Quartz: Supporting QoS-Constrained Services in Heterogeneous Environments

Frank Siqueira and Vinny Cahill
Distributed Systems Group, Department of Computer Science, Trinity College Dublin, Ireland
E-Mail: Frank.Siqueira@cs.tcd.ie, Vinny.Cahill@cs.tcd.ie

Abstract

This paper describes an architecture that addresses common problems found in systems supporting QoS specification and enforcement, such as lack of flexibility and expressiveness in the specification of requirements and dependencies on specific platforms. The Quartz QoS architecture aims to solve these problems by adopting a highly extensible and platform-independent design.

1. Introduction

'Quality of Service', or QoS for short, represents the set of requirements imposed by a user (human being or software component) on the behaviour of services being provided to an application. In order to achieve the level of QoS requested by users, applications such as distributed multimedia, including video on demand, media broadcast and video telephony, as well as real-time control systems, which are migrating from embedded architectures to open environments, require that the access to resources provided by the underlying system support occur in a predictable fashion.

Despite the evolution of computing platforms, computational resources are still scarce because of the increased complexity of applications such as those described above. Modern networks and operating systems provide predictable behaviour through the use of resource reservation mechanisms. However, most applications are still being adapted to make use of these mechanisms in order to support QoS specification and enforcement.

QoS architectures consist in middleware software that provides applications with mechanisms for QoS specification and enforcement, organising the resources provided by the system with the intent of fulfilling the QoS requirements imposed by the user. Substantial work on QoS architectures can be found in the literature [1][2][3]. However, the architectures proposed so far consider only part of the overall problem of QoS specification and enforcement [4].

QoS specification can be done expressing requirements at different levels of abstraction: at the application level (e.g. stating the quality desired from a video application) or at the system level (e.g. the network bandwidth or processing time required by the application). Requirements specified at different levels are related, but differ strongly in their interpretation. For the user it is easier to abstract from the system level and concentrate on his own view of quality at high level. However, many QoS architectures do not provide mechanisms for mapping QoS requirements between different levels of abstraction, forcing the user to deal with a system-level notion of quality that may not be clear for him.

The specification of QoS requirements at application level creates another problem. QoS requirements are represented in different manners for distinct application areas. Consequently, a QoS parameter that is appropriate for describing video quality (such as 'window size' or 'colour depth') may be inadequate for other application areas such as audio broadcast and real-time systems. The adoption of a limited set of QoS parameters for QoS specification constrains the expressiveness of the user and limits the range of applications that could make use of a QoS architecture.

Furthermore, most of the QoS architectures proposed so far target only a particular configuration of processing and communication hardware. This tight dependency on a specific platform constrains their application in open environments, where heterogeneity is always present. Real-time operating systems combined with ATM are the most popular platforms for the development of QoS architectures because of their suitability for the implementation of QoS mechanisms for resource reservation [1][2]. Some architectures are also targeted at particular application areas, with distributed multimedia being the one where the technology is most mature [5].

In this paper we present Quartz, a generic QoS architecture for open systems which addresses the limitations of previous proposals in this area. The main requirements considered in the design of Quartz were:

- support for high expressiveness, allowing users to specify QoS requirements according to the notion of quality that is appropriate at application level;
- transparency of the characteristics of reservation mechanisms and platforms present at lower levels;
- adequacy to open systems, in which different protocols and hardware coexist; and
- support for dynamic resource adaptation to be performed by the system without loss of service consistency at application level.

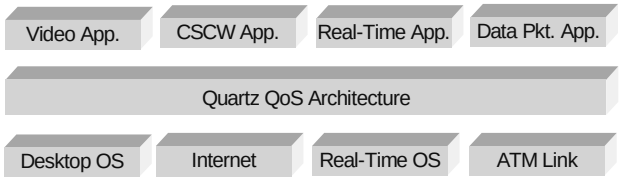

Figure 1. Quartz in a heterogeneous environment

The remaining of this paper is organised as follows. Section 2 presents our proposed architecture, and section 3 describes a prototype implementation and its integration into a CORBA-based environment. Finally, section 4 presents some conclusions and plans of future work.

2. The Quartz Architecture

In order to fulfil the requirements listed previously, we have adopted a highly flexible, extensible, and platform-independent architectural design that allows us to support different application fields and lower-level platforms. The development of an architecture with these characteristics represents an important challenge in this area of research.

Figure 1 illustrates the use of Quartz in a heterogeneous environment. Applications requiring QoS enforcement use the mechanisms provided by Quartz to specify their requirements. To provide the required QoS, Quartz employs the resource reservation protocols available in the target network and operating system.

In order avoid having a translator for each combination of application field and reservation protocol, we have adopted a three-step translation process. Application-specific QoS parameters are first translated into generic application-level parameters, which are then translated into generic system-level parameters and balanced between the network and the operating system. Finally, generic system-level parameters are translated into the lower-level parameters understood by each of the reservation protocols used by the application. Since the user deals only with QoS parameters understood at his abstraction level and meaningful for his application field, his power of expression is not affected.

2.1. Architectural Components

The *QoS agent*, the major component of the Quartz architecture, is responsible for implementing the QoS mechanisms necessary for the provision of services with the quality requested by the user. This involves two main tasks: the translation and balancing of QoS parameters between different levels of abstraction, and the interaction with the underlying reservation mechanisms provided by the resource reservation protocols present in the system. Because of the intrinsically open nature of the target environment, different component-specific reservation protocols may be available.

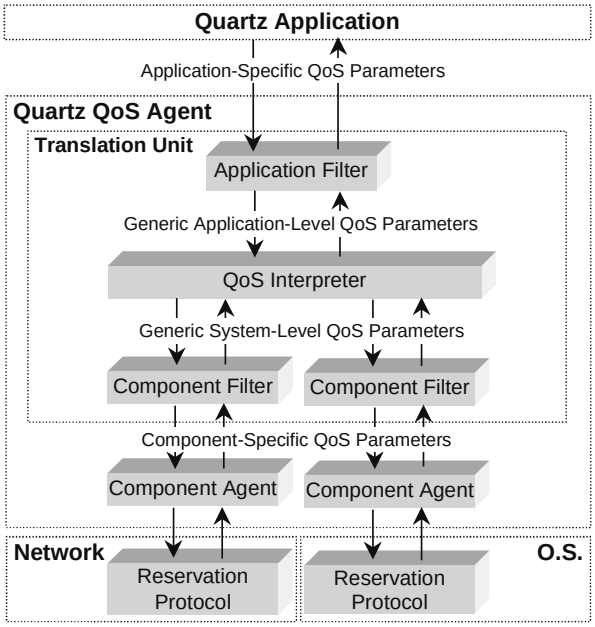

Figure 2. Detailed structure of the QoS agent

The QoS agent, as illustrated by Figure 2, is composed of a *translation unit* and multiple *component agents* associated with the reservation protocols responsible for administering the use of the available resources.

The translation unit contains *QoS filters* and a *QoS interpreter*. QoS filters can be subdivided into application and component filters, which are responsible for translating their respective set of specific QoS parameters to and from the generic set of parameters at the same abstraction level. The QoS interpreter maps between the two sets of generic parameters and balances resources between the network and the operating system.

Finally, the *component agents* get the values of QoS parameters provided by the translation unit and perform the necessary reservation of resources using the corresponding reservation protocol.

2.2. Extending the Architecture

The structure of Quartz is highly portable, reusable and extensible, because the particularities of application fields, system components and reservation protocols are encapsulated by application and component filters, and component agents respectively. Changes at application or system level imply the replacement of filters and component agents, or the plugging of new ones into the QoS agent. Filters are supposed to be simple to implement because they handle parameters described at the same abstraction level. The most complex translation step is executed by the QoS interpreter provided by Quartz. Since component agents are restricted to dealing with a single reservation protocol, the knowledge necessary to implement them is limited.

2.3. Resource Adaptation and Rebalancing

The Quartz architecture provides support for dynamic resource adaptation at the lower level. Resource adaptation results in adjustments in the level of QoS provided by the system to the application. In Quartz, the QoS agent reports dynamic changes in QoS to the user as application-level QoS parameters. Consequently, the translation unit has to provide a reverse translation path, with a set of component-level QoS parameters being translated into application-level QoS parameters.

However, in some cases the resource adaptation can be accommodated at a low level without interfering with the QoS seen by the application. In the Quartz architecture, some QoS requirements such as cost and delay are fulfilled by the sum of resources provided by both the operating system and the network. The component responsible for dividing QoS requirements between components is the *balancing agent*, which is basically a resource trader that is encapsulated by the interpreter.

When one of the operating system or the network reduces the number of resources allocated for the application, the balancing agent execute a process called 'rebalancing of resources'. This process tries to compensate for the loss of resources on the one side requesting more resources from the other. When this is possible, the resource adaptation occurs only at the lower level, and the quality seen by the application is not affected. In this case, the adaptation is completely transparent from the application's point of view.

If rebalancing at the lower level fails, Quartz tells the application to adapt its requirements in order to decrease the consumption of resources. This can be done reducing the quality of a video stream or changing the compression method used for data transfer for example.

3. Example of Application

We have developed a functional prototype of Quartz in order to analyse its behaviour while supporting applications with QoS requirements. This first prototype has component agents for the RSVP protocol [6] and for the real-time mechanisms provided by Windows NT©.

This prototype is being integrated into a complete framework for the deployment of applications with QoS constraints in CORBA-based systems [7]. The proposed framework relies on the audio and video streaming mechanism adopted by OMG for media transfer between objects distributed over the network. For synchronisation of media we have adopted the notification service, an extension to the CORBA event service that allows the specification of constraints on the latency of events propagated between objects.

The CORBA streaming mechanism interacts with Quartz in order to impose QoS constraints on services used by the application. All tasks related to QoS specification and enforcement, including the translation of QoS parameters and reservation of resources, are handled by Quartz. This makes the application more portable and simplifies the job of the programmer. In addition, the QoS agent controls the amount of resources provided by the operating system and the network, and starts adaptation when necessary.

4. Conclusions and Future Work

In this paper we have described Quartz, a QoS architecture that addresses the problems associated with QoS specification and enforcement in heterogeneous environments. Quartz allows the development of distributed applications such as multimedia tools and real-time systems providing QoS-constrained behaviour. The architecture makes the lower-level aspects of resource reservation transparent to the user, while allowing the necessary control through notification in the case of resource adaptation. The design of Quartz allows its easy extension to support new classes of applications, operating systems and communication infrastructures.

We have illustrated the use of Quartz by presenting a framework for developing multimedia applications based on CORBA. This framework, which is currently being implemented, relies on the QoS agent to provide QoS-constrained behaviour in heterogeneous systems. In the sequence, we intend to evaluate the architecture by implementing more applications with QoS requirements and by executing a series of performance measurements.

Acknowledgements

This work is supported by Iona (http://www.iona.com) and Capes (http://www.capes.gov.br).

References

[1] A. Campbell, G. Coulson and D. Hutchison, "A Quality of Service Architecture", ACM Computer Communications Review, Vol. 24(2), pp. 6-27, April 1994.

[2] A. Lazar, K.-S. Lim, and F. Marcocini, "Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture", IEEE Journal of Selected Areas in Communication, No. 7, pp. 1214-27, September 1996.

[3] K. Nahrstedt and J. M. Smith, "The QoS Broker", IEEE Multimedia, Vol. 2(1), pp. 53-67, Winter 1995.

[4] R. Steinmetz and L.C. Wolf "Quality of Service: Where are We?", 5th Int'l Workshop on Quality of Service, May 1997.

[5] A. Vogel et al, "Distributed Multimedia & QoS: A Survey", IEEE Multimedia, Vol. 2(3), pp. 10-19, Summer 1995.

[6] R. Braden et al, "Resource Reservation Protocol: Version 1 Functional Specification", IETF RFC2205, Sept. 1997.

[7] Object Management Group, "OMG Technical Library", http://www.omg.org/library/, 1991-98.