determines (e.g., based on its own updated Adjacency Matrix or a failure to pass an RTT token) that it can not reach its successor directly.

The requestor **shall** examine its updated connectivity Adjacency and Distance Matrices and identify a relayer with minimum distance to the successor to which to pass the REL-token. Then the designated relayer **shall** forward the REL token to the designated successor if it is adjacent, or, using its own connectivity data, identify an additional relayer to which to pass the REL token.

A REL token may be passed through an arbitrary number of relayers to reach the target station, with the source and destination addresses (i.e., REL.SA and REL.DA) updated as required for each link in the token-relay path. For all REL tokens used along the token-relay path, the new-successor (i.e., *REL.NS*) field **shall** specify the address of the target station designated to receive the right-to-transmit.

When a station has received a REL token for which it is both the destination and the target, it will behave as if it received a RTT token; i.e., a station **shall** declare that it has received the right-to-transmit when it receives a REL token with (REL.DA == this.address) AND (REL.NS == this.address), where this.address is the station's own address, per the register definitions of section L.4.1. This practical limitation prevents the target from taking the right to transmit if it happens to receive a REL token addressed to some other station that is unneeded as relayer.

REL-token field values shall be encoded in the Type-6/EOW-type-15 Management message in accordance with the requirements of Table L-15.

Table L-15 - REL Token Field Values

| Field | Length [bit] | Format | Comment |
|-------|-------------------------------|----------------------|--|
| FC | 8 | unsigned integer | Fixed value defining the REL type: 0x05. |
| DA | 32 | S5066 Address | The address of the station which is requested to relay this token. |
| SA | 32 | S5066 Address | Source Address: address of station sending this token. |
| RO | 32 | S5066 Address | Address of the <i>ring owner</i> . |
| SN | 32 | unsigned integer | Generation Sequence Number: Incremented only by the originator of the REL-token. |
| GSN | 32 | unsigned integer | Generation Sequence Number: Incremented only by the originator and only if this station is the <i>ring owner</i> of the REL-token. |
| NS | 32 | unsigned integer | Specifies where to relay the right-to-transmit to: the relay target. |
| NON | 32 | unsigned integer | Number of nodes in the token-originator's ring. |
| TOL | length in bytes per Table L-9 | format per L.3.2.2.1 | Transmit-Order-List |
| DM | length in bytes per Table L-9 | format per L.3.2.2.2 | Distance Matrix |

L.3.3.6 Token: Delete (DEL)

A *delete* token **shall** be used by a station to direct another station to delete a token from the protocol. Tokens may require deletion when they have incorrect format or represent WTRP-management data that is incorrect.

DEL-token field values **shall** be encoded in the Type-6/EOW-type-15 Management message in accordance with the requirements of Table L-16. Note that the DEL token **shall** omit the TOL and DM fields of any token it deletes that may contain them.

Table L-16 - DEL Token Field Values

| Field | Length [bit] | Format | Comment |
|-------|-----------------|------------------|--|
| FC | 8 | unsigned integer | Defined constant for the DEL type: 0x06. |
| DA | 32 | S5066 Address | (destination) address of the node to which the DEL is being sent |
| SA | 32 | S5066 Address | Source Address: address of station sending this token. |
| RO | 32 | S5066 Address | Equal to RO-value of the token to be deleted |
| SN | 32 | unsigned integer | Equal to SN- value of the token to be deleted |
| GSN | 32 | unsigned integer | Equal to GSN- value of the token to be deleted |
| NS | 32 | unsigned integer | Not-used |
| NON | 32 | unsigned integer | Equal to NON- value of the token to be deleted |

L.4 SPECIFICATION AND PROTOCOLS

WTRP operation is defined in terms of the state machine specified here. Specifications and definitions are provided for states, auxiliary attributes. Data structures for representing the transmit-order-list and distance matrix are recommended, and operators based on these defined data structures defined.

L.4.1 Station registers and flags

While operational, a station **shall** maintain the *registers* and *flags* listed Table L-17. Their contents can be a STANAG 5066 node address, a Boolean flag that could be set, or a counter.

Table L-17 - Station registers and flags

| Register / Flag | Type | Description |
|----------------------------|--------------------------|---|
| <i>this.address</i> | STANAG 5066 Node Address | The station's address. The address has been derived from the operational parameters. |
| <i>this.snCounter</i> | 32 bit unsigned integer | Stores the last received RTT <i>Sequence Number</i> . |
| <i>this.gsnCounter</i> | 32 bit unsigned integer | Stores the last received RTT <i>Generic Sequence Number</i> . |
| <i>this.useRelayFlag</i> | Boolean | Flag which is set if it is equal to true. |
| <i>this.relayAddress</i> | STANAG 5066 Node Address | Address of the station's <i>relayer</i> . This value of this register will be ignored if the <i>this.useRelayFlag</i> is not set. |
| <i>this.slsFlag</i> | Boolean | Flag which indicates if the station should solicit; set in accordance with section L.4.8, Receiving the right-to-transmit. |
| <i>this.txTokenCounter</i> | 8 bit unsigned integer | Counter which is used to count the number of times the same token has been sent. |

L.4.2 Station ring information

The following functions are used in the outbound transition tables to illustrate the stations access to ring information and used to administrate the ring information. Note that while this functional notation is based on object-oriented software design, the STANAG imposes no requirement to use this method of implementation.

ring.getSuccessor()

Return type: STANAG 5066 Node Address

Description: Returns the address of the station's *successor*.

The following **shall** always be true:

ring.getSuccessor() == ring.getSuccessor(this.station)

If a station is not a ring member or has initialized a *self-ring* then the following shall be true:

ring.getSuccessor() == this.address

ring.getSuccessor(<ring member>)

Return type: STANAG 5066 Node Address

Description: Returns the address of the *successor* of the given address of a *ring member*.

ring.getSuccessor().nextHop()

Return type: STANAG 5066 Node Address

Description: Returns the address of the next *station* on the token-relay-path to the station's *successor*. If the *station* and it's *successor* are adjacent in the network topology, then

ring.getSuccessor().nextHop() == ring.getSuccessor() is TRUE.

ring.getSuccessor(<ring member>).nextHop()

Return type: STANAG 5066 Node Address

Description: Returns the address of the next *station* on the token-relay-path to the *successor* of the given address of a *ring member*. If the *<ring member>* and it's *successor* are adjacent in the network topology, then

ring.getSuccessor(<ring member>).nextHop() ==
ring.getSuccessor(<ring member>) is TRUE.

ring.getPredecessor()

Return type: STANAG 5066 Node Address

Description: Returns the address of the station's *predecessor*.

The following **shall** always be true:

ring.getSuccessor() == ring.getSuccessor(this.station)

If a station is not a ring member or has initialized a *self-ring* then the following **shall** be true:

ring.getPredecessor() == this.address

ring.getPredecessor (<ring member>)

Return type: STANAG 5066 Node Address

Description: Returns the address of the *predecessor* of the given argument *ring member*. The given argument shall be a ring member.

ring.isMember(<node address>)

Return type: Boolean (True or false)

Description: Returns true if the given *node address* is a ring member, otherwise false.

ring.getOwner()

Return type: STANAG 5066 Node Address

Description: Returns the *ring owner*.

The following **shall** always be true:

ring.isMember(ring.getOwner()) == true

In a self-ring the following **shall** always be true:

ring.getOwner() == this.address

ring.setOwner(<node address>)

Description: Sets the *ring owner*, the argument **shall** be a *ring member*.

ring.setSuccessor(<new successor>)

Description: Sets the station successor to the given argument *new successor*.

Calling the following function **shall** have the exact same effect:

ring.setSuccessor(this.station, <successor>)

ring.setSuccessor(<ring member>, <new successor>)

Description: Sets the station *successor* to the given argument *new successor*.

If address **A** = *ring.getPredecessor(B)* then after calling
ring.setSuccessor(A, C) the following **shall** be true:

ring.isMember(C) == true
ring.getSuccessor(A) == C
ring.getSuccessor(C) == B
ring.getPredecessor(B) == C
ring.getPredecessor(C) == A

If the *new successor* is already defined as the *ring member*'s successor
this function will have no effect.

ring.setPredecessor(<new predecessor>)

Description: Sets the station *predecessor* to the given argument *new predecessor*.

Calling the following function **shall** have the exact same effect:

ring.setPredecessor(this.station, <predecessor>)

ring.setPredecessor(<ring member>, <new predecessor>)

Description: Sets the station *predecessor* to the given argument *new predecessor*.

After executing this function, the following **shall** be true:

ring.isMember(new predecessor) == true
ring.getPredecessor(ring member) == new predecessor

If the *new predecessor* is already defined as the *ring members*
predecessor this function will have no effect.

ring.drop(<ring member>)

Description: Remove the given *ring member*.

Assume that three stations **A**, **B** and **C** are part of the same ring, where **A**
= *ring.getPredecessor(B)* and **C** = *ring.getSuccessor(B)*.

After calling *ring.drop (B)* the following must be true:

ring.isMember(B) == false
ring.getSuccessor(A) == C
ring.getPredecessor(C) == A

If the given ring member is as well the ring owner, the predecessor of the
given member will be promoted to ring owner.

isFromDifferentRing(<token>)

Description: Returns: *!ring.isMember(token.SA) && ring.isMember(token.DA)*,

where *token.SA* is the token's source address and *token.DA* is the
token's destination address.

L.4.3 Adjacency Matrix updates: the station detection table

Updates to the station's local adjacency matrix are based on the station's detection table (*detTab*), on which the following operators are defined, and used in later descriptions of state-machine operation.

detTab.setHeard(<node address>)

Description: Add a station's address to the connectivity table (i.e., Adjacency Matrix) if it has been heard. If the station is already on the list, it should be updated, otherwise added. In both cases a timestamp is made when the station has been heard. If a station is not updated within the time-out as defined by CON_TABLE_EXPIRE, the station shall be considered as not heard.

detTab.hasHeard(<node address>)

Return type: Boolean (True or false)

Description: Returns true if the station has been updated no longer than the time-out by CON_TABLE_EXPIRE defines, otherwise false.

By definition, the detection table is a representation of the station's column-entry in the adjacency matrix, through the following relationship.

for station j , $a_{ij} = 1$, whenever a $detTab.hasHeard(i) == \text{True}$.

This is the minimum requirement for updates to the station's Adjacency matrix. A node **may** implement the additional optional actions outlined in Section L.2.6 in which reception of ACK tokens may be used to infer the existence of bidirectional links in the network.

L.4.4 Data Structures and Transmit-Order-List Reconfiguration

With the requirement above for Mapping between TOL Node Addresses and DM Indices, this section presents data structures used to describe the Transmit-Order-List Optimization Algorithm (TOLOA). Since internal processing and data structures to implement the TOLOA are implementation choices, the data structures presented here are not mandatory for compliance with this STANAG. But the data structures do maintain the required linkages between the TOL and DM as each are modified by the TOLOA, have use in describing the required performance and calculation methods in optimizing the transmit order list, and are used for that purpose herein.

TOLOA processing is defined with respect to an Augmented Distance Matrix (ADM) that contains both the TOL and DM within a single data structure, as shown in overview in Figure L-6 and in detail in Figure L-7.

$$\text{ADM} = \begin{array}{c|c} \mathbf{0} & \mathbf{TOL^T} \\ \hline \mathbf{TOL} & \mathbf{DM} \end{array}$$

Figure L-6 – Augmented Distance-Matrix Structure

The ADM is an $(N+2) \times (N+2)$ square matrix, with N the number of nodes in the transmit order list. The TOL is contained within the ADM twice, as a column of ordered pairs of the S'5066 Address and IP-management information, and as a row of the same information. The TOL and TOL^T elements in the ADM are transpose representations of the same data, of size $N \times 2$ and $2 \times N$. The DM is contained once in the ADM, as an $N \times N$ submatrix in the lower right area of the ADM. The upper-left 2×2 submatrix of the ADM plays no role in the operation of the TOLOA and, as the values in it are arbitrary, they may be set to zero by definition. A detailed representation of the ADM is shown in Figure L-7.

| Row Index | Column Index | 0 | 1 | 2 | 3 | $N+2$ |
|-----------|--------------|---------------|---------------|----------------|----------------|------------------|
| 0 | 0 | 0 | 0 | $TOL_{0,1}$ | $TOL_{1,1}$ | $TOL_{N-1,1}$ |
| 1 | 0 | 0 | 0 | $TOL_{0,2}$ | $TOL_{1,2}$ | $TOL_{N-1,2}$ |
| 2 | | $TOL_{0,1}$ | $TOL_{0,2}$ | $dist_{0,0}$ | $dist_{0,1}$ | $dist_{0,N-1}$ |
| 3 | | $TOL_{1,1}$ | $TOL_{1,2}$ | $dist_{1,0}$ | $dist_{1,1}$ | $dist_{1,N-1}$ |
| $N+2$ | | $TOL_{N-1,1}$ | $TOL_{N-1,2}$ | $dist_{N-1,1}$ | $dist_{N-1,1}$ | $dist_{N-1,N-1}$ |

Figure L-7 – Augmented Distance Matrix, Detail

TOL-optimization can result in a reordering of the elements in the TOL to minimize the number of token-relays. Any reordering of the TOL can be represented as a sequence of one or more changes in TOL position between two nodes. Linkage between the TOL and its transpose representation in the ADM requires that TOL-reordering in the ADM be accomplished by swapping the pairs of rows and their corresponding columns. These swap operations, between a pair of rows and the corresponding pair of columns, preserve the linkage between the node-address information in the TOL and the node-to-node distance information in the DM; the swap operations additionally preserve the source-destination minimum distance in the DM while reordering the TOL (and TOL^T). An example, with sample values substituted in the ADM, is shown in Figure L-8, as an evolution of the ADM-element values from (a) initial state, (b) after a row-swap and (c) after a column-swap.

| Row Index | Column Index | 0 | 1 | 2 | 3 | 4 | 5 |
|-----------|--------------|--------|--------|----------|----------|----------|----------|
| 0 | 0 | 0 | 0 | $S(1)$ | $S(2)$ | $S(3)$ | $S(4)$ |
| 1 | 0 | 0 | 0 | $I(1)$ | $I(2)$ | $I(3)$ | $I(4)$ |
| 2 | | $S(1)$ | $I(1)$ | 0 | $D(1,2)$ | $D(1,3)$ | $D(1,4)$ |
| 3 | | $S(2)$ | $I(2)$ | $D(2,1)$ | 0 | $D(2,3)$ | $D(2,4)$ |
| 4 | | $S(3)$ | $I(3)$ | $D(3,1)$ | $D(3,2)$ | 0 | $D(3,4)$ |
| 5 | | $S(4)$ | $I(4)$ | $D(4,1)$ | $D(4,2)$ | $D(4,3)$ | 0 |

(a) – ADM Initial State

| Row Index | Column Index | 0 | 1 | 2 | 3 | 4 | 5 |
|-----------|--------------|------|------|--------|--------|--------|--------|
| 0 | | 0 | 0 | S(1) | S(2) | S(3) | S(4) |
| 1 | | 0 | 0 | I(1) | I(2) | I(3) | I(4) |
| 2 | | S(3) | I(3) | D(3,1) | D(3,2) | 0 | D(3,4) |
| 3 | | S(2) | I(2) | D(2,1) | 0 | D(2,3) | D(2,4) |
| 4 | | S(1) | I(1) | 0 | D(1,2) | D(1,3) | D(1,4) |
| 5 | | S(4) | I(4) | D(4,1) | D(4,2) | D(4,3) | 0 |

(b) – ADM after row-swap for S(1) and S(3); altered cells highlighted in grey.

| Row Index | Column Index | 0 | 1 | 2 | 3 | 4 | 5 |
|-----------|--------------|------|------|--------|--------|--------|--------|
| 0 | | 0 | 0 | S(3) | S(2) | S(1) | S(4) |
| 1 | | 0 | 0 | I(3) | I(2) | I(1) | I(4) |
| 2 | | S(3) | I(3) | 0 | D(3,2) | 0 | D(3,4) |
| 3 | | S(2) | I(2) | D(2,3) | 0 | D(2,1) | D(2,4) |
| 4 | | S(1) | I(1) | D(1,3) | D(1,2) | 0 | D(1,4) |
| 5 | | S(4) | I(4) | D(4,3) | D(4,2) | D(4,1) | 0 |

(c) row-swapped ADM after column-swap for S(1) and S(3); altered cells highlighted in grey.

Figure L-8 – ADM Row-Swap and Column-Swap Operations: Example

Note that after a row-and-column swap, both the TOL and TOL^T elements contain the re-ordered transmit order list and the DM element has zeroes on its main diagonal, as required for proper structure in the ADM. Note too that after a row-swap alone (or column-swap alone) neither of these conditions apply; both a row-swap and column swap is required to preserve the ADM structure and element linkages when reordering the TOL; and must occur together before another swap operation is performed. Offered without proof is the fact that the order of operations in a row-and-column swap does not affect the outcome.

Notation for additional ring information that should be maintained and operands upon that information is as follows:

ring.ADM();

token.ADM();

Return type: a (N+2)x(N+2) Matrix object, or null

Description: Returns the ring's (or token's) Augmented Distance Matrix, or null, if one is not defined.

adm.TOL()
ring.ADM().TOL()
token.TOL()

Return type: an Nx2 Matrix object, or null

Description: Returns the ADM's (or ring's or token's) Transmit-Order-List (TOL), STANAG 5066 addresses and solicit tags in the first column and related IP-network information in the second column, or null, if the TOL is not defined.

adm.DM()
ring.ADM().DM()
token.DM()

Return type: an NxN Matrix object, or null

Description: Returns the ADM's (or ring's or token's) Distance Matrix, or null, if one is not defined.

ring.ADM().RCL()

Return type: integer;

Description: Returns the ring-cycle length of the current transmit order list.

adm1.equals(adm2)
tol1.equals(tol2)
dm1.equals(dm2)

Return type: Boolean, true if the respective object1 equals object2, otherwise 0.

Description: Determines if the respective objects *obj1* and *obj2* are equal; Transmit-order-lists are equal if they are the same length and if the sequence of STANAG 5066 addresses they contain differ only by a cyclic-shift of all elements (this is a looser definition than strict equality of all elements in the list); distance matrices are equal if all their elements are equal; augmented distance matrixes are equal if their TOL are equal and the associated embedded distance matrices are equal.

Sample Usage the statements *ring.TOL().equals(RTT.TOL())* and *ring.DM().equals(RTT.DM())* both return true if the station's stored values of the transmit order list and distance matrix are the same as those received in the current right-to-transmit token; the test is made to determine if an attempt to optimize the transmit order list may be made.

L.4.5 Transmit-Order-List Optimization Algorithm (TOLOA)

Transmit-order-list modification shall be performed for the various reasons as outlined in section L.2.7.

For a new ring member:

- the joining station shall compose an augmented distance matrix using the TOL and DM received from the inviting station;
- the joining station shall insert itself into the ADM as the inviting node's successor, updating the TOL and DM with the joining station's local adjacency information and recomputing the DM
- if the RCL of the resulting TOL is equal to the number of nodes in the ring, then no further optimization is possible, other the joining node shall attempt to modify the TOL to produce an RCL of minimum size;

For an unreachable (i.e. a lost or dropped) station:

- the station that detects the loss shall remove the affected station from the TOL and DM, and
- update the TOL and DM to minimize the RCL;

For RCL minimization under stable ring topologies:

- a station determines that there is an opportunity for TOL optimization when:

$(ring.ADM().TOL().equals(RTT.TOL()) == true) \text{ AND}$
 $(ring.ADM().DM().equals(RTT.DM()) == true) \text{ AND}$
 $(ring.ADM().RCL() > N)$

i.e., when the TOL and DM have remained unchanged for at least one cycle and the ring-cycle length is greater than the number of nodes in the ring.

- a station that determines that there is an opportunity for TOL optimization may reorder the TOL and DM so that the resulting RCL is minimized.

L.4.6 Modified Nearest Insertion Method (MNIM) for RCL Optimization and Ring Reconfiguration

Specific algorithms for computing the minimum-length RCL are implementation choices and not specified herein; however a recommended method for computing a TOL with a minimum-length (or near minimum-length) RCL is a modification of the Nearest Insertion Method (MNIM) described by Gregory L. Heileman, *Data Structures, Algorithms and Object-Oriented Programming*, University of New Mexico ISBN 0-07-027893-8; the MNIM is algorithm used to find good but approximate solutions to the well known Travelling Salesman Problem (TSP) in operations research that is related to RCL optimization.

The MNIM performs, in effect, an optimal "virtual-joining" sequence (VJS) for the network, with the goal of obtaining a TOL with a smaller RCL than the current ring. The virtual

joining sequence used to construct the TOL starts with a single node (the node performing the VJS), and adds to the current TOL a randomly chosen node⁴ of those nodes that have not been added.

With $TOL_k = (n_0, n_1, n_2, \dots, n_k)$ the state of the transmit-order list after k nodes have been added, a randomly chosen node n_j from the set of remaining nodes that have not be added **shall** be inserted between the two nodes n_i and $n_{((i+1) \bmod k)}$, as illustrated in Figure L-9 – Insertion of node j into the TOL, to minimize the increase in RCL, i.e., that minimizes

$$\Delta RCL_i = \text{dist}(n_i, n_j) + \text{dist}(n_j, n_{((i+1) \bmod k)}) - \text{dist}(n_i, n_{((i+1) \bmod k)})$$

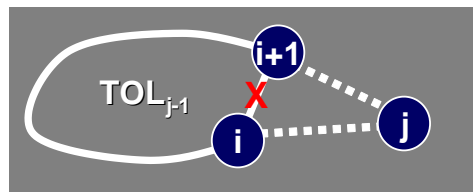


Figure L-9 – Insertion of node j into the TOL

The MNIM algorithm continues inserting nodes in the TOL, in a VJS that minimizes the incremental RCL for each joining node, until there are no new nodes to join.

If, following a TOL optimization attempt, the RCL is less than the original TOL, the new TOL is forwarded to the ring when the node completes its transmission and passes the RTT token containing the new TOL and DM to its successor.

Note that the distances used in computing the RCL are those between the nodes including relays. Rather than looking at the TOL Optimization Algorithm as defining an optimum virtual-joining-sequence, an alternate way of looking at the TOLOA is to consider that it:

- converts the (possibly incomplete) adjacency matrix of unit-length links to a complete distance matrix with links, maintaining also the minimum-length-relay-path for each pair of nodes;
- solves the travelling salesman problem (TSP) of finding the minimum length tour through the distance matrix.

The state-transition descriptions of section L.4.11 uses the following object notation to indicate that the TOLOA has been executed on the ring's internal data structures:

ring.TOLOA()

Description: Updates the station's DM, TOL, and ADM, and computes the minimum-length-relay-path for each pair of nodes in the DM.

⁴ Heileman describes a different approach of choosing the next node, by performing a search of which of the nodes is nearest to the existing tour and this method is that for which the Nearest Insertion Method is named; the unmodified NIM adds additional computational complexity and was found to generate some pathological solutions that are avoided with the modification described here where the next node to be added is chosen randomly.

L.4.7 Solicitor list (i.e. the list of responders to SLS tokens)

slsList.isEmpty()

Return type: Boolean (True or false)

Description: Returns true if the solicit list contains no entries.

slsList.add(<node address>) ()

Description: Add *node address* to solicit list.

slsList.clear()

Description: Remove all entries from the solicit list.

slsList.pickRandom()

Description: Returns a random selected node from the soliciting list.

L.4.8 Receiving the right-to-transmit

Action # 1: Under normal ring operation, the right-to-transmit **may** be received in the states: MON, RLY and IDL. If a station received a RTT-token with $RTT.DA == this.address$ or a REL-token with $REL.DA == this.address$ and $REL.NS == this.address$, the right to transmit has been received.

Action # 2: If the right-to-transmit is received a station **shall** determine if the solicit-flag (*this.slsFlag*) has to be set, and an invitation to join the network sent to any potential solicitors. The solicit flag **shall** be set (true) when following equation is true:

$$(token[SN] \bmod (\Delta \times ringSize)) \bmod (\Delta + 1) == 0$$

where:

- *token[SN]* equals the sequence number from the received right-to-transmit token (i.e., equal to $RTT.SN$ or $REL.SN$, depending on whether or not the right-to-transmit was passed directly or relayed).
- $\Delta = SLS_INTERVAL \times ringSize$;
- *ringSize* is the number of nodes in the ring, and may be obtained by a variety of (what should be) equivalent methods, e.g., based on the difference between the current and last RTT token-sequence numbers, by direct extraction of the number-of-nodes field *token.NON* in the received token, or counting the number of entries in the TOL.
- *SLS_INTERVAL* is the scalar tuning parameter, per section L.5, that controls the frequency at which solicit invitations are offered in the network.

If the solicit flag must be set, it is also marked in the SOL field (see Table L-6) of the node's entry in the TOL in the token payload.

Action # 3: The Generation Sequence Number **shall** be used to detect if the *ring owner* has dropped out of the network. When the following condition is true:

(ring.getOwner() != this.address) && (this.gsnCounter == RTT.GSN)

the station **shall** promote itself as the new *ring owner*: *ring.setRingOwner(this.address)*).

Action # 4: The next step **shall** be to update the counters *this.snCounter* and *this.gsnCounter*; update of these counters shall be done **after** the value of *this.slsFlag* has set.

this.snCounter = token.SN

this.gsnCounter = token.GSN

Action # 5: The station **shall** update the Distance Matrix DM and Transmit Order List TOL it received from its predecessor in preparation for passing the right to transmit to its successor. This update may result in changes to the either the DM or TOL.

L.4.9 Passing the right-to-transmit

Action #: If the successor has been heard (*this.hasHeard(this.getSuccessor())*) the right-to-transmit **shall** be forwarded using a RTT-token, otherwise a REL-token **shall** be used.

Action #: The RTT-token **shall** have the following setting:

- *RTT.DA = this.getSuccessor()*
- *RTT.SN = this.snCounter + 1*
- If the station is the *ring owner*: *RTT.GSN = this.gsnCounter + 1* otherwise *RTT.GSN = this.gsnCounter*.

The REL-token **shall** have the following settings:

- *REL.DA = this.getPredecessor()*
- *REL.NS = this.getSuccessor()*
- If the station is the *ring owner*: *REL.GSN = this.gsnCounter + 1* otherwise *REL.GSN = this.gsnCounter*.

L.4.10 Overall State Diagram

Figure L-10 below shows the complete state diagram of WTRP. Each state is described in detail in the following sections. The set of states are divided into three subsets that correspond to a station unaffiliated with any token ring, a station soliciting membership in a ring that has invited it to join, and a station in operation within an active token ring.

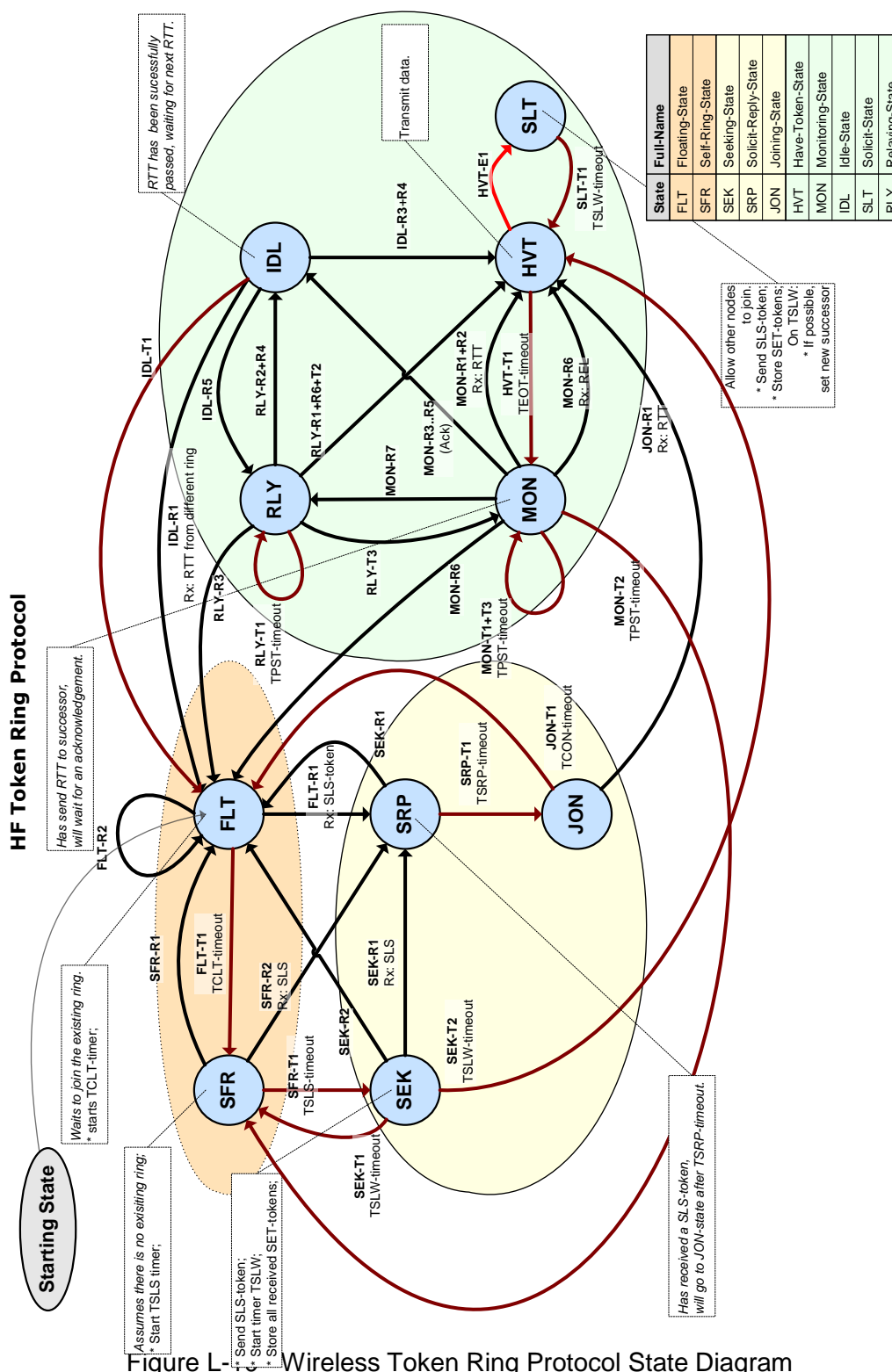


Figure L- Wireless Token Ring Protocol State Diagram

L.4.11 State-Machine Specification

This section and its subsections specify the actions of the WTRP state machine. For every state there are state-entry actions and an outbound-transition table defined. When entering a state, a station first **shall** execute the state-entry actions and then it **shall** wait for an event to occur which triggers one of the transitions to the next state defined in the outbound-transition table. There are two types of event that trigger state transitions:

- Events caused by timeouts; a timeout event is prefixed with the label “Exp” (i.e., for expiry or expired), followed with the name of the timer causing the timeout.
- Events caused by received data; this event is prefixed with the label “Rcv” (i.e., for ‘received’), followed with the type of D_PDU received. The D_PDU type can be ‘D_PDU’, which can be any of the D_PDU types; ‘token’ which can be any of the defined token D_PDU’s; or a specific token annotated by: RTT, REL, SLS etc.

Only one transition rule **shall** be executed after an event: this **shall** be the first and only the first transition for which the condition is met as the state-machine logic examines the outbound-transitions in the order in which they are listed in the table.

L.4.11.1 Common state specification

On every received D_PDU the method *detTab.setHeard(<node address>)* **shall** be called where the argument is the source address (SA) of the received D_PDU. This action is not specified in the transition table, since it applies to receive processing for every D_PDU, in every state.

L.4.11.2 Floating State (FLT)

The *Floating State* is the WTRP start-up state and the state in which a station is not part of a ring and waits to join a ring. The floating state is a *listening-only* state. A station **shall** stay in the FLT state until there is a joining opportunity (i.e., an SLS token is received inviting the station to solicit membership in the ring) or the TCLT timer expires.

The TCLT timer is used to determine when a station **will** assume that there is no existing ring present. If this timer times out (i.e., expires) the station **shall** proceed to *Self Ring State* (SFR).

If the station receives a SLS token it **shall** transit to the *Joining State* (JON).

L.4.11.2.1 Floating State entry actions

On this state entry the station **shall** execute the following actions:

- Reset ring information: Clears all ring members but the station itself.
- Clear the relay flag (i.e., the station reverts to a mode where it will not relay-tokens).

- Start the TCLT timer.

L.4.11.2.2 Floating State outbound transitions

Outbound transitions from the FLT state are shown in the table below. The most significant outbound transition is to the Solicit-Reply (SRP) State, which occurs when a node receives an invitation to join a ring. For other transitions, the node either remains in the FLT state waiting to receive invitations or transits to the SFR state where it will wait before deciding to send its own invitations for nodes to join its ring.

Table L-18 - FLT outbound transitions

| transition | event | Condition | Action | next state |
|------------|-----------------|------------------------|--|------------|
| FLT-R1 | Rcv: SLS | RCV.DA == this.address | detTab.hasHeard(SLS.SA) ring.setPredecessor(SLS.SA) ring.setSuccessor(SLS.NS) this.setRingOwner(SLS.RO) this.useRelayFlag = !detTab.hasHeard(SLS.NS) | SRP |
| FLT-R2 | Any other token | | detTab.hasHeard(D_PDU.SA) Restart TCLT Timer | N/A |
| FLT-T1 | Exp: TCLT | | | SFR |

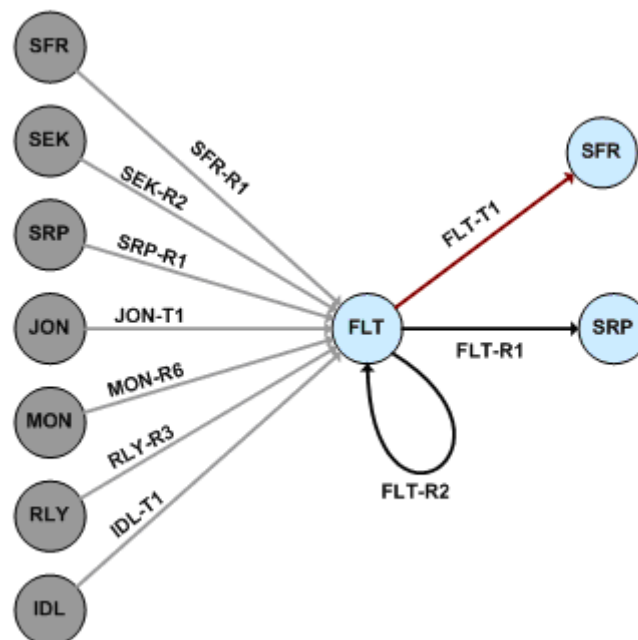


Figure L-11 - FLT Outbound transitions

L.4.11.3 Self Ring State (SFR)

In this state a station has concluded there is no ring to join and therefore will setup a new ring by itself. This condition is called the *self ring*. This state is like the Floating State (FLT) in that it is a listening-only state.

On state entry the TSLS timer **shall** be set to a random timeout value, which is used to avoid collisions with any other station trying to setup a ring. If this timer (i.e., if TSLS) times out the station **shall** transit to the *Seeking State* (SEK), sending an invitation to nodes to join its network as an outbound action during the transition.

While waiting for the TSLS timeout, an SLS token **might** be received from another station in SEK state (i.e., the station receives a SEK token where SEK.SA == SEK.NS); in this case the station **shall** transit to the *Solicit Reply State* (SRP), where it will respond to the invitation.

If any other D_PDU or token is received the station **shall** go to the *Floating State* (FLT).

L.4.11.3.1 Self Ring State (SFR) entry actions

On SFR-state entry a station **shall** start its TSLS Timer with a random time-out value.

L.4.11.3.2 Self Ring State (SFR) outbound transitions

Outbound transitions for the SFR state are shown in the table below. Transitions from the SFR state are triggered: when the station receives an invitation from another node and then will reply; when the station receives any other DPDU from another station and thus should wait for an invitation; or when the station hears nothing for some time and then sends its own invitation.

Table L-19 - SFR-State Outbound-Transition Table

| transition | event | condition | action | next state |
|------------|-----------|------------------|---|------------|
| SFR-R2 | Rcv: SLS | SLS.NS == SLS.SA | ring.setSuccessor(SLS.SA) this.setRingOwner(SLS.SA) | SRP |
| SFR-R1 | Any D_PDU | | | FLT |
| SFR-T1 | Exp: TSLS | | SEND: SLS where SLS.DA = <i>broadcast address</i> [N.B. – this invitation to join is effectively an entry action to the SEK state] | SEK |

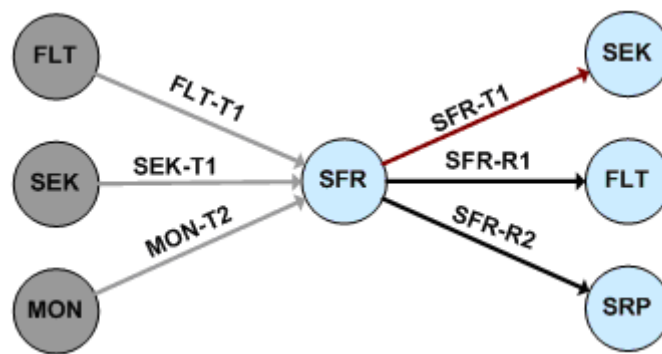


Figure L-12 - Outbound transitions from SFR state

L.4.11.4 Seeking State (SEK)

The *Seeking State* is a state in which a station in a *self ring*, the seeking node, broadcasts an SLS token inviting new ring members and listens for replies from solicitors until the TSLW timer times out. The initial TOL and DM for the self ring have a single entry, the seeking node. Any solicitor **will** reply by sending a SET token with updated TOL and DM containing the seeking node and soliciting node. For each solicitor that the seeking node hears, the solicitor's SLS reply **shall** be recorded in the *solicitor list*. When the TSLW timer expires a solicitor **shall** be selected at random from the *solicitor list* and will be set as the station's successor⁵. The station will then update its TOL and DM for the two-node ring and proceed to the HVT state.

If an SLS token is received before the TSLW timer expires, then another ring (possibly a self-ring) is active within radio range of the station. The station in this case **shall** cease its invitations to other nodes to join its self-ring and **shall** transit to the SRP state, with intent to join the ring it has detected.

Reception by a station when it is in the SEK state of any token other than a SET or SLS token shall force the station into the FLT state, as reception of such tokens is an indication there is an active ring within radio range, the self-ring condition no longer applies and the station should not be soliciting to form its own ring.

L.4.11.4.1 Seeking State entry actions

On state entry from the SFR state the station has broadcast an SLS Solicit Successor token, and shall start the TSLW timer waiting for replies.

L.4.11.4.2 Seeking state outbound transitions

Outbound transitions from the SEK state are shown in the table below. As noted above, reception of SET tokens does not force an outbound transition; the SET tokens are stored for later processing and the station remains in the SEK state. Reception of an SLS token forces a transition to the SRP solicit-reply state and reception of any other token forces a transition to the FLT floating

⁵ This simple action in processing solicitations when the ring is initially formed between two nodes should be compared to the more generalized response to a solicitation by a station in the SLT state in a larger ring.

state; both of these triggering events invalidate the station's assumption that it can form its own ring but with different responses by the station. As long as the station receives only SET tokens, it will continue to store them until the TSLW timer expires, at which point the station picks a successor and proceeds to the HVT state and subsequent operation in a two-node network. If no responses of any kind are heard after waiting for replies, the station returns to the SFR self-ring state.

Table L-20 - SEK state outbound transitions

| transition | event | condition | action | next state |
|------------|-----------------|----------------------------|---|------------|
| SEK-A1 | Rcv: SET token | SET.DA == this.address | slsList.add(SET) detTab.setHeard(SLS.SA) | N/A |
| SEK-R1 | Rcv: SLS token | | slsList.clear() ring.setRingOwner(SLS.RO) ring.setPredecessor(SLS.SA) ring.setSuccessor(SLS.NS) ring.setTOLandDM(SLS.payload) | SRP |
| SEK-R2 | Any token | | | FLT |
| SEK-T1 | Exp: TSLW timer | slsList.isEmpty() == false | ring.setSuccessor(slsList.pickRandom()) slsList.clear() | HVT |
| SEK-T2 | Exp: TSLW timer | slsList.isEmpty() == true | | SFR |

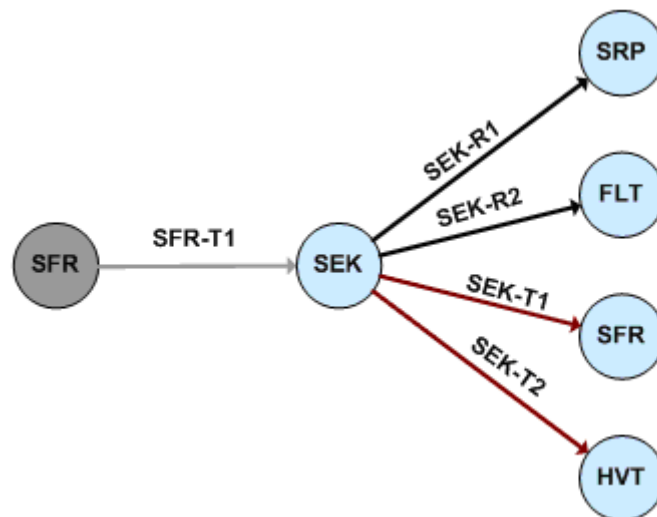


Figure L-13 - Outbound transitions from SEK state

L.4.11.5 Solicit Reply State (SRP)

In the *Solicit Reply State* a station will reply to an invitation to join the network (i.e., to an SLS token received from another node) and attempt to join the ring.

The TSRP timer **shall** be started on state entry to determine the moment to reply.

While waiting in the *Solicit Reply State* for the TSRP time to expire, other SET tokens **might** be received and should be ignored, as they could be expected as replies by other stations to the invitation.

Receipt of other message traffic indicates that the channel is not clear during the protocol's invite-and-solicit dialog for new ring members (an error condition in the protocol). The error condition is best handled by having all solicitors return to a known state (the FLT state) in which they do not transmit.

The soliciting station **shall** reply by sending a SET token to the inviting station, the SLS-token originator, which contains an updated TOL and DM payload with the soliciting station's connectivity information. The updated TOL and DM **may** place the soliciting station as the *direct successor* to the inviting station, or it **may** place the soliciting station as a *delayed successor* in some other location in the TOL that minimizes the resulting ring-cycle length.

L.4.11.5.1 State entry actions:

Set the TSRP timer with the following timeout value:

$pickedSlotNum = pickRandom(numSolicitSlots)$

$timeout[TSRP] = pickedSlotNum * solicitSlotTime$

L.4.11.5.2 Solicit Reply State Outbound Transitions

Outbound transitions from the solicit transition can be categorized as error recovery or incremental success in joining the network. If any token other than a SET token is heard, the station effectively declares error and transitions to the FLT state, where it will wait for another invitation to join. If the channel remains clear of unexpected traffic, the node sends its solicitation (a SET token) to join the network and transits to the JON state where it will wait to see if its solicitation succeeded.

Table L-21 - SRP outbound transition table

| transition | Event | condition | action | next state |
|------------|------------------|-----------------------------|--|------------|
| SRP-R1 | Rcv: token.!=SET | | | FLT |
| SRP-T1 | Exp: TSRP | slsList.empty() == false | Send SET token where: SET.SA = this.addres SET.DA = ring.getPredecessor() SET.RO = ring.getRingOwner() SET.NS = ring.getSuccessor() SET.TOL= updateTOLwithOwnData() SET.DM = updateDMwithOwnData() | JON |

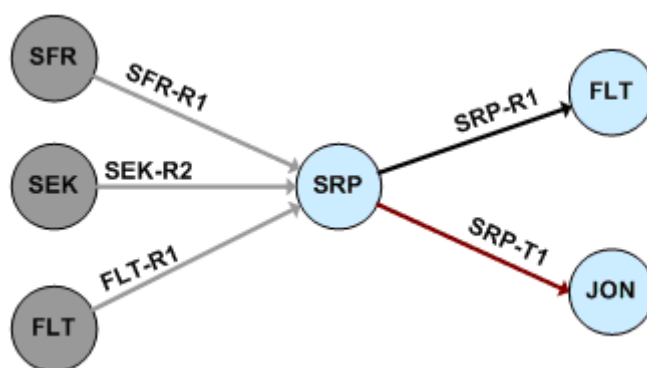


Figure L-14 - Outbound transitions from SRP state

L.4.11.6 Joining State (JON)

In the *Joining State* a station has replied to solicitation opportunity by sending a SET token and is now waiting for the *right-to-transmit token*.

The test that the soliciting station has successfully joined the ring is straightforward; the soliciting station receives an RTT token. But with the options that the soliciting station may insert itself in the updated TOL as either a *direct successor* or a *delayed successor*, the length of time the soliciting station should wait to receive the RTT is variable, as a function of the number of stations D in the TOL between the inviting and soliciting stations; if D = 0, the soliciting station is a direct successor and if D > 0 it is a delayed successor.

L.4.11.6.1 Joining State entry actions

Start the TCON timer, computed in accordance with section L.6.4, for the value of D corresponding to the soliciting station's placement in the TOL.

L.4.11.6.2 Joining State outbound transitions

Outbound transitions for the JON state are shown in the table below. A station either succeeds with its solicitation to join, and transits to the HVT state as a ring member and able to transmit, or fails and transits to the FLT state where it will wait for the next invitation from a ring member (or its own declaration that there is no ring present and eventual transition to the SFR self-ring state).

Table L-22 - JON-State Outbound-Transition Table

| transition | event | condition | action | next state |
|------------|-----------------|------------------------|---|------------|
| JON-R1 | Rcv: RTT token | RTT.DA != this.address | | N/A |
| JON-R2 | Rcv: RTT token | RTT.DA == this.address | this.slsFlag = false this.snCounter = RTT.SN this.gsnCounter = RTT.GSN ring.setPredcessor(RTT.SA) ring.setRingOwner(RTT.RO) | HVT |
| JON-T1 | Exp: TCON timer | | | FLT |

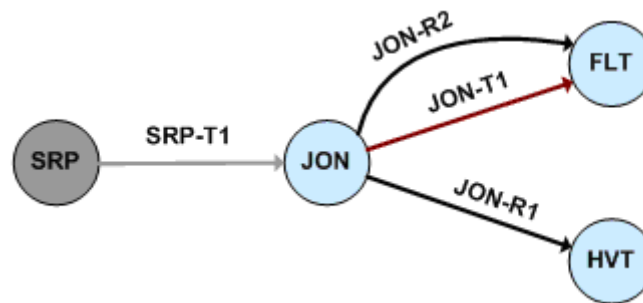


Figure L-15 - Outbound transitions from JON state

L.4.11.7 *Have Token State (HVT)*

In the *Have Token State* a station has received the *right to transmit token* (RTT) and is therefore at the moment the channel “owner”, allowed to send D_PDU as long as the length of its transmission does not exceed the *maximum time to transmit*.

L.4.11.7.1 Have Token State entry actions

On entry to the HVT state, the station **shall** compare the TOL and DM data received in the RTT to its local TOL and DM, and **shall** make any required changes or optimizations to the TOL and DM.

If the solicit-flag is set (*this.slsFlag*) the station **shall** transit directly to the SLT state. After the SLT-state has completed the station returns to the HVT state **shall** proceed with the following entry actions⁶.

On entry to the HVT state a station **shall** first determine if it has *user data* (i.e., data from DTS layer) to send or not. If not it **shall** proceed with passing the right-to-transmit to its successor; otherwise, it **shall** start to prepare the user-data for transmission. Transmit preparation is done by accepting a set of D_PDU from the DTS layer. The station **shall not** accept more D_PDU from the DTS-layer than it can send in one transmission without exceeding the *maximum transmit time* (MTT), including any MAC-layer DPDU (i.e., tokens). The last D_PDU sent in the transmission opportunity **shall** always be a *right-to-transmit* token [N.B. see L.4.9, “Passing the right-to-transmit” for more details].

L.4.11.7.2 Have Token State outbound transitions

Outbound transitions from the HVT state are shown in the table below.

The conditional transition to the SLT state to invite new ring members has been noted above.

Transitions from the HVT state to the MON state **shall not** be allowed until the transmission has completed. If the station is using a communications interface that buffers the transmit data, an *end of transmit* (TEOT) timer **shall** be set to generate the transition event to the MON state after the data has been sent over the physical medium; the station **may** additionally use an interrupt signal generated by the communications interface to signal the same event.

Table L-23 - HVT-State outbound transition table

| transaction | event | condition | action | next state |
|-------------|-------------|---|-----------------------------|------------|
| HVT-E1 | State entry | <i>this.slsFlag</i> == True | <i>this.slsFlag</i> = False | SLT |
| HVT-T1 | Exp: TEOT | completed transmitting data (i.e., any user data and tokens) | | MON |

⁶ This transition is labelled “HVT-E1”. See also L.4.8, “Receiving the right-to-transmit”.

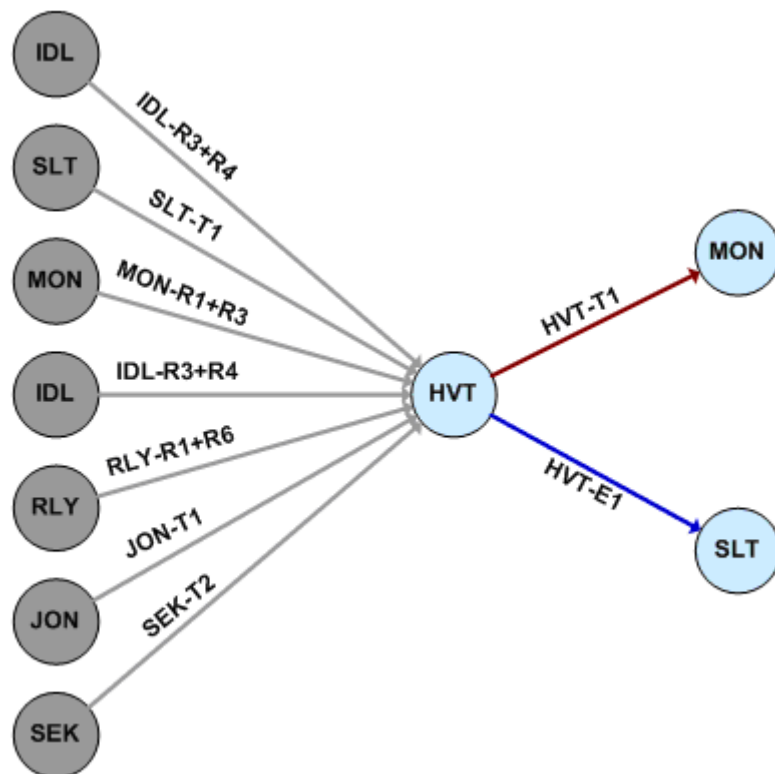


Figure L-16 - Outbound transitions from HVT state

L.4.11.8 Monitoring State (MON)

The *Monitoring State* is a state in which a station has finished transmitting data, passed a RTT token to its successor (or, alternatively, passed a REL token to the next-hop node on the path to its successor) and is waiting for an acknowledgement. An acknowledgement can be any D_PDU indicating that the successor or next-hop node must have received the *right to transmit token* (i.e., RTT or REL token), including the explicit ACK token. Any DPDU received in the MON state with source address equal to the station's successor are taken as implicit acknowledgement that the successor received the RTT token and has taken control of the radio channel.

At expiration of the TPST timer the token (RTT or REL) is considered to be lost (i.e., not received by the next-hop node on the path to the successor). The station shall resend it for a number of times until the MAX_TOKEN_PASS count has been reached.

If the MAX_TOKEN_PASS count has been reached the station declares the next-hop node unreachable, deletes the next-hop node from further consideration and from the station's knowledge of the network, reorganizes the network data structures to pick a new next-hop node, and sends the *right-to-transmit token* to that node. This process continues until the *right-to-transmit token* is either successfully passed or attempts to pass the *right-to-transmit token* to all other nodes in the network have failed and the station collapses to a self-ring.

A station in monitoring state will try to process all *relay requests* addressed to it as the *relator*. A *relay request* is identified by the station when it receives a *relay token* (REL) with the destination address (REL.DA) set to the station's address (i.e., this.address).

L.4.11.8.1 Monitoring State entry actions

On state entry the station **shall** start the TPST timer, setting the time the station shall wait for receipt of an explicit or implicit acknowledgement of the *right-to-transmit*.

For any entry other than a self-transition, i.e, if the preceding state is not the MON state, the *transmit-token-counter* (this.txTokenCounter) **shall** be set to 1. Self-transitions may or may not reset the *transmit-token-counter*, as noted below.

L.4.11.8.2 Monitoring State outbound transition table

Outbound transitions from the MON state are shown in the table below.

Transitions representing a successful pass of the *right-to-transmit* (RTT or REL token) are to the HVT or IDL states.

Transitions representing failure to pass the *right-to-transmit* are to the FLT or SFR states, with transition to the SFR state occurring after attempts to pass the *right-to-transmit* to all known nodes in the network have failed.

A self-transition back to the MON state is for error-recovery following a failed attempt to pass the *right-to-transmit*, to allow for another attempt and retransmission of the token.

On re-entry to the MON state, the *txTokenCounter* may or may not be reset to 1. The *txTokenCounter* **shall not** be reset until MAX_TOKEN_PASS attempts to pass the right-to-transmit to the next-hop node have been made and failed, at which point the next-hop node is declared unreachable. The unreachable node is then deleted from the station's database (i.e., from the station's detection table, distance matrix, and any other internal data structures that depend on these), a new TOL is generated, the next-hop node to the station's (possibly new) successor identified, the *txTokenCounter* reset to 1, and a right-to-transmit token (either a RTT or REL token, as appropriate) sent to the next hop.

Table L-24 - MON-State Outbound-Transition Table

| transition | event | Condition | Action / Comments | next state |
|------------|-------------------|---|---|------------|
| MON-R1 | Rcv: RTT | RTT.DA == my.address | ring.setPredecessor(RTT.SA) See: L.4.8, "Receiving the right-to-transmit". | HVT |
| MON-R2 | Rcv: REL | REL.DA == this.address AND REL.NS == this.address | See: L.4.8, "Receiving the right-to-transmit". | HVT |
| MON-R3 | Rcv: ACK | RTT.SA == ring.getSuccessor() RTT.DA == my.address | (Explicit acknowledgement) | IDL |
| MON-R4 | Rcv: RTT | ring.isMember(RTT.SA) | setSuccessor(RTT.SA, RTT.DA) (Implicit acknowledgement) | IDL |
| MON-R5 | Rcv: Any D_PDU | ring.isMember(D_PDU.SA) | (Implicit acknowledgement) | IDL |
| MON-R6 | Rcv: RTT | isFromDifferentRing(RTT) | | FLT |
| MON-R7 | Rcv: REL | REL.DA == this.address AND ring.isMember(REL.NS) | this.relayTarget = REL.NS (Forward relay token to REL.NS) | RLY |
| MON-R8 | Rcv: REL | REL.DA == this.address AND !ring.isMember(REL.NS) | this.snCounter = REL.SN this.gsnCounter = REL.GSN Take over right to transmit | HVT |
| MON-T1 | Exp: TPST | txTokenCounter < MAX_TOKEN_PASS | this.txTokenCounter += 1 Resend RTT: RTT.DA = this.successor Restart TPST timer | MON |
| MON-T2 | Exp: TPST | <u>ring.getSuccessor == own.address</u> | | SFR |
| MON-T3 | Exp: TPST | <u>// attempts to pass the token to the next-hop node have failed, and there are still other nodes to try</u> | ring.drop(ring.getSuccessor().nextHop()); ring.TOLOA(); // re-optimize resulting ring, which // computes new next-hop node if (ring.getSuccessor().nextHop() == ring.getSuccessor()) { //Send RTT:token RTT.DA = ring.getSuccessor() } else { //Send REL token REL.NS = ring.getSuccessor(); REL.DA = ring.getSuccessor().nextHop(); } Restart TPST-timer. this.txTokenCounter = 1 Send right-to-transmit token, // see L.4.9 for further details. | MON |

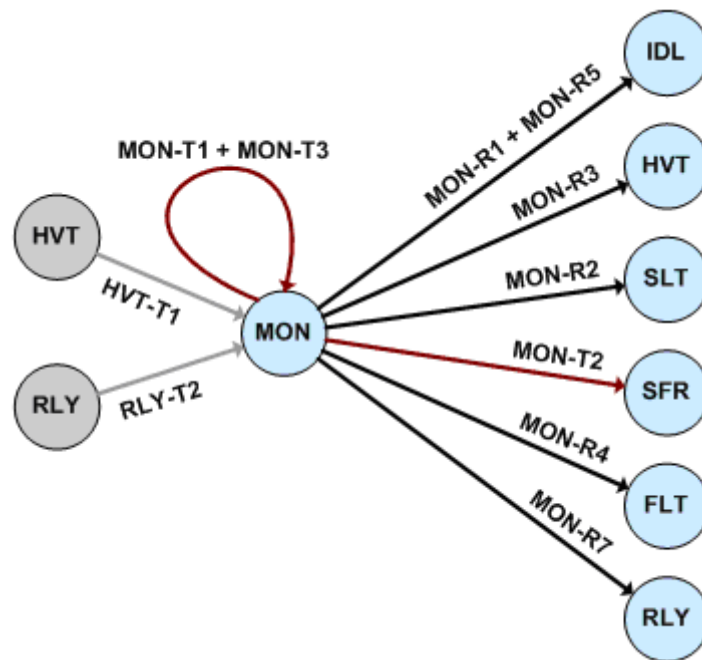


Figure L-17 - MON Outbound transitions from monitoring state

L.4.11.9 Idle State (IDL)

In the *Idle State* the station waits for the *right-to-transmit token* until its TIDL timer expires. When it receives an RTT token, it transits directly to the HVT state, where it may transit to the SLT state and invite new members before returning to the HVT state (as in Figure L-19).

If the TIDL timer expires the station assumes an error-condition in the protocol and transits to the FLT state to recover.

L.4.11.9.1 Idle State entry Actions

The station **shall** start its TIDL timer on state entry.

L.4.11.9.2 Idle State (IDL) outbound transitions

Outbound transitions from the IDL state are shown in the table below. They are in two broad categories, marking either the station's continued operation in a ring with transitions to the HVT or RLY states under correct operation of the protocol, or the detection of an error-condition and exit to the FLT state as recovery action.

Correct continued operation is marked by events in which the station receives the right-to-transmit, either as an RTT token for the ring in which it belongs or as a REL-token forwarding the right-to-transmit to a designated successor in the TOL.

Error conditions for which recovery is required occur when the station hears an RTT token for an unknown ring (transition IDL-R1), or fails to receive the RTT after waiting a prescribed time (transition IDL-T1). The first error condition implies that a new ring has been detected (or formed) and that recovery and ring reformation is required to merge rings. The second error

condition occurs because the ring or some station(s) in it have failed and the token has been lost. In either case, the recovery action is to transit to the FLT state as the starting point for error-recovery.

Table L-25 - IDL State Outbound-Transition Table

| transition | event | condition | action | next state |
|------------|-------------------|---|--|------------|
| IDL-R3 | Rcv: RTT | RTT.DA == this.address | this.predecessor = RTT.TOL(this.address).getPredecessor(); Further action are described in: L.4.8, "Receiving the right-to-transmit". | HVT |
| IDL-A1 | Rcv: RTT | RTT.DA != this.address | detTab.setHeard(RTT.SA) | N/A |
| IDL-R1 | Rcv: RTT | RTT.DA != this.address RTT.RO != ring.getOwner() | // drop to FLT state and wait for opportunity to join the new ring . | FLT |
| IDL-R4 | Rcv: REL token | REL.DA == this.address AND REL.NS == this.address | (Received the right-to-transmit) | HVT |
| IDL-R5 | Rcv: REL | REL.DA == this.address AND ring.isMember(REL.NS) | this.relayTarget = REL.NS (Forward relay token to REL.NS) | RLY |
| IDL-R6 | Rcv: REL | REL.DA == this.address AND !ring.isMember(REL.NS) | this.snCounter = REL.SN this.gsnCounter = REL.GSN Take over right to transmit | HVT |
| IDL-T1 | Exp: TIDL | | | FLT |

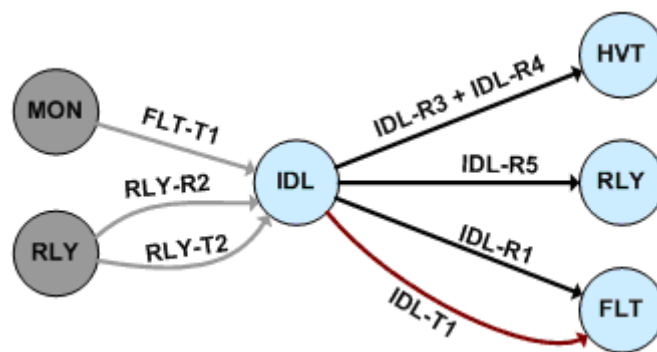


Figure L-18 - Idle State outbound transitions

L.4.11.10 Soliciting State (SLT)

The purpose of the SLT state is to allow new stations to join the ring. A station in the *Soliciting State* is a ring member and has received the *right-to-transmit*. A station in this state **shall** invite new stations to join by broadcasting an SLS token containing the TOL and DM as it knows them. After this broadcast the station **shall** listen for *solicitors* (i.e., nodes attempting to join the network) until the TSLW timer times out.

While in the SLT state, the station **shall** record the received replies (i.e., source-address, TOL, and DM of the SET tokens) of all solicitors.

After the TSLW times out, the station **shall** select a random solicitor from the solicitor list (if any) and accept the updated TOL and DM from the selected solicitor, adding it to its ring database. Then the station **shall** transit to the HVT token state.

L.4.11.10.1 Soliciting State (SLT) entry actions

On state entry the station **shall**:

- Explicitly acknowledge the right-to-transmit by broadcasting an ACK-token, to reduce the chance of collision between repetitions of the RTT token by the station passing it and any solicitations by joining stations.
- invite unaffiliated stations to join the ring by broadcasting a solicit-successor token (SLS), where

SLS.DA=15.255.255.255 ,

SLS.NS= *ring.getSuccessor()*,

SLS.TOL=*ring.ADM().TOL()*, and

SLS.DM = *ring.ADM().DM()*.

- Start the TSLW timer and listen for solicitations to join (i.e., SET tokens).

L.4.11.10.2 Soliciting State (SLT) outbound transitions

Outbound transitions from the SLT state are shown in the table below.

The normal transition (SLT-T1) to the HVT state occurs when the TSLW time expires, and is made whether or not new stations have solicited to join. Any soliciting stations (SLT-A1) are added to the solicitors list from which a random selection will be made at the transition.

The error transition (SLT-R1) to the FLT state occurs if another ring is detected.

Table L-26 - SLT-State Outbound-Transition Table

| transition | event | condition | action | next state |
|------------|-----------|--------------------------|---|------------|
| SLT-A1 | Rcv: SET | SET.DA == this.address | Add SET.SA, SET.TOL and SET.DM to <i>solicitors list</i> . | N/A |
| SLT-R1 | RTT | isFromDifferentRing(RTT) | | FLT |
| SLT-T1 | Exp: TSLW | | If list of solicitors contains at least one solicitor: ring.setPredecessor(SLS.SA) Send SET token where: SET.SA = this.address SET.DA = this.predecessor = SLS.SA SET.RO = ring.getOwner() SET.NS = this.successor = SLS.NS | HVT |



Figure L-19 - Outbound transitions from soliciting state

L.4.11.11 Relaying State (RLY)

In the *Relaying State* a station that received a relay (REL) token takes responsibility to forward it to the *relay target*. A relay token holds the *right-to-transmit*, just like a RTT token, but is passed thru a station in the RLY state because the sender (the station addressed by the REL.SA) has assumed that it can not reach the target *relay target* directly.

The station will wait in RLY state until:

- a *relay token* (REL) it sent previously has been acknowledged,
- another relay request is done,
- it receives the right-to-transmit,
- it detects another ring, or
- the TPST timer expires and the relay token has already been sent for MAX_TOKEN_PASS times.

L.4.11.11.1 Relay State entry actions

On state entry the station **shall**:

- Explicitly acknowledge the received REL-token.

- Send a *relay token* (REL) to relay target address *relayTarget*. (REL.DA = *relayTarget*, REL.NS = *relayTarget*).

After passing the relay token, the transmit token counter **shall** be set to 1 (*this.txTokenCounter* = 1) and the TPST timer shall be started.

L.4.11.11.2 Relaying State (RLY) outbound transitions

Outbound transitions from the RLY state are shown in the table below.

Table L-27 - Relay State (RLY) Outbound-Transition Table

| transition | event | condition | action | next state |
|------------|-----------|--|--|------------|
| RLY-A1 | Rcv: REL | REL.DA == this.address this.relayAddress = REL.NS | Restart TPST -timer. | N/A |
| RLY-R3 | Rcv: RTT | !ring.isMember(RLY.RO) (see para. L.8.3.4, <i>Colliding Rings</i>) | | FLT |
| RLY-R1 | Rcv: RTT | RTT.DA == this.address | setPredecessor(RTT.SA) Further actions taken as prescribed in: L.4.8, "Receiving the right-to-transmit". | HVT |
| RLY-R5 | Rcv: REL | REL.DA == this.address REL.NS == this.address | | HVT |
| RLY-R2 | Rcv: ACK | ACK.DA == this.address AND ACK.SA == this.relayAddress | | IDL |
| RLY-R4 | Any D_PDU | D_PDU.SA == this.relayTarget | | IDL |
| RLY-T1 | Exp: TPST | txTokenCounter < MAX_TOKEN_PASS | Resend REL token. this.txTokenCounter += 1 Start TPST-timer. | N/A |
| RLY-T2 | Exp: TPST | ring.getSuccessor(this.relayTarget) == this.address | | HVT |
| RLY-T3 | Exp: TPST | | | MON |

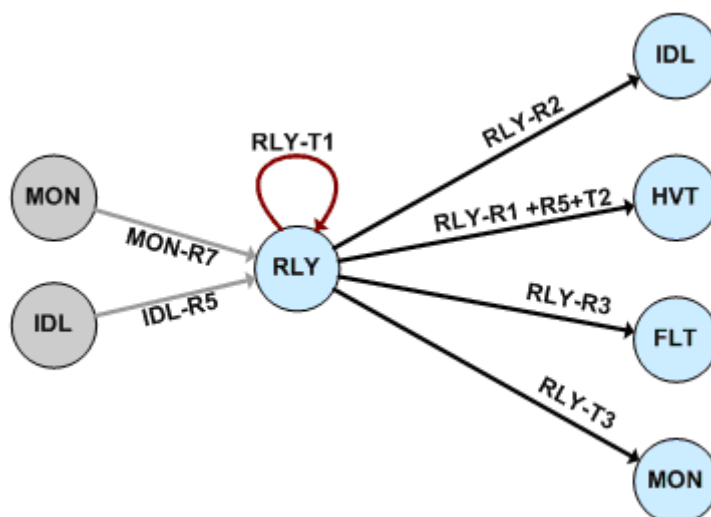


Figure L-20 - Outbound transitions from RLY state

L.5 WTRP STACK OPERATION PARAMETERS

WTRP operation shall depend on a number of parameters whose default values shall be those defined in the Table below. These parameters shall be modifiable by an operator to change the protocol responsiveness or behaviour in response to different operational requirements or tradeoffs (e.g., increased collision probability for newly joining nodes versus reduced solicitation overhead).

Table L-28 - Default values for WTRP Scalar Tuning Parameters

| Parameter Name | Default Value | Units | Comments |
|-------------------|------------------------|-------------|---|
| MAX_TX_TIME | 25.0 | seconds | Maximum Transmit Time |
| MAX_NET_SIZE | 8 | integer | the maximum number of nodes allowed in the ring. |
| SLS_INTERVAL | 8 | integer | Controls the average interval a solicit opportunity should be created by a <i>ring member</i> . (See also L.4.8). |
| MAX_TOKEN_PASS | 3 | integer | Specifies the number retransmissions of a <i>right-to-transmit</i> after which a station shall stop trying. |
| NUM_SLS_SLOTS | 4 | integer | Number of solicit opportunities; a parameter to calculate the TSLW timeout. |
| NUM_SEK_SLOTS | 3 | integer | Number of asynchronous seeking slots; a parameter to calculate the TSLS timeout. |
| <i>modemSpeed</i> | <i>≈2400 ... 64000</i> | bits/second | modem speed. |
| <i>txDelta</i> | <i>≈0.1 ... 3.0</i> | seconds | Delay between the start of data transmission by the WTRP processor, and the start of data reception by the processor. |

L.6 TIME-RELATED CALCULATION

This paragraph describes how the timeouts for the timers (see L.2.1.6) are calculated.

L.6.1 Timer Primitives

The following primitives should be set to calculate the timer timeouts:

| Timer Primitive | Unit | Definition |
|----------------------|------------------|--|
| <i>modemSpeed</i> | Bits per seconds | Transmission speed. |
| <i>txDelta</i> | Seconds | Delay between transmission and reception. |
| <i>MAX_TX_TIME</i> | Seconds | Maximum transmission time if a station has right-to-transmit |
| <i>NUM_SLS_SLOTS</i> | Number of slots | Number of solicit slots. |
| <i>NUM_SEK_SLOTS</i> | Number of slots | Number of asynchronous seeking slots |
| <i>TOKEN_SIZE</i> | Number of bytes | Maximum length of a token in bytes |

L.6.2 Time Slots

There are two types of timeslots: synchronous and asynchronous. A synchronous timeslot is used in scenarios requiring reply with a token to a broadcast. Because all of the stations planning to reply to that broadcast have heard the broadcast the timings will be synchronized. The *replySlotTime* shall be used for these scenarios. The *asyncSlotTime* shall be used as a collision-avoidance mechanism for the first traffic sent in scenarios where no other stations have been heard; these scenarios, no natural synchronization mechanism exists and replies will be asynchronous.

$$\triangleright \text{replySlotTime} = \text{txDelta} + \frac{8 \bullet \text{TOKEN_SIZE}}{\text{modemSpeed}}$$

$$\triangleright \text{asyncSlotTime} = 2 \bullet \text{txDelta}$$

L.6.3 Timers with static timeouts

The TSLW, TCLT, TPST and TIDL have a static timeout that can be calculated using the following formulas:

$$\triangleright \text{timeout}_{\text{TSLW}} = (\text{NUM_SLS_SLOTS} + 1) \bullet \text{replySlotTime}$$

$$\triangleright \text{timeout}_{\text{TIDL}} = \max(2 \bullet \text{timeout}_{\text{TSLW}}, \text{MAX_TX_TIME} + 2 \bullet \text{txDelta})$$

$$\triangleright \text{timeout}_{\text{TCLT}} = \text{timeout}_{\text{TIDL}} - \text{asyncSlotTime}$$

L.6.4 Timers with dynamic timeouts

Dynamic timeouts randomize station's transmissions to reduce the probability of collisions in their replies to a shared event.

➤ $timeout_{TSLs} = replySlotTime + (n+1) \bullet asyncSlotTime$

Where n is the randomly chosen slot number: $0 \leq n < NUM_SEK_SLOTS$.

➤ $timeout_{TSRP} = n \bullet replySlotTime$

Where n is the random chosen slot number: $0 \leq n < NUM_SLS_SLOTS$.

➤ $timeout_{TCON} = (NUM_SLS_SLOTS + 1) \bullet replySlotTime - timeout_{TSRP} + D \bullet MAX_TX_TIME$

Where $timeout_{TSRP}$ is the last calculated timeout of the TSRP-timer, and D is the number of stations in the TOL between the inviting station and the soliciting station.

➤ $txDelta < timeout_{TEOT} \leq MAX_TX_TIME$

➤ $timeout_{TPST} = n \bullet replySlotTime$

Where n is equal to 4 if the successor is reached thru relaying; otherwise 2.

L.7 TOKEN-RING MANAGEMENT

Management of the Token-Ring is described in the sections that follow.

L.7.1 Ring and Station Parameters

Each station shall keep a connectivity table, which contains a transmit order list (of the ring in which it is a member) and a global list. The transmit order list describes the order in which ring members receive the RTT token, and the global list contains all stations that this station has heard in the network.

The other parameters a station shall keep include: the ring owner of the ring to which it belongs, the current sequence number, the current generation sequence number, its successor and its predecessor.

A node can be a member of only one ring at a time.

L.7.2 Dealing with the Hidden Node Problem (3-node linear ring)

The Hidden Node problem occurs in a wireless network where two nodes are sending to a common destination, but are each unaware that the other exists, thus resulting in collision. In the figure below, where S_A , S_B , and S_C are members of a same ring, station S_A and station S_B are in communication range of each other, and station S_B and station S_C are likewise in communication range of each other. Stations S_A and S_C are not within communication range of each other. Both S_A and S_C might transmit without interfering with each other, but their transmissions will interfere (are said to collide) if they transmit to S_B at the same time. Stations S_A and S_C are said to be hidden from each other.

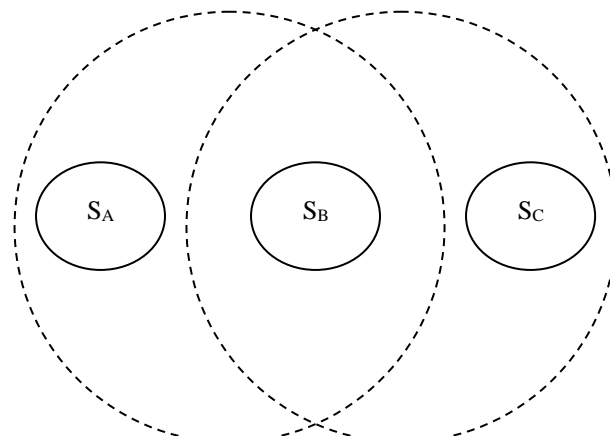
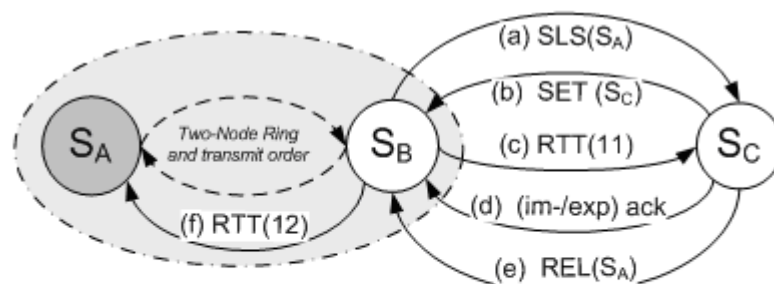


Figure L-21 - Hidden-Terminal Scenario

WTRP does not require that every member in a token ring hear all other members, only that every member is within communication range of its predecessor and successor.

In WTRP, a node can only join a ring if it has received a SLS token from its future predecessor and it can reach its future successor. An exception is made for the three-node chain topology, as shown in the Optimistic-Joining Scenario (Figure L-22) below.



Scenario:

- Station S_A and station S_B are in a two-member ring. Station S_A is the ring owner.
- Station S_B solicits for a new successor by broadcasting a SLS token, and S_C responds with a SET token.
- S_B selects S_C to be its new successor, passes it the right_to_transmit token RTT and waits the associated ACK.
- S_C uses REL token to request S_B to forward the right-to-transmit to S_A .

Figure L-22 – Optimistic Joining Scenario (two-nodes to three-nodes) that results in Token-Relay Operation in a Three-Node Linear Network

If a station (i.e., station S_C) receives an SLS token but can not hear its future successor (i.e., Station S_A), then it examines the ring size given in the SLS token: SLS{NON}. If there are exactly two members in the ring, then the SLS-token recipient shall attempt to join the ring by responding with a SET token, and then enter into a special mode called the token relaying mode.

This mode exists to allow the ring to grow in this special topology. The restriction to two nodes growing to three is intended to reduce the number of token relays that exist in the ring to promote efficiency.

Upon receipt of an RTT token, the node in *relaying* mode converts the RTT token into a REL token and passes it to a chosen relayor, its predecessor, instead of its successor⁷. The chosen relayor receives the REL token, updates it and passes the REL token to the relay target. The figure below shows the three-node chain topology formed by S_A , S_B , and S_C that results from the optimistic joining scenario of Figure L-22. Station S_C can not reach its successor S_A , so it relays the REL token through S_B , which is in the same communication range as both S_A and S_C . S_B then converts the REL token to a RTT token and pass this RTT token to S_A .

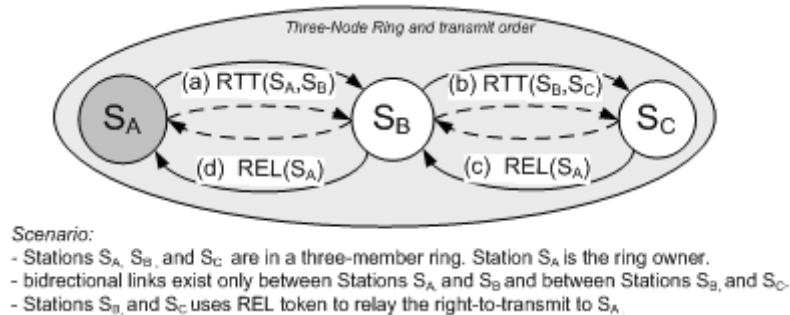


Figure L-23 - Token-Relay Operation in a Three-Node Linear Network

Under this specification of the token-ring protocol, further use of the token-relay mode is prohibited, to reduce inefficiency in the ring operation. Nodes may employ optimistic joining only in the ring-growth scenario from two nodes to three nodes, and may use token relay only in this special case of the three-node linear network.

L.8 FAILURE RECOVERY ANALYSIS

Three basic operations for a HF token ring are examined: initial ring formation, ring growth, and ring maintenance. We first describe all events leading up to the successful completion of each operation; for each operation, we also examine scenarios where one or more of these events fail. We will depict events occurring in each phase using a *state-time* diagram. A state-time diagram is a diagram where the horizontal lines represent state execution by stations, with time progressing from left to right. An arrow from one process (i.e., station or node) to another represents a message (nominally a token) being sent, with the send event at the base of the arrow. Internal events within the station have no arrows associated with them.

L.8.1 Failure Analysis of Initial Ring Formation/Creation

Let us consider the scenario where there are no failures in message reception, and S_A and S_B successfully form a two-member ring. Initially, there are only two self rings: ring S_A and ring S_B , formed by the two stations S_A and S_B respectively.

In the two figures below (Figure L-24 and Figure L-25), station S_B times-out first from the SFR state and seeks a successor by sending an SLS token ('step (a)'). Then station S_A responds by sending a SET token to S_B ('step (b)'), which then sends the RTT(0) token to S_A ('step (c)'), accepting station S_A as a new ring member and granting it the right to transmit. Station S_A

acknowledges the RTT (an explicit ACK token is shown S_B ('step (d)'), though any DPDU transmission may serve as an implicit acknowledgement of receipt of the RTT). Given this as the basic scenario and the desired message-sequence exchange diagram, additional scenarios are examined below where there are failures at one point or another in the message exchange.

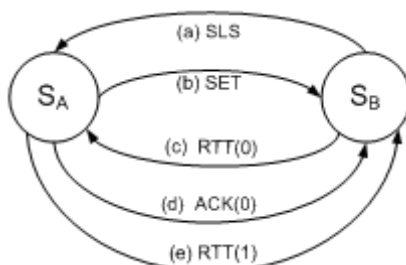


Figure L-24 - Graphical representation of two stations, S_A and S_B , forming a two-member ring.

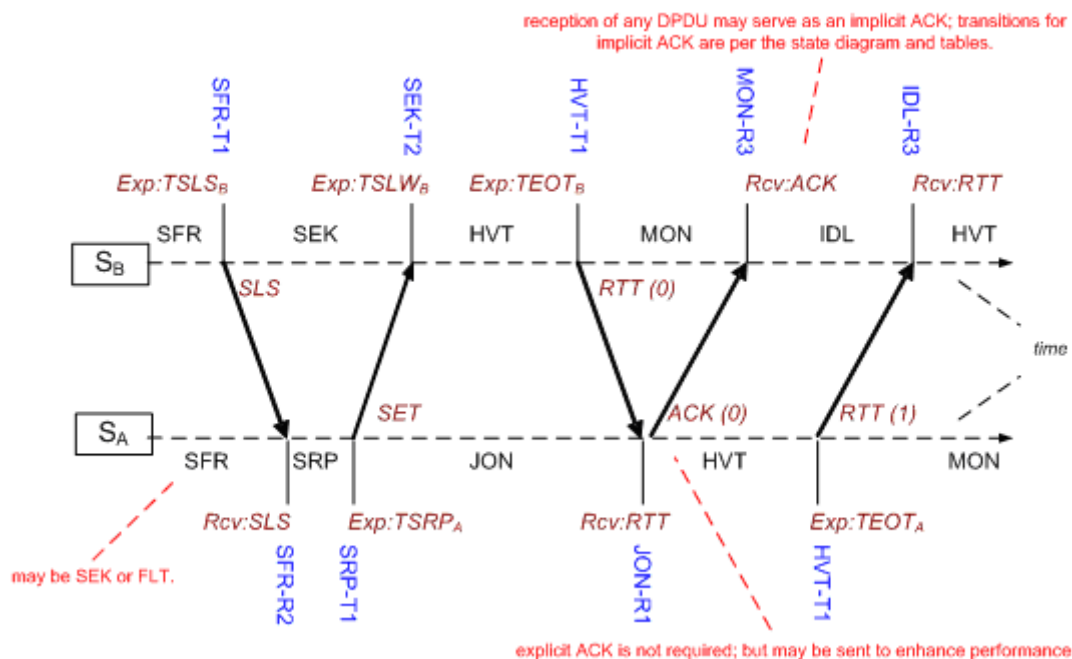


Figure L-25 - State and Time diagram that shows station S_A and station S_B forming a two-node ring

L.8.1.1 Lost SLS Token

If the SLS token from S_B is lost (i.e., not received), then station S_A will stay in the SFR state, and S_B will eventually time out from its TSLW timer and go back to SFR state. The sequence of events just described is shown in Figure L-26.

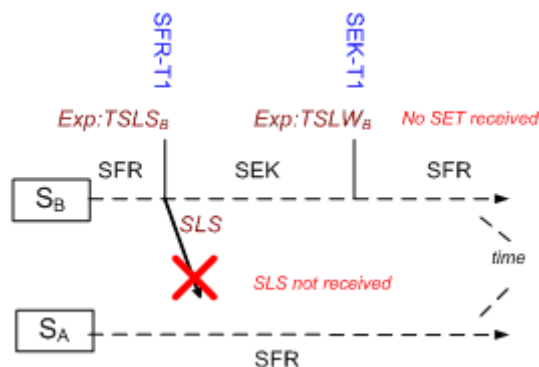


Figure L-26 - State and Time diagram of the scenario where SLS token from S_B is lost.

In this case, the failure recovery action takes place entirely at station S_B , as S_A will be unaware that an SLS has been sent and remains in the SFR state throughout the scenario. After station S_B sends the SLS token, it waits for SET responses until its TSLW expires and, hearing none, the station returns to the SFR state.

L.8.1.2 Lost SET Token

Consider now the following scenario where the SLS from S_B is successfully transmitted, but the SET token from S_A is lost. In this case, both stations must take failure recovery actions, as each has sent a message and is awaiting a response.

As before, station S_B is waiting for responses to its invitation to join the ring. The TSLW timer at station S_B will expire and, having heard no SET tokens, the station will transition back to the SFR state. At station S_A , where a SET token has been sent and the station is waiting for a reply to confirm that it has entered the ring, the TCON timer expires without station S_A hearing a response and the station returns to the FLT state. The message-sequence diagram for this scenario is shown in the figure.

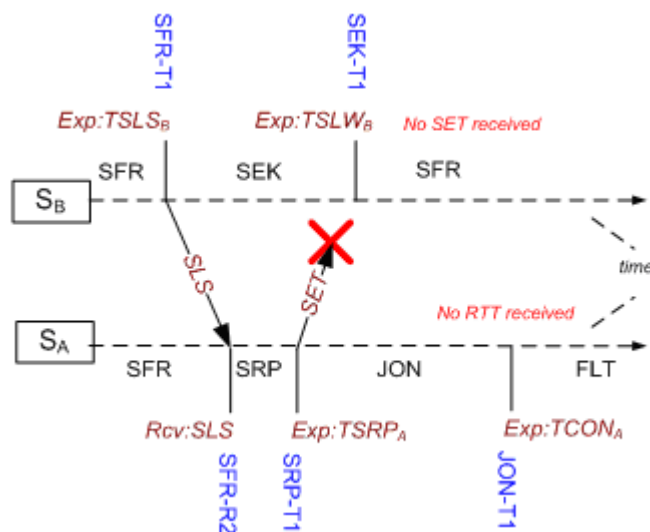


Figure L-27 - State and Time diagram of the scenario where SET token from S_A is lost.

state. Worst-case, the scenario returns to a condition with both nodes in the SFR or FLT states and eventually attempting to initialize the ring once again. Short of this, the two nodes go through some repetitive attempts to form the ring that ultimately succeed. In the best case, there are no lost tokens and the ring forms smoothly as in Figure L-25.

L.8.2 Failure Analysis of Ring Growth

First, let us consider a growth scenario where there are no failures but contention to join the ring, where a two node ring has two nodes waiting to join. At the 5th rotation (for example), the sequence number of station S_A is nine and the sequence number of station S_B is ten. Station S_B invites others to join by broadcasting an $SLS(S_A)$ token (i.e., a solicit token announcing station S_A as the successor). Both S_C and S_D receive this $SLS(S_A)$ token from S_B , and both send a SET token to S_B at separate instants, as determined by the random slot-selection algorithm. As shown in the ring-growth diagram of Figure L-29, S_B selects S_C to be its new successor and sends $RTT(11)$ to S_C . Upon receipt of $RTT(11)$, S_C moves from JON state to HVT state and sends either an explicit acknowledgement to S_B or sends any other traffic that S_B can hear and will treat as an implicit acknowledgement; after completing its data transmission, station S_C sends the RTT token (i.e., $RTT(12)$) to station S_A . The scenario ends with a three-node ring as in Figure L-30, and station S_D awaiting the next SLS token (from any ring node) to solicit to join.

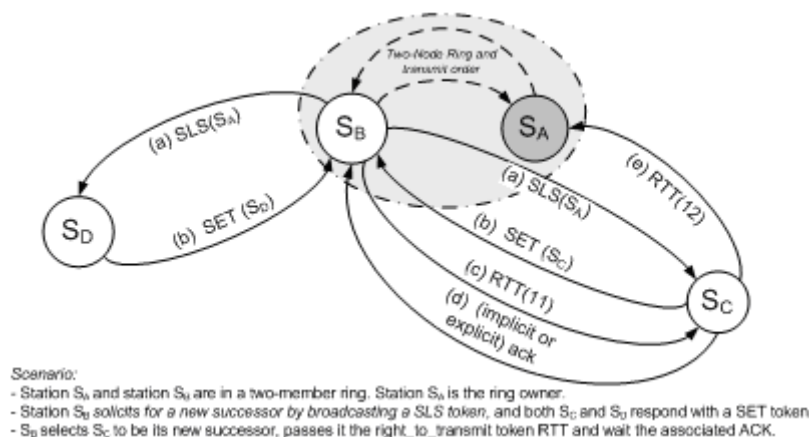


Figure L-29 - Ring-Growth Scenario: Two-Node Ring with growth-potential to four nodes

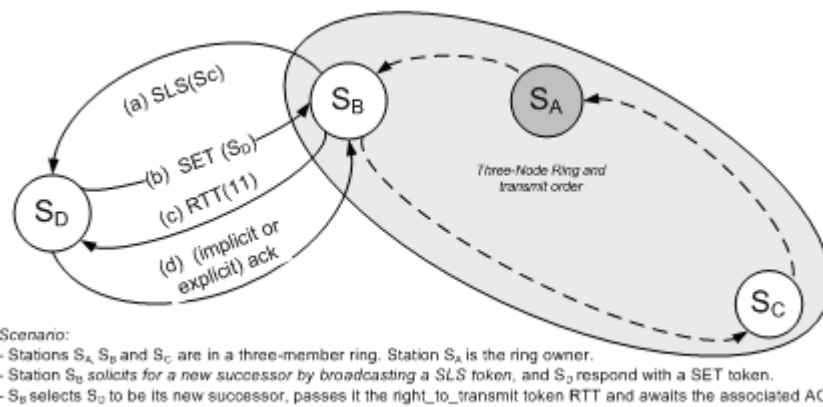


Figure L-30 - Ring-Growth Scenario: the newly formed Three-Node Ring with growth-potential to four nodes

The message-sequence diagram follows directly in this scenario. It is shown in the figure below, and captures the same sequence of events called out in the discussion of the ring-growth diagram of Figure L-29.

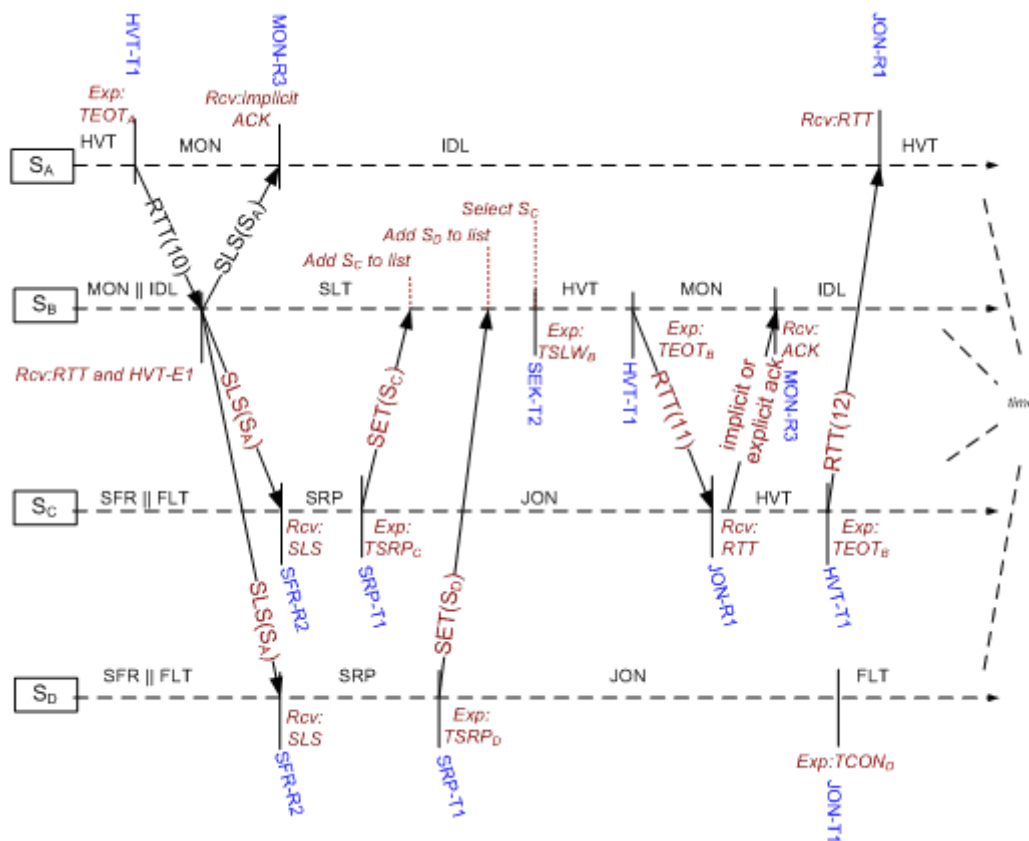


Figure L-31 - Solicitation Scenario with non-colliding responders S_C and S_D .

In the scenario in which $SLS(S_A)$ is successfully delivered, but SET token from either S_C and S_D is lost, then the node from which S_B receives a SET token will join the ring, just as in Figure L-31.

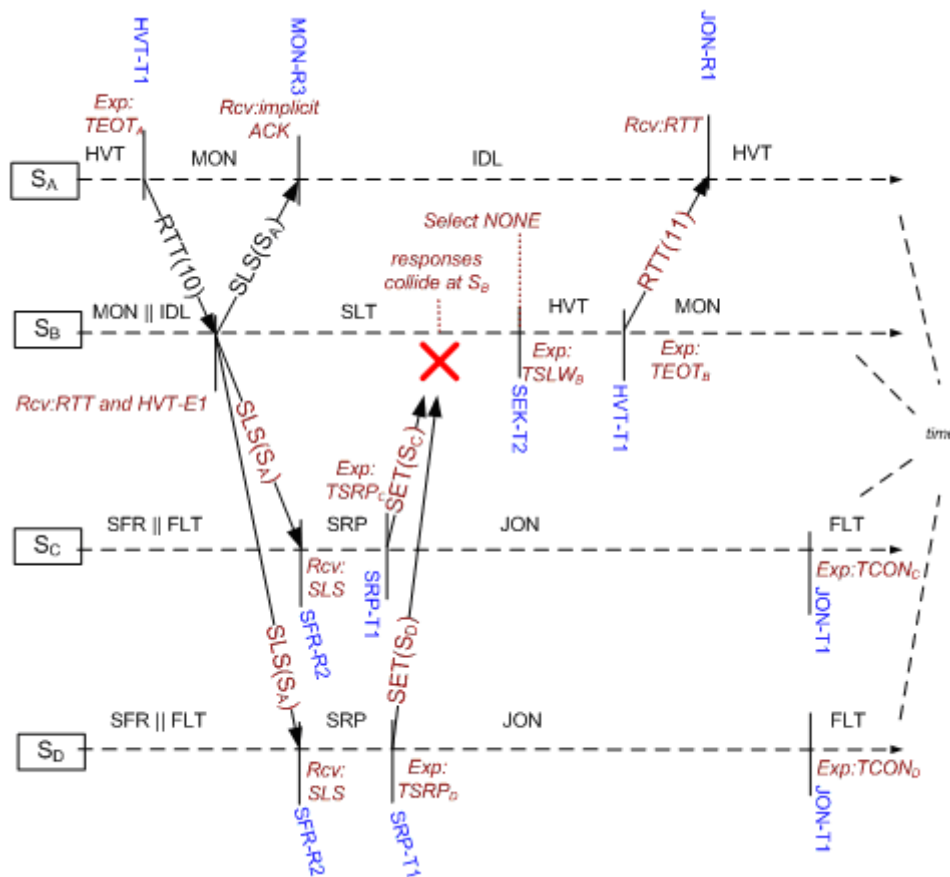


Figure L-33 - Solicitation Scenario with colliding responders S_C and S_D

L.8.3 Failure Analysis of Ring Maintenance

Other failures to maintain data parameters in the ring are analyzed below, with the protocols enacted to recover from them.

L.8.3.1 Multiple RTT Tokens

WTRP's state-machine has multiple error-recovery mechanisms that are designed to reduce contention for control of the channel, control that is maintained properly by having a single right-to-transmit. However, multiple RTT tokens might arise in the ring, as a result of the error recovery mechanisms or during initial ring formation as stations in the SFR state act to seek new ring members. Since a properly maintained ring has only one right-to-transmit token (circulating in the form of an RTT or REL token), the existence of multiple RTT tokens is a failure event that requires correction by the ring members.

Most recovery mechanisms in WTRP to mitigate the effects of multiple RTT tokens are conservative and passive, and entail the station's retreat to the FLT state to avoid channel contention, and later re-formation of, or re-entry into, a new ring. While this strategy is time consuming, it is designed to 'do no harm' through a passive correction of multiple tokens by avoiding further confusion on the channel.

A more active strategy to recover from multiple RTT tokens is defined by Ergen et al in the reference token-ring protocol and included here in the WTRP protocol, though its use is optional and is deliberately limited to a just few cases.

If enabled, active response to the presence of multiple RTT tokens **shall** be triggered when a station receives RTT tokens out of sequence, from past ring-cycles, or from different ring owners than its own (see section L.8.3.4), and **requires** the detecting station to send a DEL token to the originator of any RTT token that should be removed. The detecting station **will** send a DEL token for all RTT tokens except the one with the highest priority, where token priority for RTT tokens A and B is defined on two parameters:

- the token generation-sequence number (GSN), i.e., RTT token A has lower priority than RTT token B if $A.GSN < B.GSN$, and
- the ring-owner address, i.e., RTT token A has lower priority than RTT token B if $A.RA < B.RA$.

The priority rules effectively favor RTT tokens from rings whose owners have larger addresses and from older (and presumably more stable) rings.

Specifically, when active response is enabled, the detecting station **shall** send a DEL token to the source of each RTT it receives whenever:

- $(RTT.GSN < ring.GSN)$ OR
- $(RTT.GSN == ring.GSN)$ AND $(RTT.RA < ring.RA)$

L.8.3.2 Losing the Ring Owner

Loss of the Ring-Owner is detected, as required in section L.4.8, receiving the right-to-transmit, when a station receives an RTT token with GSN lower than or equal to that of the last RTT token it received (i.e., with $RTT.GSN == this.gsnCounter$), an indication that the RTT token has made a full circuit of the ring without the owner incrementing the Generation-Sequence-Number as required.

The node that detects this failure claims the ring ownership, updating the RTT token by inserting its own address as Ring-Owner and updating the Generation-Sequence-Number before passing it to its successor. Upon receipt of the RTT token, a station updates its ring owner and its generation sequence number. After one ring rotation, all members will have updated their ring owner and generation sequence number, and thus recovered from the

Note some characteristics of this algorithm:

- Loss of owner is NOT declared by a node when it receives an RTT with a different owner than that stored in its database, for if it did, the mechanism described here would dissolve into an endless set of updates with each successor node in the TOL taking over ring-ownership from its predecessor. Rather, loss of ring ownership is detected by noting that a ring owner (i.e., any node acting as ring owner) has failed to perform an action that the ring owner is required to do, i.e., update the generation sequence number. Since this algorithm requires the new owner to perform that action, subsequent nodes

detect that the ring is owned by some active node, and they take no further action other than noting the address of the current ring owner and the GSN, actions a node is required to do on every receipt of the RTT, whether or not the ring-owner is lost or not.

- The loss-of-ring-owner detection algorithm does not conflict with the algorithm that detects colliding rings, since the node does not use the ring-owner address (as such) to determine if a token has been generated by a node in another ring (see Section L.8.3.4, below).

L.8.3.3 Unreachable Successors

A functional ring requires that a station's successor be connected in both directions by a working link, i.e., that the successor be reachable. An unreachable successor is a ring failure that must be detected and corrected. This failure is detected when a station, after passing the RTT token to its successor, fails to receive any ACK token, either implicitly or explicitly, from its successor. The station that detects this failure first retries passing the RTT token to its successor for MAX_TOKEN_PASS times. If it still fails to get any response from its successor, then the station deletes the unreachable node from its TOL and Distance Matrix, sets the successor of its successor to be its new successor and passes the RTT token and payload with these updated parameters to its new successor (computing and using an intermediate token-relay node if necessary). This action effectively drops the unreachable successor from the ring. The unreachable successor will recover when its TIDL timer expires and it awaits an SLS token and the opportunity to rejoin the ring. Inefficiencies in the TOL that may arise from this action are resolved through Transmit-Order-List Optimization in later circulations of the RTT when the network topology has stabilized, in accordance with the requirements of Section L.4.5.

L.8.3.4 Colliding Rings

It's possible for two different rings to collide. It happens when one member of a ring enters the communication range of another ring, effectively, when it receives an RTT token from a node that is not in its TOL. When a node in a ring, including a self ring, receives tokens from a different ring which has higher *ring priority* than its own ring, then it leaves its own ring and goes to FLT state. Hence, a ring of lower priority is able to trim away members which may collide with members from another ring of higher priority; the ring with higher priority may take in nodes which have broken away from the ring with lower priority.

The *ring priority* of a self ring is the lowest among all rings; the *ring priority* of a ring with any relaying also is considered the lowest. Thus,

- a node in SFR state in which a node belongs to a self ring automatically leaves its ring and goes to FLT state upon hearing from any different ring;
- a node in RLY state in which a node is involved in relaying a RTT token automatically leaves its ring and goes to FLT state upon hearing from any different ring.

- However, a node in SLT, IDL, and MON states in which a node belongs to a non-self-ring only leaves its ring and goes to FLT state if its ring priority is lower than that of the different ring it hears.

Additional ring-priority rules:

- Ring R1 has higher priority than ring R2, if R1 is older than R2, or the generation sequence number of R1 is higher than that of R2;
- if R1 and R2 happen to have the same generation sequence number, then the ring whose ring-owner address is larger has higher ring priority.

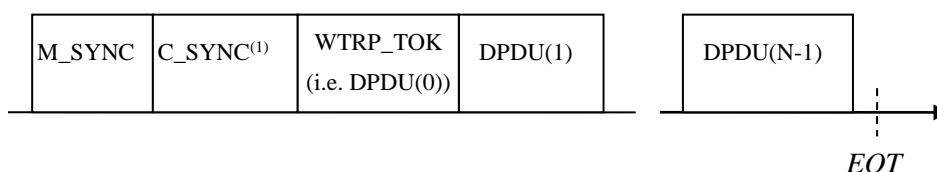
If token-deletion is enabled (see section L.8.3.1), stations in the ring of higher priority may send DEL tokens to stations in the ring of lower priority.

L.9 TOKEN TRANSMISSION

While in theory and in practice it may be possible to transmit WTRP management tokens at any time during a station's (i.e., RTT token-holder's) transmission, practical concerns dictate standardization of the token location within a transmission interval. In particular, timer-related events governing the WTRP operation are often referenced with respect to the transmission or reception of a token - and as the natural duration of the transmission interval varies as a function of the station's traffic load it is desirable to reduce the dependency of expected response time to a minimum.

Depending on token-type, tokens can be transmitted at the beginning or at the end of the transmission intervals as shown in Figure L-34 and Figure L-35; additionally, state machine operation requires that some tokens be the sole message in a transmission.

In head-end token placement, the WTRP token(s) shall be transmitted as the first DPDU(s) in the transmission interval following the modem synchronization (M_SYNC) and crypto-synchronization (C_SYNC, if any) preambles, and preceding any other DPDUs sent by the station during its transmission interval.



(1) – crypto-synchronization preamble will not be present in unsecured systems.

Figure L-34 - Head-End Token Placement

In tail-end token placement, the WTRP tokens **shall** be transmitted as the last DPDUs in the transmission interval following any other DPDUs transmitted by the station.

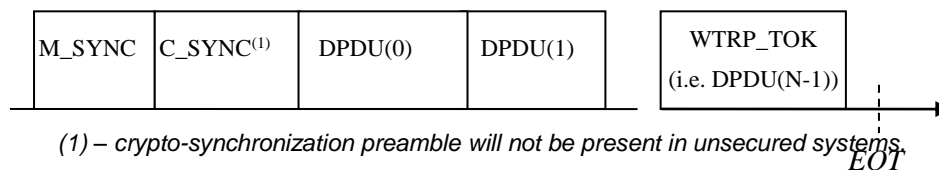


Figure L-35 - Tail-End Token Placement

Some protocol actions require that the token be the sole DPDU in the transmission interval.

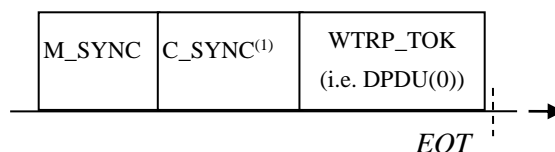


Figure L-36 – Sole-Token Placement

Token placement requirements within the transmit interval **shall** be a function of token-type, as specified in Table 23 below.

Table L-29 - Required token placement in a transmission interval.

| Token Type | Placement | Comment |
|------------|-----------|---|
| RTT | TAIL | reduces the dependency of the response time on any traffic offered by the RTT token sender. |
| ACK | HEAD | reduces the dependency of the response time on any traffic offered by the acknowledging node. |
| SLS | SOLE | State-machine requires SLS as the only token in the transmit interval |
| SET | SOLE | State-machine requires SET as the only token in the transmit interval |
| REL | SOLE | Fastest passes of the right-to-transmit to the successor. |
| DEL | HEAD | if used, deletes invalid tokens at earliest opportunity |

The end-of-transmission (EOT) field values for tokens - just as for any DPDU in the transmission interval - shall be computed in accordance with S'5066 Annex C Section C.3.2.3.

L.10 WTRP SUPPORT TO THE INTERNET PROTOCOL

WRTP, as a MAC-layer protocol, may be used to support any of the client-services defined in STANAG 5066 Annex F. It has some specific features intended to support the IP-client and Ether-client types of Annex F Sections F.12 and F.11 (respectively).

L.10.1 IP-/MAC-Address-Resolution Support

The token-payload's transmit-order-list provides a list of ordered pairs of MAC-layer (i.e., STANAG 5066) and Network-Layer (i.e., Internet Protocol) addresses. The TOL serves as an address-resolution table between the STANAG 5066 and IP addressing spaces, and obviates the need for an explicit Address-Resolution Protocol (ARP) when using WTRP.

STANAG 5066 implementations of WTRP **shall** enable client-management functions (e.g., using the S_MANAGEMENT_REQUEST and S_MANAGEMENT_INDICATION primitives of Annex A) to extract the TOL as an address resolution table for use by IP-client and Ether-client services.

L.10.2 Dynamic IP address Assignment Service

The IP addresses in the TOL imply an IP subnet range within which they are defined. Stations **should** exploit this information to implement a dynamic IP-addressing service in WTRP.

Consequently, a station joining the WTRP network **shall** examine the TOL, determine the IP subnet in use, pick an unused address in the subnet range, pair it with its STANAG 5066 address and thereby reserve it for its use when it inserts itself in the TOL and returns the TOL that is embedded in the SET token when it solicits to join.

Competing IP address choices by stations joining at the same invitation opportunity **shall** be resolved by the inviting station.

As the subnet range is ambiguous (the TOL has no associated IP-subnet mask to fully specify the range), a 24-bit mask (i.e., a 255.255.255.0 mask) **shall** be assumed unless overridden by network management information.

IP-addresses obtained in this manner **shall** be assigned automatically to the IP-client or Ether-client attached to the WTRP subnet.

Index of Figures

| | |
|---|----|
| Figure L-1 - Normal Token-Ring Operation | 8 |
| Figure L-2 - Ring Formation | 10 |
| Figure L-3 - Ring Entry | 11 |
| Figure L-4 – DM-elements used to compute the Ring-Cycle Length (general case) | 15 |
| Figure L-5 – DM-elements used to compute the Ring-Cycle Length (example, 4-node network) | 15 |
| Figure L-6 – Augmented Distance-Matrix Structure | 34 |
| Figure L-7 – Augmented Distance Matrix, Detail | 35 |
| Figure L-8 – ADM Row-Swap and Column-Swap Operations: Example | 36 |
| Figure L-9 – Insertion of node j into the TOL | 39 |
| Figure L-10 - Wireless Token Ring Protocol State Diagram | 42 |
| Figure L-11 - FLT Outbound transitions | 44 |
| Figure L-12 - Outbound transitions from SFR state | 46 |
| Figure L-13 - Outbound transitions from SEK state | 47 |
| Figure L-14 - Outbound transitions from SRP state | 49 |
| Figure L-15 - Outbound transitions from JON state | 50 |
| Figure L-16 - Outbound transitions from HVT state | 52 |
| Figure L-17 - MON Outbound transitions from monitoring state | 55 |
| Figure L-18 - Idle State outbound transitions | 56 |
| Figure L-19 - Outbound transitions from soliciting state | 58 |
| Figure L-20 - Outbound transitions from RLY state | 60 |
| Figure L-21 - Hidden-Terminal Scenario | 63 |
| Figure L-22 – Optimistic Joining Scenario (two-nodes to three-nodes) that results in Token-Relay Operation in a Three-Node Linear Network | 63 |
| Figure L-23 - Token-Relay Operation in a Three-Node Linear Network | 64 |
| Figure L-24 - Graphical representation of two stations, S_A and S_B , forming a two-member ring. | 65 |
| Figure L-25 - State and Time diagram that shows station S_A and station S_B forming a two-node ring | 65 |
| Figure L-26 - State and Time diagram of the scenario where SLS token from S_B is lost. | 66 |
| Figure L-27 - State and Time diagram of the scenario where SET token from S_A is lost. | 66 |
| Figure L-28 - State and Time diagram of the scenario where right_to_transmit token from S_B is lost. | 67 |
| Figure L-29 - Ring-Growth Scenario: Two-Node Ring with growth-potential to four nodes | 68 |
| Figure L-30 - Ring-Growth Scenario: the newly formed Three-Node Ring with growth-potential to four nodes | 69 |
| Figure L-31 - Solicitation Scenario with non-colliding responders S_C and S_D . | 69 |
| Figure L-32 - Solicitation Scenario with Lost SLS Token | 70 |
| Figure L-33 - Solicitation Scenario with colliding responders S_C and S_D | 71 |
| Figure L-34 - Head-End Token Placement | 74 |
| Figure L-35 - Tail-End Token Placement | 75 |
| Figure L-36 – Sole-Token Placement | 75 |

Index of Tables

| | |
|--|----|
| Table L-1 - RTT Token Field Values | 5 |
| Table L-2 - Token Field descriptions for Ergen et al's <i>original</i> WTRP specification (historically informative only). | 15 |
| Table L-3 – EOW-Token Message (Extended Form) – Type 6 Management DPDU w/ ID-Management EOW | 16 |
| Table L-4 - EOW-WTRP-Token Message (Extended Form) - Type 6.15 Management DPDU w/ ID-Management EOW | 18 |
| Table L-5 – EOW-Token Message (Extended Form) – Subtype-15/WTRP Token-Management DPDU Payload format. | 20 |
| Table L-6 – Structure of the Transmit-Order List | 21 |
| Table L-7 – Distance-Matrix Encoding (when the number of network nodes is even) | 22 |
| Table L-8 – Distance-Matrix Encoding (when the number of network nodes is odd) | 23 |
| Table L-9 – Token Payload Size in bytes as a function of network size | 24 |
| Table L-10 - Frame-Control-Field Values | 25 |
| Table L-11 - RTT Token Field Values | 26 |
| Table L-12 - ACK Token Field Values | 26 |
| Table L-13 - SLS Token Field Values | 27 |
| Table L-14 - SET Token Field Values | 28 |
| Table L-15 - REL Token Field Values | 29 |
| Table L-16 - DEL Token Field Values | 30 |
| Table L-17 - Station registers and flags | 30 |
| Table L-18 - FLT outbound transitions | 44 |
| Table L-19 - SFR-State Outbound-Transition Table | 45 |
| Table L-20 - SEK state outbound transitions | 47 |
| Table L-21 - SRP outbound transition table | 49 |
| Table L-22 - JON-State Outbound-Transition Table | 50 |
| Table L-23 - HVT-State outbound transition table | 51 |
| Table L-24 - MON-State Outbound-Transition Table | 54 |
| Table L-25 - IDL State Outbound-Transition Table | 56 |
| Table L-26 - SLT-State Outbound-Transition Table | 58 |
| Table L-27 - Relay State (RLY) Outbound-Transition Table | 59 |
| Table L-28 - Default values for WTRP Scalar Tuning Parameters | 60 |
| Table L-29 - Required token placement in a transmission interval. | 75 |

**ANNEX M – ADAPTIVE TIME-DIVISION MULTIPLE ACCESS
PROTOCOLS USING STANAG 5066 DTS SUBLAYER MESSAGING
(INFORMATIVE)**

**THIS ANNEX RESERVED
FOR FUTURE USE**

LEFT BLANK INTENTIONALLY

ANNEX N – GUIDANCE ON ADDRESS MANAGEMENT IN STANAG 5066 NETWORKS (INFORMATIVE)

N.1 INTRODUCTION

This annex provides guidance for allocation and use of STANAG 5066 addresses in operational systems. It defines a block addressing scheme [¹] that can be used for formally planned and ad-hoc operations with a minimal amount of system reconfiguration required.

N.2 MANAGED ADDRESSES

This annex **does not** alter the requirements for size, format, or representation of node addresses defined elsewhere in this STANAG, notably but not limited to the following:

- Annex A section A.2.2.28.1 – Node ADDRESS Encoding for all [S_]Primitives;
- Annex C section C.3.2.4 – Size of Address [encoding in DPDUs];
- Annex C section C.3.2.6 – Source and Destination Address [encoding in DPDUs];
- Annex F section F.11.5.4 – Derivation of 48-bit pseudo-Ethernet media-access-control (MAC) addresses from STANAG 5066 addresses.

Addresses in this annex are given in the dotted-decimal format defined [²] in Annex C section C.3.2.6, and abbreviated generically in the form w.x.y.z

N.3 NODE ADDRESS-BLOCK ASSIGNMENTS

Only full-length (i.e., 28-bit) S'5066 node addresses **shall** be subject to the assignment recommendations of this annex.

Regional blocks **shall** be defined by the 4 most-significant bits of the full-length address (i.e., the first element w of the dotted-decimal form w.x.y.z).

¹ This Annex bases its address-block allocation strategy on a proposal put forth by the US Navy as part of their concept of operations for the Battle-Force E-Mail 66 system, which is one of the larger deployments of STANAG-5066 based systems managed under a single operational authority.

² As of STANAG 5066 Edition 2

Node addresses less than full-length (i.e., with $w = 0$) **may** be assigned and managed on an ad hoc basis in unique situations; they are otherwise unmanaged by any provision of this Annex. Management authorities for less-than-full-length addresses are not defined.

Managed regional address blocks **shall** be defined by the 4 most-significant bits of the full-length address, in accordance with the Table below.

Table N-1 — Top-Level Address-Block Assignments by Region

| Address Range | Regional Assignee | Management POC | Comment |
|---------------------------|---------------------------------------|----------------|--|
| 0.0.0.0 — 0.255.255.255 | unassigned | ad-hoc | Variable-length addresses are unmanaged |
| 1.0.0.0 — 1.255.255.255 | United States | US DoD | includes US Armed Forces and Homeland Security as major S'5066 users |
| 2.0.0.0 — 2.255.255.255 | North America | TBD | |
| 3.0.0.0 — 3.255.255.255 | North America | TBD | other than US government |
| 4.0.0.0 — 4.255.255.255 | South America | TBD | |
| 5.0.0.0 — 5.255.255.255 | NATO | TBD | |
| 6.0.0.0 — 6.255.255.255 | Europe | TBD | |
| 7.0.0.0 — 7.255.255.255 | Europe | TBD | |
| 8.0.0.0 — 8.255.255.255 | Asia | TBD | |
| 9.0.0.0 — 9.255.255.255 | Asia | TBD | |
| 10.0.0.0 — 10.255.255.255 | Africa | TBD | |
| 11.0.0.0 — 11.255.255.255 | Middle East | TBD | |
| 12.0.0.0 — 12.255.255.255 | Australasia, New Zealand, and Oceania | TBD | |
| 13.0.0.0 — 13.255.255.255 | Non-Governmental Organizations | TBD | includes, e.g., the International Committee of the Red Cross |
| 14.0.0.0 — 14.255.255.255 | other | TBD | |
| 15.0.0.0 — 15.255.255.255 | other | TBD | |

Authority to assign and manage a specific address for a specific system **may** be devolved (perhaps more than once) to a regional or sub-regional administrator. Devolution of management authority within a regional block is beyond the scope of this Annex.

A system's address need not — indeed, should not — be changed when it moves from one geographic region to another. Regional membership and responsibility to assign an address to a system is based on the region in which the system is registered or affiliated (both of these processes are outside of the scope of the STANAG), rather than the system's location. The major exceptions to this provision are systems belonging to nations involved in NATO operations, where the option of using a national or NATO address may be exercised. Even in this case, however, the election to use an address in one block or another is based on administrative or operational criteria, not physical location.

Whether or not an address manager is assigned for a given administrative region, nodes **should** be assigned addresses from a regional block with which they are registered or affiliated. This will reduce the risk of assigning an address already in use, even absent more active administrative co-ordination.

Entities or individuals wishing to nominate themselves for the role of regional / sub-regional address administrator should contact the NATO C3 Staff (Attn: Network Domain Branch), NATO Headquarters, Brussels, BE.

Within the global-regional blocks, specific 5066 addresses **shall** be assigned to individual nations using the second element “x” of the dotted-decimal address form w.x.y.z. Multiple values of “x” can be pre-assigned in order to give Nx65535 address nodes per nation or organizational unit. Exceptions to this general rule are the US Government and NATO, which will likely be large-scale implementers of the STANAG and are assigned their own unique regional “w” blocks.

Sub-regional blocks allocations are defined in the subsections below. The lists of nations is based on the “Independent States in the World” (URL: http://www.state.gov/www/regions/independent_states.html) as released by the US Office of the Geographer and Global Issues, January 21, 2000.

N.3.1 North-American National Address Schema

Sub-regional assignments for North American national and organizational entities are defined below.

The US Government, as a large scale implementer and deployer of S'5066 systems, is allocated the 1.x.y.z address block and organizational address assignments within this block **shall** conform to the table below.

Table N-2 — North American (US Government) National Addressing Schema (1.x.y.z)

| “1” . “x” | Organization | | “1” . “x” | Organization |
|------------------|------------------------------|--|------------------|--|
| 1.1 | US Navy | | 1.8 | US Joint Forces Commands |
| 1.2 | US Marine Corps | | 1.9 – 19 | Other US Military |
| 1.3 | US Air Force | | 1.20 | US Federal Emergency Management Agency |
| 1.4 | US Army | | 1.21 | BATF |
| 1.5 | US Special Operations Forces | | 1.22 | US Federal Bureau of Investigation |
| 1.6 | US Coast Guard | | 1.23 – 1.255 | Other US Govt |
| 1.7 | US Military Sealift Command | | | |

S'5066 system address allocations for North American nations — excluding systems registered with the US Government — **shall** conform to the table below.

Table N-3 — North American (non-US Government) National Addressing Schema (2-3.x.y.z)

| “w” . “x” | Country | | “w” . “x” | Country |
|------------------|--------------------|--|------------------|-----------------------------------|
| 2.1 | Antigua & Barbuda | | 2.15 | Guatemala |
| 2.2 | Bahamas | | 2.16 | Haiti |
| 2.3 | Barbados | | 2.17 | Honduras |
| 2.4 | Belize | | 2.18 | Jamaica |
| 2.5, 2.6, 2.7 | Canada | | 2.19 | Mexico |
| 2.8 | Costa Rica | | 2.20 | Nicaragua |
| 2.9 | Cuba | | 2.21 | Panama |
| 2.10 | Dominica | | 2.22 | St. Kitts and Nevis |
| 2.11 | Dominican Republic | | 2.23 | St. Lucia |
| 2.12 | Ecuador | | 2.24 | Trinidad & Tobago |
| 2.13 | El Salvador | | 3.1 | United States (non-US Government) |
| 2.14 | Grenada | | | |

N.3.2 South American National Address Schema

S'5066 system address allocations for South American nations **shall** conform to the table below

Table N-4 — South American National Addressing Schema (4.x.y.z)

| “w” . “x” | Country | | “w” . “x” | Country |
|------------------|----------------|--|------------------|----------------|
|------------------|----------------|--|------------------|----------------|

| | | | | |
|-----|-----------|--|------|-----------|
| 4.1 | Argentina | | 4.7 | Guyana |
| 4.2 | Bolivia | | 4.8 | Paraguay |
| 4.3 | Brazil | | 4.9 | Peru |
| 4.4 | Chile | | 4.10 | Suriname |
| 4.5 | Colombia | | 4.11 | Uruguay |
| 4.6 | Ecuador | | 4.12 | Venezuela |

N.3.3 NATO Address Schema

NATO is expected to be a large user of S'5066 systems and consequently has its own top-level address block assigned to it. Sub-allocations within this address block are made to NATO nations and commands and **shall** conform to the table below.

Table N-5 — NATO Addressing Schema (5.x.y.z)

| "w" . "x" | Country | "w" . "x" | Country |
|-----------|----------------|--------------|---|
| 5.1 | Belgium | 5.12 | Norway |
| 5.2 | Czech Republic | 5.13 | Poland |
| 5.3 | Denmark | 5.14 | Portugal |
| 5.4 | France | 5.15 | Spain |
| 5.5 | Germany | 5.16 | Turkey |
| 5.6 | Greece | 5.17 | United Kingdom |
| 5.7 | Hungary | 5.18 | United States |
| 5.8 | Iceland | 5.19 | Allied Command for Operations |
| 5.9 | Italy | 5.20 | Allied Command for Transformation |
| 5.10 | Luxembourg | 5.21 | NATO CIS Services Agency |
| 5.11 | Netherlands | 5.22 | NATO Consultation, Command and Control Agency |

Allocations within the NATO block are made for national forces provided in support of NATO operations, e.g., to a NATO Response Force (NRF), Combined Joint Task Force (CJTF) or Standing NATO Force (SNF). National forces participating in a NATO mission or operation **should** use addresses in the NATO block. However, nations belonging to NATO **may** elect to use addresses within their own allocated block instead the NATO allocation. The option to use a nationally administered address or a NATO administered address for any given system should be resolved as part of the operational planning for any mission in which the system participates.

N.3.4 European National Address Schema

S'5066 system address allocations for European nations **shall** conform to the table below.

Table N-6 — European National Addressing Schema (6-7.x.y.z)

| “w” . “x” | Country | | “w” . “x” | Country |
|------------------|----------------------|--|------------------|---------------------|
| 6.1 | Albania | | 6.27 | Lithuania |
| 6.2 | Andorra | | 6.28 | Luxembourg |
| 6.3 | Austria | | 6.29 | FYROM ³ |
| 6.4 | Belarus | | 6.30 | Malta |
| 6.5 | Belgium | | 6.31 | Moldova |
| 6.6 | Bosnia & Herzegovina | | 6.32 | Monaco |
| 6.7 | Bulgaria | | 6.33 | Netherlands |
| 6.8 | Croatia | | 6.34 | Norway |
| 6.9 | Cyprus | | 6.35 | Poland |
| 6.10 | Czech Republic | | 6.36 | Portugal |
| 6.11 | Denmark | | 6.37 | Romania |
| 6.12 | Estonia | | 6.38, 39, 40 | Russia |
| 6.13 | Finland | | 6.41 | San Marino |
| 6.14, 6.15 | France | | 6.42 | Slovakia |
| 6.16, 6.17 | Germany | | 6.43 | Slovenia |
| 6.18 | Greece | | 6.44 | Serbia & Montenegro |
| 6.19 | Holy See | | 6.45 | Spain |
| 6.20 | Hungary | | 6.46 | Sweden |
| 6.21 | Iceland | | 6.47 | Switzerland |
| 6.22 | Ireland | | 6.48 | Turkey |
| 6.23, 6.24 | Italy | | 6.49 | Ukraine |
| 6.25 | Latvia | | 6.50, 51, 52 | United Kingdom |
| 6.26 | Liechtenstein | | 6.53 | Yugoslavia |

³ Turkey recognises the Republic of Macedonia by its constitutional name

N.3.5 Asian National Address Schema

S'5066 system address allocations for Asian nations **shall** conform to the table below.

Table N-7 — Asian National Addressing Schema (8-9.x.y.z)

| “w” . “x” | Country | | “w” . “x” | Country |
|------------------|----------------|--|------------------|----------------|
| 8.1 | Bangladesh | | 8.12 | Malaysia |
| 8.2 | Bhutan | | 8.13 | Maldives |
| 8.3 | Brunei | | 8.14 | Mongolia |
| 8.4 | Cambodia | | 8.15 | Myanmar |
| 8.5 | China | | 8.16 | Nepal |
| 8.6 | India | | 8.17 | Phillipines |
| 8.7 | Indonesia | | 8.18 | Singapore |
| 8.8 | Japan | | 8.19 | Sri Lanka |
| 8.9 | Korea, North | | 8.20 | Taiwan |
| 8.10 | Korea, South | | 8.21 | Thailand |
| 8.11 | Laos | | 8.22 | Vietnam |

N.3.6 African National Address Schema

S'5066 system address allocations for African nations **shall** conform to the table below.

Table N-8 — African National Addressing Schema (10.x.y.z)

| “w” . “x” | Country | | “w” . “x” | Country |
|------------------|-----------------------------|--|------------------|----------------|
| 10.1 | Algeria | | 10.27 | Lesotho |
| 10.2 | Angola | | 10.28 | Liberia |
| 10.3 | Benin | | 10.29 | Madagascar |
| 10.4 | Botswana | | 10.30 | Malawi |
| 10.5 | Burkina Faso | | 10.31 | Mali |
| 10.6 | Burundi | | 10.32 | Mauritania |
| 10.7 | Cameroon | | 10.33 | Mauritius |
| 10.8 | Cape Verde Islands | | 10.34 | Morocco |
| 10.9 | Central African Republic | | 10.35 | Mozambique |
| 10.10 | Chad | | 10.36 | Namibia |
| 10.11 | Comoros | | 10.37 | Niger |
| 10.12 | Congo (Brazzaville) | | 10.38 | Nigeria |
| 10.13 | Congo (Kinshasha) | | 10.39 | Rwanda |
| 10.14 | Côte d'Ivoire | | 10.40 | Sao Tome |
| 10.15 | Djibouti | | 10.41 | Senegal |
| 10.16 | Egypt | | 10.42 | Sierra Leone |
| 10.17 | Equatorial Guinea | | 10.43 | Somalia |
| 10.18 | Eritrea | | 10.44 | South Africa |
| 10.19 | Ethiopia | | 10.45 | Sudan |
| 10.20 | Gabon | | 10.46 | Swaziland |
| 10.21 | Gambia | | 10.47 | Tanzania |
| 10.22 | Gambia | | 10.48 | Togo |
| 10.23 | Ghana | | 10.49 | Tunisia |
| 10.24 | Guinea | | 10.50 | Uganda |
| 10.25 | Guinea-Bissau | | 10.51 | Zambia |
| 10.26 | Kenya | | 10.52 | Zimbabwe |

N.3.7 Middle East National Address Schema

S'5066 system address allocations for Middle Eastern nations **shall** conform to the table below.

Table N-9 — Middle East National Addressing Schema (11.x.y.z)

| “w” . “x” | Country | | “w” . “x” | Country |
|------------------|----------------|--|------------------|----------------------|
| 11.1 | Afganistan | | 11.12 | Lebanon |
| 11.2 | Armenia | | 11.13 | Oman |
| 11.3 | Azerbaijan | | 11.14 | Pakistan |
| 11.4 | Bahrain | | 11.15 | Qatar |
| 11.5 | Georgia | | 11.16 | Saudi Arabia |
| 11.6 | Iran | | 11.17 | Syria |
| 11.7 | Iraq | | 11.18 | Tajikistan |
| 11.8 | Jordan | | 11.19 | Turkmenistan |
| 11.9 | Kazakhstan | | 11.20 | United Arab Emirates |
| 11.10 | Kuwait | | 11.21 | Uzbekistan |
| 11.11 | Kyrgyzstan | | 11.22 | Yemen |

N.3.8 Australia, New Zealand, and Oceania National Address Schema

S'5066 system address allocations for Australia, New Zealand and nations in Oceania **shall** conform to the table below.

Table N-10 — Australia, New Zealand and Oceania National Addressing Schema
(12.x.y.z)

| “w” . “x” | Country | | “w” . “x” | Country |
|------------------|------------------|--|------------------|------------------|
| 12.1 | Australia | | 12.9 | Papua New Guinea |
| 12.2 | Fiji | | 12.10 | Samoa |
| 12.3 | Kiribati | | 12.11 | Seychelles |
| 12.4 | Marshall Islands | | 12.12 | Solomon Islands |
| 12.5 | Micronesia | | 12.13 | Tonga |
| 12.6 | Nauru | | 12.14 | Tuvalu |
| 12.7 | New Zealand | | 12.15 | Vanuatu |
| 12.8 | Palau | | | |

N.3.9 Non-Governmental Organization (NGO) and Other Address Schema

S'5066 system address allocations for Non-Governmental Organizations and other entities **shall** conform to the table below.

Table N-11 — Non-Governmental Organization and Other Addressing Schema (13-15.x.y.z)

| "w" . "x" | NGO | | "w" . "x" | NGO |
|-----------|----------------|--|-----------|-----|
| 13.1 | United Nations | | | |
| 13.2 | Red Cross | | | |
| | | | | |
| 14.0-255 | Other | | | |
| | | | | |
| 15.0.255 | Other | | | |

N.4 IP-CLIENT/ETHER-CLIENT ADDRESS MANAGEMENT

The provisions of this section apply to the management of Internet Protocol Addresses assigned to IP-client [4], PPP-client [5] or Ether-Client [6] interfaces used with STANAG 5066.

N.4.1 Representation of a STANAG 5066 Node Addresses as a Pseudo IEEE 802 48-bit Media-Access-Control (MAC) address

A number of features for IP address management (e.g., the Address Resolution Protocol for IP Version 4 or Stateless Autoconfiguration for IP Version 6 [7]) presume availability of a media-access control (MAC) address in the 48-bit format prescribed by IEEE-802 or Ethernet.

Pseudo-Ethernet (48-bit) addresses required for IPv4 or IPv6 address management functions should be derived from the full-length STANAG 5066 node address in accordance with Annex F section F.11.5.4, Figure – 17. This address is referred also to in this Annex as the interface identifier for the IP-client or Ether-client, as appropriate to the discussion.

⁴ Per the specification of Annex F section F.12

⁵ Per the specification of Annex F section F.11

⁶ Per the specification of Annex F section F.10

⁷ RFC 2462, *IPv6 Stateless Address Autoconfiguration*