Modeling QoS Provision on Adaptable Communication Environments

Antônio Tadeu A. Gomes, Sérgio Colcher, Luiz Fernando G. Soares
{atagomes, colcher, lfgs}@inf.puc-rio.br
Departamento de Informática — PUC-Rio
R. Marquês de São Vicente, 225, Rio de Janeiro — RJ, 22453-900, Brazil

***Abstract*—This paper proposes an approach for representing and programming QoS functions in communication systems. It presents a model that gives adequate support for defining: (i) communication environments and their adaptation mechanisms, and (ii) frameworks that delineate QoS-specific abstractions that appear within any communication environment.**

I. INTRODUCTION

The planning, design, implementation and deployment of communication systems have become complex enough to justify the creation of a "Telecommunications Service Engineering" [11]. The concepts, principles and rules of this new discipline have been mostly borrowed from software engineering. However, it has its own requirements and should also incorporate results from other areas, such as intelligent, active and programmable networks [7].

The large diversity of system requirements brings many difficulties to the Telecommunications Services Engineering. An approach which is being recently considered is to devise models in which communication service platforms support some kind of *programmability* or *adaptability* to meet both the wide range of different demands and the presence of run-time changes in applications requirements and, in addition, the fast technological evolution.

This paper focus on the QoS provision in programmable and adaptable service programming platforms. Its main goal is to propose appropriate abstractions for representing and programming QoS functions. Despite having some similarities with (and being based on) other existing proposals, this work presents two additional contributions.

First, it aims at defining a *recursive structuring model* [4], named *Service-Composition Model* (SCM) that gives enough support for defining both telecommunications services and their adaptation mechanisms. This approach is a first step towards homogenizing the treatment of various aspects related to service programmability and adaptability, in particular QoS provision.

As a second contribution, this work provides a kind of abstract language that allows service designers to describe a particular telecommunications system architecture and, moreover, reuse some interfaces and data structures commonly found in distinct parts of this architecture. We envision that the model alone is not sufficient to help service designers build communication systems, since more structuring is desired to regulate the adaptations. Our proposal relies on the use of frameworks [9] as a strategy to delineate aspect-specific interfaces and data structures that appear in any service provider. Various related frameworks are being developed together with the model, including the QoS-related frameworks presented in this paper. Indeed, we show by means of these frameworks how QoS functions are recurrent on the several sorts of providers that exist in a communication system, and how they participate in the overall QoS orchestration.

II. RELATED WORK

Most of the previous development involving QoS provision has focused only on specific parts of a protocol architecture [3] [10] and not on the integrated architecture as a whole. They focus on a layering principle that emphasizes horizontal communications between protocol peer entities. However, like OSI-RM, they do not take into account the vertical communication between entities of adjacent layers. There is also important work concerning QoS provision on operating systems [8] [5]. Nevertheless, when we consider communication systems where end-to-end QoS guarantees are necessary, all elements must be carefully integrated to really achieve the QoS required by the end users.

Focusing on end-to-end QoS provision, several research groups have proposed some approaches to treat the system architecture as a whole, being collectively denominated QoS architectures [1]. These architectures try to consider the various aspects and parts of a system involved in the end-to-end communication, as illustrated in Fig. 1.

The several QoS architectures have in common the definition of sets of interfaces, generally implemented as programming libraries. By defining proper orchestration

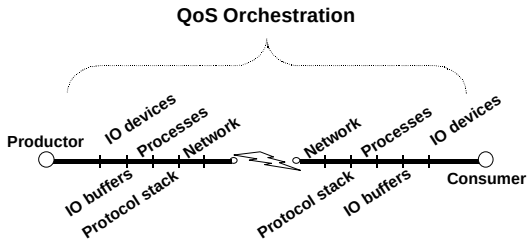

Figure 1 - QoS end-to-end requirements (extracted from [2])

algorithms to manage these interfaces, communication systems can provide end-to-end QoS guarantees. The ever increasing diversity of QoS characteristics required by applications and, therefore, how these characteristics will be treated at different parts of the system is one of the greatest difficulties in attaining QoS orchestration through the QoS architecture approach. The limitations imposed are strictly related to the lack of ways to adapt a service during its operation.

Other proposals to the end-to-end QoS provision are based on the concept of generic communication environments. Above these environments, services are implemented through creation and association of components. Programming support offered by environments defined above CORBA and ODP, such as those of the TINA Model [2] and the Binding Model [6], are the most relevant examples of this strategy.

As far as we know, none of the current approaches presents a recursive structuring model as this paper proposes — based on the service provider abstraction — that allows homogenizing the treatment of various aspects related to service programmability and adaptability, in particular QoS provision. In addition, the present paper offers a strategy to delineate interfaces and data structures that appear in any service provider to regulate QoS adaptations.

III. Service Composition Model

The *Service Composition Model* (*SCM*) defines two basic elements: (i) *user components*, which correspond to entities that make direct use of services, and (ii) *providers*, which are responsible for the service provision. Both components and providers can be either hardware or software, and their association defines the concept of a *service-offering environment*, as exemplified in Fig. 2.

In SCM, the service provider abstraction is defined without any concern with the dispersion of its user components. For instance, components can be as close as objects contained in a single process or as far as objects located on internetworked machines. A special abstraction called *MediaPipe* (shown in Fig. 3) is created to capture the idea of a virtual resource over which QoS specifications can

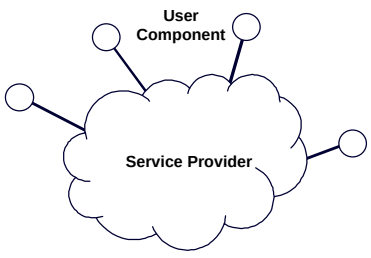

Figure 2 - Service-offering environment.

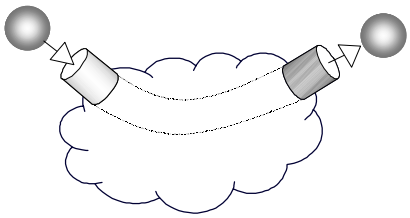

Figure 3 - Graphical representation of a MediaPipe

be defined and handled. MediaPipes can be point-to-point or multipoint and are defined by service provider elements as follows.

Special elements named *service components* accomplish the internal implementation of a provider. In order to perform their functions, service components may communicate with each other using more primitive providers, called *infrastructure providers*, as shown in Fig. 4(a). Besides the infrastructure provider, every provider must offer a way for user components to communicate with service components, in order to request services. *Access service providers* furnish this feature.

The infrastructure and access providers can also be structured so that the service components pictured in Fig. 4(a) act as user components. Within these providers, other service components, infrastructure and access service providers show up. This nested organization has some similarities with the layering principle defined by OSI-RM. The Fig. 4(b) suggests this analogy by illustrating the abstraction of a MediaPipe *C* between user components *A* and *B*. Once *C* internal structure is revealed, we can see that *C* is implemented by making use of four MediaPipes, two vertical (related with the access providers) and two horizontal (related with infrastructure providers), in addition to service components X, Y and Z.

However, SCM introduces other characteristics absent in the OSI-RM. For example, the local system environments are outside the scope of OSI-RM. By representing them using the abstraction of access service providers, SCM presents to service designers a way to model QoS provision in an end-to-end scale. Since SCM suggests that the abstraction of a provider may represent all types of communication, it also has a strong potential for design and implementation *reuse*. For instance, we can define appropriate interfaces and data structures to generically model QoS provision in any sort of provider. This homogenization lessens the work of introducing new QoS requirements and changing existing ones. In addition, it is suited for modeling QoS orchestration, since QoS functions will expose similar interfaces and data structures that will facilitate the implementation of algorithms to offer end-to-end QoS.

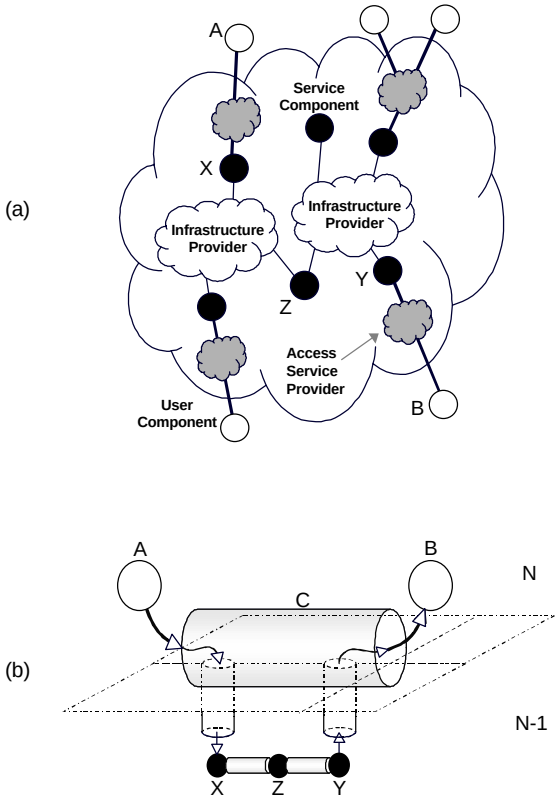

Figure 4 - The provider structure

A. *Meta Level Services*

In order to be adaptable, a service-offering environment needs a special *meta service* that acts upon the main service provider and its elements, allowing adaptations. Examples of common meta services include the signaling mechanisms and the routing protocols.

Meta services can perform adaptations on services pertaining to any level of nesting. Moreover, the same provider can be the target of several meta services and the same meta service can act upon multiple providers, regardless of their levels (see Fig. 5). In any of these cases, the meta service is structured, as any service, according to SCM. An example of the relationship between meta services and target services is shown in Section IV.

Meta services can also be targets of other meta services, constituting what should be called a *meta service tower*. More details about SCM can be found in [4].

IV. USING FRAMEWORKS TO MODEL QOS PROVISION

Although SCM provides a method for defining both communication services and their adaptation mechanisms, more structuring is desired to regulate these adaptations.

Frameworks capture design decisions that are common to a particular domain. Usually, various parts of a framework cannot be previewed, so they should be left incomplete or "subject to variations". These *hot-spots* [9] allow the definition of adaptable structures and interfaces regardless of the specificities of the domain.

There are parts of a service in which adaptations should be done before operation. In these parts, hot-spots are only related to the providers dispersion level, that is, the specific environment in which the service is inserted. There are other parts, however, that should be maintained flexible during service operation, such as the points in the service that are programmed to deal with different QoS requirements. Indeed, the QoS frameworks presented in this paper provide a single QoS model from which some of the hot-spots are responsible for regulating service adaptability while others are in charge of treating environment specific issues.

A. *Phases of QoS Provision*

During service operation, QoS provision can be divided in two main phases: (i) the *QoS negotiation* and (ii) the *QoS maintenance*. The structures and interfaces that appear in these phases are targets of the *QoS frameworks*. Negotiation and maintenance mechanisms, together with the target service, make up a meta service tower responsible for establishing and controlling QoS adaptations, as can be seen in Fig. 5.

The QoS negotiation meta service is responsible for the creation of MediaPipes. This involves, recursively, the creation of MediaPipes in the access service and infrastructure providers, as already illustrated in Fig. 4. The creation process begins at *admission controllers*, which have the task of testing the feasibility of the demand[1]. The admission controllers start a set of mechanisms whose function is to identify the involved internal providers and, if the creation is possible, to set a portion of the QoS provision responsibility to each of them. Among these mechanisms, the *routing agents*, the *negotiators* and the *mappers* play central roles.

Routing agents are responsible for choosing the adequate internal providers based on the "reachability" information of the user components. Routing agents use a routing database, which is constructed by the routing meta service, in order to choose one route that satisfies the desired QoS. The routing meta service is modeled by multicast frameworks also defined in [4].

[1] A MediaPipe creation request includes, besides other information, the load characterization and the desired QoS specification, as described in [4]

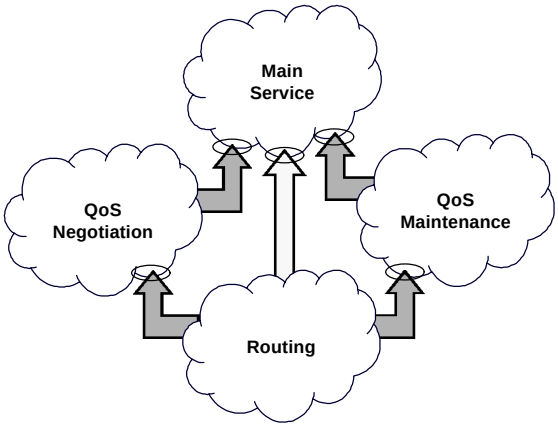

Figure 5 - QoS meta service tower

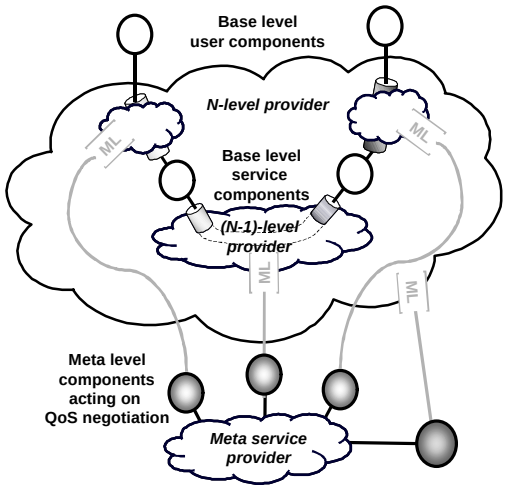

Figure 6 - QoS negotiation

Negotiators take part on the QoS orchestration by defining portions of responsibility for each internal provider in the chosen route. As different providers participate in this process and each of them may have a distinct QoS view (as discussed later), it is necessary to translate the defined portion of QoS responsibility on these distinct views, what is done by the *mappers*.

The overall negotiation process is repeated recursively in each internal provider — by means of their own admission controllers — until a dispersion level at which the admission controllers can perform the admission tests on primitive resources. Primitive resources are those in which there are no further internal providers to which requests should be forwarded to. If the tests indicate that resources are available, the admission controllers reserve the necessary resources and return an affirmative response. Recursively, the admission controllers of a provider will receive responses from all the internal providers. If all responses are affirmative, the admission controllers reserve the necessary resources in each of their internal providers and return an affirmative response to their predecessor provider in the nesting. If, at any level, one or more internal providers cannot offer the desired QoS guarantees defined by the negotiators, the admission controllers can recalculate the division of responsibility (through the negotiators), or even compute another alternative route of internal providers (by means of the routing agents). Fig. 6 presents a schematic view of the negotiation process.

The provider must guarantee the maintenance of the negotiated QoS for the whole MediaPipe life cycle. QoS maintenance may result in the creation, modification or destruction of MediaPipes. Among the mechanisms in charge of QoS maintenance, we can highlight the *monitors*, the *tuners* and the *adjustment controllers*.

Monitors keep track of the effective load generated by user components in a MediaPipe and of the QoS actually offered by the provider. If the MediaPipe cannot honor its QoS agreement (because of a resource overload, for example), the monitors must activate the tuners. The *tuners* must then reapply the QoS orchestration among the internal providers through a process similar to that one implemented by the negotiators. The *adjustment controllers* (similar to the admission controllers) take care of all the maintenance process, receiving adjustment responses from internal adjustment controllers and, in case of QoS violations, either reallocating resources (through the tuner) or signaling those responses to higher level adjustment controllers. Again, in a recursive process, these adjustment controllers may try to change the division of QoS maintenance responsibility (through the tuner) or even ask for new routes, if a choice of alternative providers shows it is necessary.

Based on the QoS functions present in the SCM meta service tower discussed in this section, QoS frameworks can be proposed as following.

B. Resource Modeling

The negotiation process translates requests for the creation of MediaPipes to resource reservation and allocation. Similarly, the QoS maintenance acts ultimately on resource allocation and scheduling control. In fact, resource-partitioning mechanisms play a central role, since they are in general the real targets of service adaptations related to QoS provision. Thus, a *resource modeling* framework is crucial to indicate the points in a target service upon which the QoS negotiation and maintenance will act.

In the framework, each resource is associated with a *virtual resource tree*. As depicted in Fig. 7, the leaves of the tree are the *virtual resources* through which the MediaPipes have access rights. The root of the tree corresponds to a basic component, called *resource scheduler*, responsible for directly managing the resource partitioning among virtual

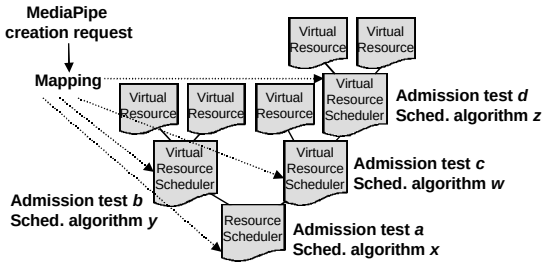

Figure 7 - A virtual resource tree

resources. The other internal nodes of the tree are specialized virtual resources, internal to the provider, called *virtual resource schedulers*. The main function of a virtual resource scheduler is to grant its children permission to use its resource capacity portion.

Each internal node of the tree, no matter if it is the resource scheduler or a virtual resource scheduler, is associated with an admission test and a scheduling algorithm that control the resource capacity partitioning. Both functions, together with the flexible tree structure, are hot-spots responsible for permitting service adaptability. CPU schedulers and virtual circuit switches are examples of resource-partitioning mechanisms. Questions pertaining to specific resource characteristics, such as activation and deactivation of virtual resources (for instance, CPU context switching among threads), are treated in the resource-modeling framework as hot-spots that model environment idiosyncrasies.

C. QoS Orchestration

The orchestration is the result of the joint action of the mechanisms involved in the negotiation and maintenance phases. These mechanisms depend upon concatenation heuristics that decide quantitatively how the portions of QoS provision responsibility will be delegated among internal providers. Hot-spots take care of specializing the framework to the particularities of each phase. The *MediaPipe concatenation* concept illustrated in Fig. 4(b) and Fig. 6 provides the most important abstraction of the QoS orchestration framework.

The following example provides a better discussion about the concatenation concept. Let us assume that there is a request for a MediaPipe whose transfer delay must be limited to x time units. The n internal MediaPipes that compose the main MediaPipe have to be configured such as each MediaPipe i is granted a portion x / p_i of the total time x, given that

$$\sum_{i=1}^{n} 1/\, p_i = 1. \qquad (1)$$

The internal providers see these portions (possibly mapped to agree with proper QoS views) as requests for the creation of internal MediaPipes. In this example, the concatenation heuristic associated with the negotiator of the main provider is in charge of determining p_i in (1). Likewise, the tuner mechanism may use another concatenation heuristic to recalculate p_i, perhaps taking into account further constraints imposed by monitoring, as for example $p_i < P_i$, where P_i designates the best performance (in terms of low delay) offered to the internal MediaPipe i.

As expected, the concatenation heuristics show up as important hot-spots for service adaptability. The way these heuristics are implemented depends on whether the mechanisms are centralized or distributed. Indeed, the dispersion of the negotiation and maintenance mechanisms is an important design decision that can deeply affect their efficiency. This decision is also modeled by the QoS orchestration framework with the help of proper hot-spots.

D. QoS Parameterization

Section IV-A stressed the need of translating QoS views from the provider to its internal providers, what is done by the *mappers and monitors*. As an example of different views, a video-on-demand application can compute the quality of a video stream in terms of frame size and frame rate, whereas a network protocol can be designed to be QoS-configurable with regard to the maximum bit rate.

Any information concerning QoS configuration can be structured by means of *service characterization parameters*. In order to make the parameterization task flexible enough, a parameterization scheme is also defined with the help of a framework.

More details about the specification of the frameworks can be found in [4].

V. Conclusion

Most of the approaches for QoS provision in communication systems have at least one the following drawbacks: (i) a narrow scope, considering isolated QoS characteristics of operating systems and protocol architectures; (ii) QoS orchestration through the management of heterogeneous interfaces, which is hard to implement; and (iii) lack of appropriate abstractions for service adaptability. This paper presented an alternative that allows a finer-grained and more homogeneous abstraction of the main QoS functions spread throughout any communication system. The proposed approach defines a conceptual structuring model which permits the design of communication systems recursively, thus improving the reuse of interfaces and data structures commonly found in

several parts of a communication system. Above this main abstraction, frameworks are used to sketch the sets of reusable interfaces and data structures related to QoS provision and, moreover, to emphasize the points in which these elements can be adapted to new QoS demands.

At present, we have been working on validating the model and the QoS frameworks by means of two main experiences. First, we are working on the definition of an homogeneous abstraction for QoS representation in the Internet. Basically, this research aims at modeling the int-serv and diff-serv architectures according to SCM, using its features to form a conceptual boundary among the service definitions, the signalling protocols and the set of QoS functions performed in switched routers.

The second experience is related with QoS provision in hypermedia systems. Its main goal is to define appropriate QoS orchestration abstractions among multimedia client applications, multimedia servers and the communication infrastructure. This involves the deployment of a hypermedia protocol that provides, among other functions, QoS mapping between client requirements and internal parameters. In addition, this research focus on application-level QoS orchestration between multimedia clients and servers, which encompasses issues such as temporal inter-media synchronization, prefetch of multimedia objects and elastic adjustment of multimedia presentations.

REFERENCES

[1] Aurrecoechea, C., et al. "A survey of QoS architectures". *ACM Multimedia Systems Journal (Special Issue on QoS Architecture)*, 138-151. May 1998.

[2] Berndt, H., et al. "Service specification concepts in TINA-C". *International Conference of Intelligence in Broadband Services and Networks*. 1994.

[3] Braden, R., et al. "Resource reservation protocol (RSVP) – version 1 – functional specification". *RFC 2205*. 1997.

[4] Colcher, S., Gomes, A. T. A., Rodrigues, M. A. A., Soares, L. F. G. "Modeling service aspects in generic communication environments". *Technical Report*, Laboratório TeleMídia, PUC-Rio. 2000.

[5] Goyal, P., Guo, X., Vin, H. M. "A hierarchical CPU scheduler for multimedia operating systems". *Operating Systems Design and Implementation (OSDI'96)*. 1996.

[6] Lazar, A. A., Lim, K. S., Marconcini, F. "Binding model: motivation and description". *Technical Report CTR 411-95-17*, Comet Group, Center for Telecomunications Research, Columbia University. 1995.

[7] Campbell et al. "A survey of programmable networks". *ACM Computer Communications Review*. Apr. 1999.

[8] Mauthe, A., Coulson, G. "Scheduling and admission testing for jitter constrained periodic threads: discussion and proof". *Technical Report MPG-96-12*, Distributed Multimedia Research Group, Lancaster University. 1996.

[9] Pree, W. *Framework patterns*. SIGS Management Briefings. SIGS Books & Multimedia. 1996.

[10] Schulzrinne, H., Casner, S. "RTP: A transport protocol for real-time applications". *RFC 1889*. 1995.

[11] Znaty, S., Hubaux, J. "Telecommunications services engineering: Principles, architectures and tools". *Proceedings of the ECOOP'97 Workshops*, Jyväskylä, Finland. Jun. 1997.