APM

POSEIDON HOUSE • CASTLE PARK • CAMBRIDGE CB3 0RD UNITED KINGDOM
+44 1223 515010 • Fax +44 1223 359779 • Email: apm@ansa.co.uk • URL: http://www.ansa.co.uk

ANSA Phase III

FlexiNet - QoS Investigation

Oyvind Hanssen

Abstract

In this report we investigate what are the main concepts and issues in the support of Quality of Service, focusing at open distributed computing and middleware platforms. We study how these issues are being addressed in research on QoS support related to middleware.

QoS is about non-functional properties in general, but research has been centred around communication protocols and multimedia. The main issues of QoS, related to each RM-ODP viewpoints are discussed. Important issues are QoS specification, QoS provision and how to deal with changing conditions. Research in QoS in middleware platforms is mainly centred around ODP, ANSA and CORBA architectures.

APM.1977.01.00	**Approved** Technical Report	02 June 1997

Distribution:
Supersedes:
Superseded by:

TABLE OF CONTENTS

1 INTRODUCTION

In a distributed system, when considering interactions between clients and server-objects, we sometimes need to be aware of properties which are not part of the object's interface (non-functional properties). Non-functional properties can be as crucial for the usability and correctness of a distributed application as the functional ones. There are many different examples of such properties: For instance in real-time or multimedia systems, the timing and synchronisation of flows or operation-invocations are regarded as important In database systems, atomicity is regarded as important.

Standardisation effort for open distributed computing (ANSA, ODP, CORBA) currently hides away non-functional properties behind the object interfaces. This is in some cases too restrictive. There is a need of platform support for controlling non-functional aspects (Quality of Service) of distributed applications. This has often to be done in an ad hoc manner, which can be difficult and error prone.

1 .1 Goals

The goals of this report is (1) to investigate what are the main concepts and issues in Quality of Service (QoS) support, when focusing on open distributed computing and middleware platforms and (2) how these issues are being addressed in research on QoS support related to middleware. We are also interested in identifying scope for further research on QoS.

1 .2 Report summary

This report uses the concepts and terminology of the ISO/ITU Reference for Open Distributed Processing [RM-ODP]. In section 2, we look at relevant concepts and issues in each of the five RM-ODP viewpoints. QoS is about non-functional properties in general, but research has been centred around communication protocols and multimedia. QoS is an aspect of bindings, but it depends on support from object and interface implementations, to be instantiated. The main issues discussed are related to QoS specification (information viewpoint), QoS provision and how to deal with changing conditions (computational and engineering viewpoint).

In section 3 we look at some of the most significant research in QoS in middleware platforms. This is centred around ODP, ANSA and CORBA architectures. This research mostly take a language based approach to QoS

specification. They address issues of explicit binding, QoS reservation and negotiation, support for change (scaling, adaptation) and engineering support.

In section 4 we briefly summarise state of the art in QoS architecture research, which aims to provide an integrated view of QoS (end-systems, networks, protocols). Good surveys on QoS architectures exist [Aurrecoechea96, Campbell96].

1 .3 Acknowledgements

The author of this report is seconded to the ANSA Phase III programme by the University of Tromso and is supported by a NATO Science Fellowship through the Norwegian Research Council grant no. 116590/410.

Thanks to Andrew Herbert, Billy Gibson, Richard Hayton and Zhixue Wu for valuable advice and comments.

2 CONCEPTS AND ISSUES

In this chapter we look at what are the main concepts and issues related to QoS in open distributed computing. Here it is useful to use the RM-ODP viewpoints to look at different aspects of QoS:

2 .1 Enterprise viewpoint

In the enterprise viewpoint we are interested in what QoS is all about, i.e. the purpose of QoS in a IT-system and who are involved in QoS.

2 .1 .1 What is QoS about

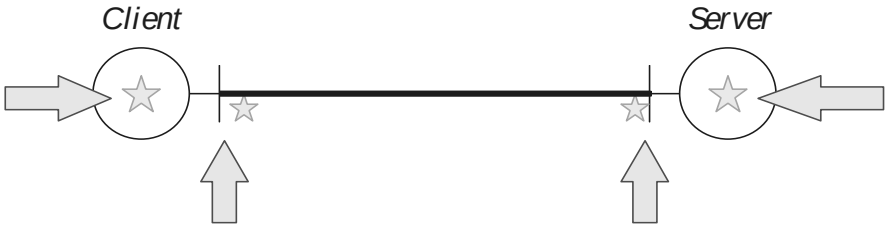
Generally, we say that the term Quality of Service capture non-functional properties such as security, dependability, synchronisation, presentation, performance, etc.

However most research on QoS focus on properties related to real-time and multimedia communications like performance and synchronisation. Therefore this investigation will be focused at multimedia related issues.

2 .1 .1 .1 Interfaces and bindings

QoS are related to bindings, i.e. bindings with different properties may be set up for the same interface, because of different client requirements or different service provided from the network.

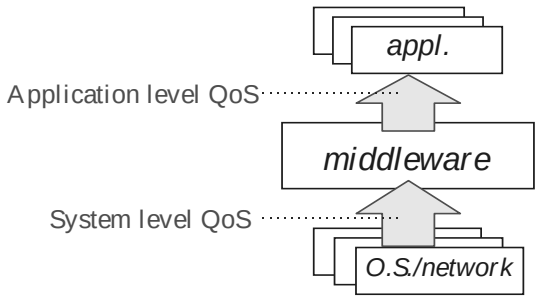
However, those properties of bindings must be supported by the object - implementation, interface implementation or even the client implementation. E.g. transactional bindings are not possible if the object-implementation doesn't have transaction support (logging, stable storage, etc.). Therefore, QoS specification may involve all these. The figure below illustrates how different components participating in a binding (object, interface, client) can have QoS specifications attached to them, i.e. what QoS they are able to support. A required QoS may be instantiated (i.e. a binding established) if the QoS can be supported by all relevant parties.



2 .1 .2 Application vs. system level

When focusing at middleware platforms, it is useful to distinguish between two levels of QoS specification and provision as the figure shows:

- Application level - QoS provided by middleware, to application programs.
- System level - QoS provided by operating system and network, to middleware.



The role of the middleware is to provide abstractions and transparencies to applications programmers, for Quality of Service. The middleware should provide the required properties by controlling resources and their QoS at the system level. There is, however a possible conflict between the use of high level abstractions and transparency, and the need for a flexible and tight control of resources. Operating systems research has addressed related issues for a while (see [Hanssen97]). Some researchers [Engler95] even argue that abstractions should be removed from the O.S. kernel to give applications full control over hardware devices.

2 .2 Information Viewpoint

In the information viewpoint we are interested in the information flowing between QoS users and QoS providers, i.e. the main issue is how QoS is specified.

The purpose of QoS specification is to capture application level QoS requirements and management policy. QoS specification is declarative in nature and is used to select, configure and maintain QoS mechanisms. When focusing at the domain of multimedia flows, it is useful to look at [Campbell,95] which has identified the following five areas of QoS specification:

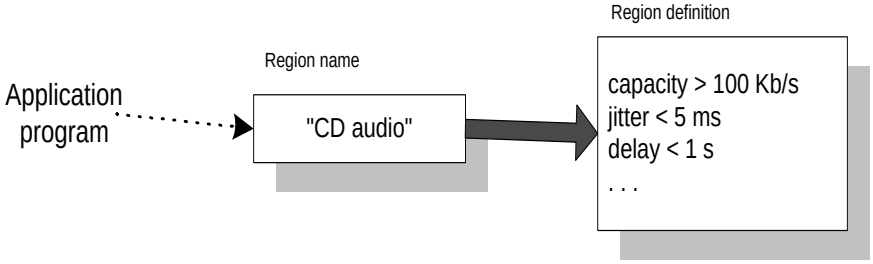
- Flow synchronisation
- Flow performance. The metrics may vary from one application to another. They can be things like throughput, delay, jitter, loss rates.
- QoS commitment policy, i.e. to what extent should the QoS be guaranteed? The choices could e.g. be between deterministic, predictive, adaptive and best effort.
- QoS Management policy, i.e. the degree of QoS adaptation to tolerate. The policy may incorporate scaling actions.
- Cost of service. High degrees of performance or commitments may have a significant cost in resource usage. If cost is a QoS parameter, applications are able to make a trade-off between cost and performance, when negotiating QoS.

2 .2 .1 Key issues in specification

When analysing a concrete model or system we are interested in how the problem of declarative QoS specification is addressed. First the QoS requirements at the application level should be application-oriented rather than system-oriented. This means that the metrics used should be related to user perception or the abstractions application programmers deal with. One could for instance talk about the invocation- or frame-rate, or about CD-Quality versus Phone-Quality. System-oriented means that metrics are related to system resources and their properties. Here, one could for instance talk about bandwidth in terms of bit per second rate or have a choice between a ISDN B-channel versus an ATM connection. A common approach to this is provide mechanisms for mapping application-oriented QoS to system level QoS. This is closely related to QoS transparency, i.e. the application programmer should deal with declarative QoS and not bother with the selection and configuration of mechanisms and resources for QoS provision.

Another issue is what type of parameters that describe the QoS. One choice is between qualitative and quantitative parameters. Qualitative parameters can be simple and intuitive to use. It is typically a choice between predefined qualities, for instance between *'slow'* , *'fast'* or '*very fast*' or between '*Voice*', '*Audio*', and '*CD-Quality-Audio'*. At the other hand there seems to be a need for quantitative specifications (especially for multimedia flows) which may be more concise and more flexible (one could for instance say something like "*speed > 50 and jitter < 5* ").

An approach to this (see section 3.4) is to use qualitative parameters at the application level (region names), which are mapped to quantitative QoS parameters, which could be accurately measured, or which could be easily mapped to system level QoS and resource allocation. An advantage of this approach, is the separation of QoS specifications (i.e. the defining the meaning of region-names) from application programs. The idea is illustrated in the figure below.



A related problem is how to deal with the complex and heterogeneous parameter space. The possible dimensions for describing QoS and the choice of metrics in a given situation can be very numerous. Different parameters may be needed in different situations and the same problem-domain can even render different QoS metrics (if different people are dealing with them). An approach to this problem is to standardise the parameters and metrics used in given situations. For example as QoS domains or classes that define a set of related parameters/metrics and (possibly) a set of related mechanisms [LiOtway94].

When trying to answer the question how QoS is specified, we may divide the research effort into the following categories:

- User/application centred approaches which typically are motivated from the desire to provide manageable abstractions for application programmers and end-users. This is mainly what has been discussed above.
- Network/technology centred approaches which typically are motivated from the need to be able to control QoS of specific technologies. Much effort is based on ATM networks and aims to develop architectures that support the provision of end-to-end QoS guarantees. A well known example is the XRM/XBIND architecture [Lazar95]. QoS is here specified as traffic classes and service classes, which represents statistical models for information streams[1], each with a set of class-specific QoS parameters. Other research based on ATM networks include [Ferrari96] and [Besse94]. Important work is also being done in QoS specification based on the Internet [Braden94].
- Formal mathematical notations, which serve as more generic models or languages for specification of QoS constraints. [Blair94a] propose a QoS constraint language QL, which is based on temporal logic. Other work include [Plagemann95], which propose to specify QoS requirements as Vector Minimum Problems, and the University of Tromso DIME project [Eliassen95], which is developing a formal type model for multimedia flows that incorporate quality attributes.

[1] Traffic classes are low level and system-oriented and service classes are more application oriented (video/audio streams).

2 .3 Computational Viewpoint

In the computational viewpoint we are focusing at the functional decomposition of system, i.e. how is QoS configured and what main components and interactions are necessary to establish and manage QoS.

In addition to the application's objects and interfaces, it is recognised that (as proposed in e.g. [Blair94a]) bindings need to be visible as computational objects (with interfaces) and that establishing of binding should be explicit. This allows one to control the QoS of bindings.

Admission testing may be a part of the binding process. This means that there is a concept of QoS failure (failure to bind because QoS requirements cannot be met). To be able to decide if interfaces can be bound, we need to be able to specify QoS annotations on interfaces to be bound and (possibly) the objects.

QoS monitors need also to be visible as computational objects that are attached to bindings.

One issue is where QoS is specified. The TINA QoS Framework [TINA] supports QoS annotations at: (1) the object, (2) the operational interface and (3) the stream interface. TINA ODL (which is an extension of OMG IDL for describing computational objects) provides a service attribute construct to capture QoS constraints. [Blair94a] proposes QoS annotation at interfaces that states what is required and what can be provided from the interface.

2 .4 Engineering Viewpoint

QoS mechanisms are selected according to user supplied QoS specifications, availability of resources and resource management policy. We distinguish between static QoS mechanisms which is related to establishment of bindings (QoS provision) and dynamic QoS mechanisms which are related to the actual communication phase (QoS control and management).

[Campbell96] has identified three main types of QoS mechanisms:

- QoS provision mechanisms, which can involve QoS mapping, admission testing and resource reservation.
- QoS control mechanisms, which do real-time traffic control to maintain agreed QoS. This include flow-shaping, flow-scheduling, flow-policing, flow-control and flow synchronisation.
- QoS management mechanisms, which could be required to ensure that agreed QoS is sustained or renegotiated. This could involve QoS monitoring, maintenance, degradation, signalling and scaling.

2 .4 .1 QoS provision

How is end-to-end QoS provided, based on the specification? This is mainly about engineering support like e.g. mappings, resource-management, protocols and algorithms for doing end-to-end QoS allocation, typically combined with some negotiation or admission testing.

Many specific mechanisms exist and are being investigated. More generic engineering models of QoS provision are being proposed within the ANSA project [LiOtway94] and the TINA framework. [Nicolaou90] investigates QoS reservation algorithms that incorporates multiple layers and mapping between them. [Blair94a] propose that QoS monitor objects are reactive objects. For more information, see section 3.

An issue is how the QoS is communicated with the application program. One could use an API or a special programming language. There is also an option of using a negotiation-protocol which match the QoS requirements with the QoS the underlying system can offer.

2 .4 .2 Dealing with changing conditions

Resource availability and network quality can change dynamically and users may have varying requirements. First we need a way to discover that a significant change has happened. This can e.g. be done by measuring the traffic to deduce the actual QoS provided (or the usage patterns) like in [Zinky96].

If violations to the agreed QoS is detected, some action has to be taken. First one could try to adjust the resources such that the required QoS can be maintained. If it cannot, some kind of renegotiation and/or scaling is necessary. Renegotiation means that the parties agree on a new QoS, i.e. if the new QoS is acceptable for the application-components, it should continue using the new (degraded) QoS. Scaling can be done by inserting filters in the flow path or by letting application components (clients, servers) adapt to the lower QoS levels or resource availability. The use of filters for scaling (see e.g. [Yeadon96]) can be especially useful in multicast groups where multiple receivers of a flow has different QoS requirements.

2 .5 Technology viewpoint

The issue here is what technology exist that support QoS? Technology could be industry standards or products. Here we should look after

- QoS support or support for flexible resource management in operating systems.
- QoS support in ATM switches, FDDI networks or other network technologies.

- QoS support in ORB's / RT-ORB's: [Schmidt97] describes the design of a real-time ORB with (limited) QoS support and has submitted the results as a RFI response to the OMG SIG on Real-Time CORBA. Important parts of this effort is the development of a Real-Time Inter-ORB Protocol (RIOP) and the architecture for a Real-Time Object-Adapter. The ANSA project has produced the RT-ANSAware and the DIMMA platform, which present some prototype technology for QoS and resource management.

2 .6 Summary

In this section we have investigated some important concepts and issues in each of the five RM-ODP viewpoints. QoS is about non-functional properties in general, but research has been centred around communication protocols and multimedia. QoS is an aspect of bindings, but it depends on support from object and interface implementations, to be instantiated.

In the information viewpoint the main issue is QoS specification. Research effort may be divided into (1) how one provide manageable abstractions for application programmers and end-users, (2) how to control QoS of specific technologies and (3) formal notations. We have focused at declarative QoS parameters. The computational viewpoint is concerned with what main components and interactions are necessary to establish and manage QoS. Here, the recognition that bindings should be explicit computational objects is important. The engineering viewpoint is concerned with how end-to-end QoS is provided. We need to consider mechanisms for provision, control and management of QoS. There may be a specific need to deal with dynamically changing conditions, by using renegotiation or scaling (through filtering or adaptation).

Technology for supporting QoS could be industry standards or products. We should look after (1) operating systems, (2) network technologies and (3) Middleware technology (ORB's).

3 MIDDLEWARE QOS RESEARCH

Most of the QoS research to date has been done in the context of individual architectural layers. We may divide this research into the following layers:

- Network (much effort in QoS provision from ATM networks).
- Transport.
- Operating system.

Until recently little work has been done in QoS support for middleware which we can consider as a fourth layer. The research mentioned here is all about distributed system platforms[2].

QoS specifications at the middleware layer should be user-oriented and declarative rather than system-oriented and imperative. Here QoS is also fundamentally an end-to-end issue. This is because we are interested in what is offered to the applications (clients and servers) by the middleware. Users should be specifying what they require, not how this is achieved. The middleware should offer abstractions that lets applications control the relevant QoS parameters and hide irrelevant details.

3 .1 IMAC - Multimedia extensions to ANSA

There has been some effort in developing experimental QoS-driven middleware. Early work has been done at Cambridge University in extending the ANSAware platform with support for multimedia. Another experiment was done at Lancaster University [Coulson92].

The IMAC architecture [Nicolaou91] is based on the ANSA architecture and its prototype implementation, based on the ANSA Testbench. It introduces new concepts and architectural services to deal with multimedia and QoS, e.g. streams and stream types.

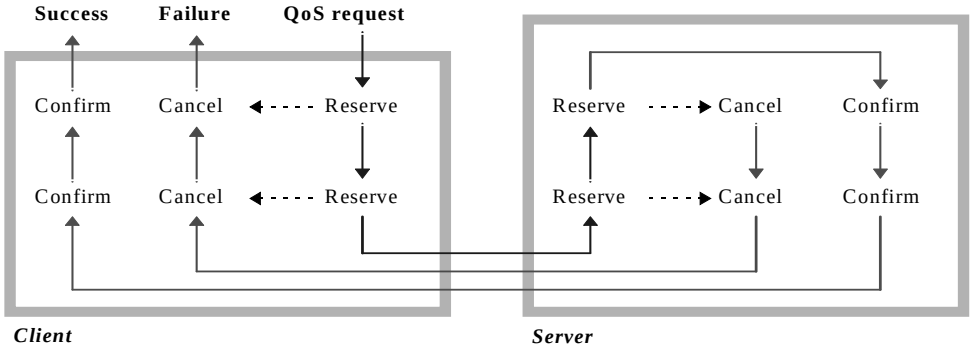
IMAC allows specification of communication-oriented QoS on a per-invocation basis, i.e. interface operations may specify a set of QoS options with which they are prepared to be invoked. Those QoS options put constraints on the underlying communication system. The communication system is viewed as a QoS provider (or server) and clients specify the services

[2] However it may be relevant to also look at multimedia database research, where QoS in some cases are addressed.

and the resources they require as constraints on the set of available services and resources. QoS negotiation is the process of matching a particular QoS constraint to the available set of QoS offers from underlying services and reserve resources for the chosen options.

The approach to QoS specification is to divide the world into QoS domains, divide each domain into QoS-layers and provide mappings between layers. The concept of QoS negotiation is extended to be per layer in this architecture. In this way IMAC provide method of mapping from declarative application-oriented QoS to the more communication oriented QoS options.

IMAC takes an out-of-band approach to end-to-end QoS. This means that the QoS options selected by a client is communicated to the server in a separate invocation. A bind-interface is provided by each server capsule. This, used with the per-layer negotiation, realise end-to-end QoS. The figure below illustrates the algorithm: As a response to a QoS request, the systems try to reserve QoS at each layer at both sides. If this fails at one layer, every reservation which has been done has to be cancelled before returning a failure indication to the user. If reservation succeeds at all layers at both sides, it is being confirmed (committed to) at each layer before a success is indicated to the user.



3 .2 RM-ODP related research

Researchers at CNET and Lancaster University has done significant work in providing support for multimedia and QoS in the context of RM-ODP. They propose an extended computational model for multimedia [Blair94a, Blair94b], which partly has been adopted by the ODP community. They also investigate engineering support with focus at the operating system level.

Support for continuos media is provided by adding the concepts of stream interfaces and stream bindings to the model. Support for real-time synchronisation is provided by adding the concepts of signals and reactive objects. To support QoS management the following are added to the model:

- The ability to specify QoS annotations on interfaces.
- The idea of explicit binding.

- The idea that bindings are objects which provide interfaces that allows one to control the behaviour (QoS) of the binding.
- The use of reactive objects for dynamic management of QoS.

3 .2 .1 QoS annotation language

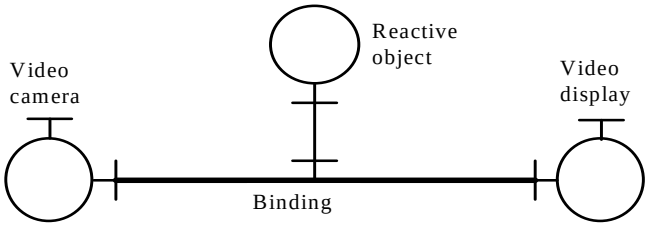
QoS annotations consist of (1) a required clause describing QoS requirements, i.e. the QoS the interface expects from its environment and (2) a provided clause describing the QoS the interface offer to its clients. To allow two interfaces to be bound, they have to be QoS compatible (requirements is met by the provision).

To provide a language for QoS annotation, a real-time logic, QL is proposed [Stefani93]. In essence, QL annotations are specified as temporal relations between signals (emission or arrival of signals at interfaces).

3 .2 .2 Reactive objects

Reactive objects are objects which guarantee real-time behaviour in terms of their reaction to incoming events by generating outgoing events. They are needed in real-time synchronisation and QoS control to provide predictability with respect to time.

In addition to real-time synchronisation, reactive objects can be used as Quality of Service monitors. They should react to QoS violations and initiate re-negotiation of the QoS currently provided. It should be possible to generate such QoS monitor automatically from QoS annotations in QL. The figure below illustrates how a QoS monitor object could be attached to a binding to monitor and manage it.



To provide temporal predictability, reactive objects should conform to the synchrony hypothesis [Stefani92] which states that a reaction to external events takes no time. In practice this means that reactions should happen within bounded time-intervals and that they should be atomic. This allows formal reasoning about time and it allows synchronisation specifications to be written in a time independent manner. To program reactive objects, one may use synchronous languages like Esterel [Berry88] (that rely on the synchrony hypothesis).

3 .3 ANSA research

Significant work has been done in extending the ANSA/ODP computational and engineering model with support for explicit binding, QoS specification and provision [LiOtway94]. This to support real-time applications and has been (partially) implemented in ANSAware-RT and in the DIMMA prototype platform. Currently, DIMMA supports QoS provision at the engineering level.

3 .3 .1 Binding framework

The ANSA team first propose to add explicit binding to the ANSA computational model, as the implicit binding model isn't good enough when interaction between objects become complicated and has QoS requirements. This will allow specification and management of QoS for bindings.

A binding framework is proposed which define the following components in the engineering viewpoint:

- Binders which creates bindings by using resource managers. Binders are regarded as transparency mechanisms and has the role of mapping QoS requirements to a lower level that could be used by resource managers. Binders may build on other binders to provide higher level QoS in a hierarchical manner.
- Binding managers, which is user programs that manipulate binders.
- Resource managers, each which provide QoS for the particular resource it manages.

3 .3 .2 QoS Framework

A framework for expression and management of QoS is proposed. It consists of the following components:

- QoS domains associated with specific areas of application or service. A QoS domain defines a specific set of QoS parameters (dimensions) and their combinations. Domains can be constructed using other domains.
- QoS management which is a set of tasks, associated to each QoS domain (e.g. QoS provision or negotiation). Tasks can be generic or domain-specific.
- Basic QoS mechanisms i.e. engineering support, associated to each QoS domain. Each domain have a set of basic mechanisms which can be different for different QoS domains.
- A common language for QoS expression. Either a language or an API based approach could be chosen for QoS expression. An API based approach implies a set of programming interfaces for each QoS domain. A language based is preferred because it provide a high level of abstraction, it is flexible and it allows the application of automation tools.

Languages are proposed for defining QoS domains (based on IDL) and for specifying QoS requirements when doing explicit binding (based on the trader constraint language).

In this framework, QoS can be associated to bindings, interfaces, objects, object/interface templates and activities. QoS associated to non-bindings are regarded as QoS templates which are activated when a particular binding is established.

3 .4 CORBA Objects and adaptability

The QuO project [Zinky97] is motivated by middleware's (e.g. CORBA) lack of support for handling environmental variables like Quality of Service. An observation is that distributed applications tend to be fragile when environment and usage patterns change (e.g. when migrating an application from LAN to WAN). The project is driven by a demand for middleware that can deal with QoS requirements, changing usage patterns and underlying resources and thereby support adaptability.

An architecture is being developed, QuO (QoS for CORBA objects) to support QoS at the CORBA layer. It extends the functional interface definition language (IDL) with a QoS description language (QDL). This capture application's expected usage patterns and QoS requirements for clients connections to objects.

The project aim to solve the problems in four complementary ways. (1) by defining concepts for integrating knowledge of system properties and QoS, (2) by masking variance in system properties, (3) by making relevant object design decisions visible and (4) by providing a framework to reduce programming effort and support reuse.

3 .4 .1 QoS Specification

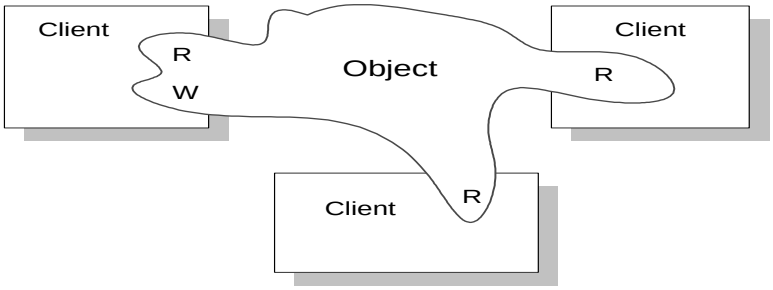
System properties are made first class entities. In this conceptual framework connections (bindings) may be established between clients and objects. Each connection is associated to a contract which capture expected usage patterns and QoS requirements. The concept of QoS regions helps simplify the problem of dealing with a n-dimensional QoS space. For a given functional interface there can be multiple behaviours, each bound to a QoS region.

3 .4 .1 .1 Connections and QoS

To provide end-to-end QoS, system information from both the server, the communications infrastructure and the client must be reconciled. There are three places to do this: in the client, the communications infrastructure or the object. Unlike most others the QuO project does it in the object.

In the QuO approach part of the object's implementation is moved into clients address space. At the client side, a stack of layered delegate objects

represent the connection with QoS. The connection boundary where the expected usage patterns and QoS requirements is agreed upon, is then located in the clients address space. Advantages with this is that there is virtually no delay between client and the connection object and that a delegate object will never fail independently of the client. Distribution and QoS management may then regarded as encapsulated in the object itself. The figure below illustrates how those "smart proxies" distribute object location and state.



3 .4 .1 .2 QoS regions

To handle divergence between expected and provided system conditions, the QuO approach allow specification of two levels of system conditions:

- Negotiated regions which reflect the expectations of QoS and usage patterns.
- Reality regions which reflect the actual client usage/object QoS, measured. For one negotiated region there may be many reality regions.

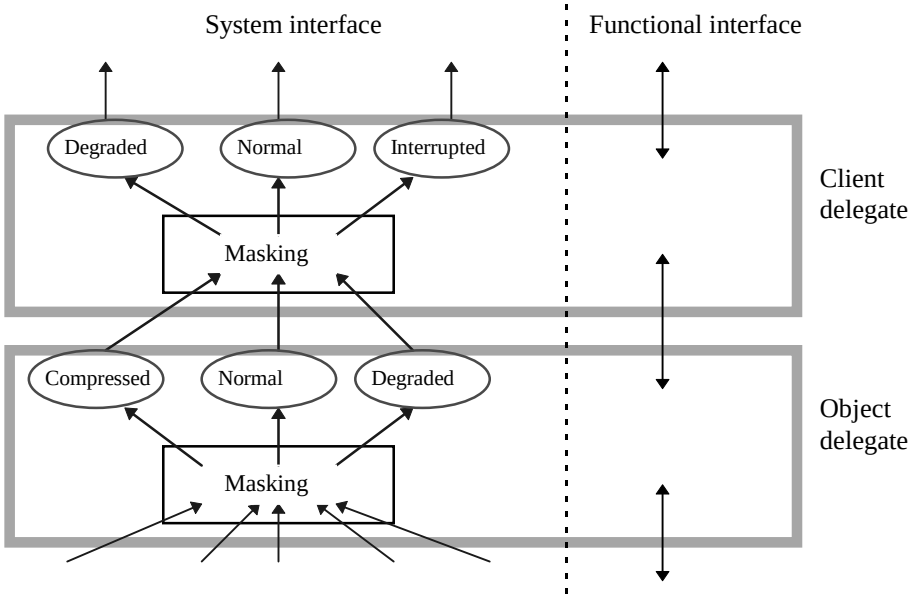
A region is defined by a set of quantitative QoS parameters with associated values or ranges of values. It is then possible to tell if the system is in a specific region by comparing region-definitions with actual measurements. Handlers may be called when transitions between regions occur, to inform the client or connection object so that adaptation or renegotiation can take place.

3 .4 .2 Adaptation and Transparency

Variance in system properties are masked by a layered stack of delegates at the client side. Each layer exports a negotiated region to the layer above. Each layer make use of various techniques to mask changing conditions and maintain QoS.

When a layer cannot maintain the QoS corresponding to the current negotiated region it has to propagate information about this upwards. Now each party can try to adapt or if clients expectations cannot be met anymore renegotiation must be triggered.

Objects may use knowledge of its different implementations to adapt. For example a delegate layer can use compression to mask lower bandwidth from the layer below. The figure below illustrates this idea.



3 .4 .3 Making key decisions explicit

The QuO approach has adopted an Open Implementation Approach [Kiczgales91] to allow object designers to expose key design decisions that affect Quality of service[3]. This makes it possible to alter the (non-functional) behaviour of the application by choosing the object implementation which is best suited for the situation. Thus the QuO approach offers a framework to exploit existing solutions to the problem of adaptability.

The meta-data which describe implementations are specified separately from the functional interfaces. A language, QDL (QoS description language) is provided to support specification of metadata. The QDL consist of three sub-languages:

- A Contract Description Language (CDL), which defines the expected usage patterns and QoS requirements for a connection to an object (QoS regions)
- A Resource Description Language (RDL), which abstracts the physical resources used by the object.
- A Structure Description Language (SDL), which defines the internal structure of an object and how it consumes resources.

[3] The Open Implementation Approach also suggest the use of reflective programming, i.e. non-functional aspects (selection of implementations/policies) are controlled trough the meta-object interface. Future research in the QuO project may investigate reflection.

3 .5 Summary

When summarising the middleware oriented research, there are two main questions we aim to answer: (1) How is QoS specification addressed and (2) how is QoS provision and management addressed?

3 .5 .1 Declarative QoS specifications

Both the IMAC and ANSA models seems to mix the specification of QoS and the choice of policy for providing it. For example the choice of protocol, encoding or scheduling policy can be a part of QoS specifications. The IMAC and ANSA models offers application-oriented end-to-end QoS specification by use of negotiation and mapping to lower level QoS. The ANSA model doesn't address negotiation or mapping mechanisms, but focus on a model for defining QoS domains, which IMAC does not go into deeply.

The CNET/Lancaster research has a different approach to QoS specification. A real-time logic (QL) for bounding temporal relations is proposed. This is more general approach to specification which allows formal reasoning, but is restricted to temporal relations. This is also a completely quantitative approach.

In the QuO architecture the object is responsible for mapping application-oriented QoS to a lower level QoS, through layered delegate object structure. Application programs can see QoS as region names, which could be regarded as simple and qualitative QoS-references. The definition of regions which is done separately from application programs and functional interface-definitions, in a special language (QDL), is quantitative and offer full flexibility. In this way the QoO architecture makes a clean distinction between qualitative and quantitative QoS-specifications which corresponds to choice of region and definition of region respectively.

3 .5 .2 QoS provision and management

In IMAC QoS provision is done by mapping and negotiation which does QoS reservations at different layers at client and server side. Conformance tests are done by matching requirements with QoS-offers. Layer specific QoS reservation is mapped to resource reservation and QoS requirements to underlying layers. There is no support for dynamic changes.

The ANSA model include explicit binding, where QoS-requirements are specified in invocations to bind-operations. Each QoS domain should define, a binding-interface. During binding, a conformance test is carried out. Resource allocation and resource management is supported by a hierarchy of binder-objects and resource-managers.

In the CNET/Lancaster approach, there is support for explicit binding and conformance test by matching requirements with offers. Reactive monitors can be generated from QL. They could be used to trigger renegotiation if necessary.

The QuO architecture supports generation of QoS monitors and adapter-code from QDL specifications. Two types of QoS-regions are supported: negotiated and reality (measured). Change in reality-regions can be met by object-adaptation and made transparent for the application programs by the layered delegate object-structure. A change in negotiated region means renegotiation and application adaptation.

3 .5 .3 Implications

The different approaches described here are not mutually exclusive when constructing practical middleware solutions. To some degree, they focus at different problems, and therefore, some aspects of these may be usefully combined. Options for future research may be to investigate how those models and techniques may be combined in practical solutions. When focusing at how applications simply and cleanly can control QoS of interactions, the Open Implementation Approach seems promising because of its clear separation of concern. It remains to see how well this can be combined with application oriented declarative QoS specification.

4 QOS ARCHITECTURES

According to [Aurrecoechea96] most developments in QoS support has been done in the context of individual architectural layers. Considerable progress has been made in the separate areas of RM-ODP, end-systems support and network support, but much less progress has been made in addressing an overall QoS architecture for multimedia communications. The current state of QoS research suffer from:

- Incompleteness
- Lack of mechanisms to support QoS guarantees
- Lack of an overall framework

To support end-to-end QoS guarantees, work on QoS driven end-system architecture needs to be integrated with network configurable QoS services and protocols. Some researchers try to address these problems by proposing architectural approaches to QoS. The intention of a QoS architecture is (according to [Aurrecoechea96]):

> *"...to define a set of quality of service configurable interfaces that formalize quality of service in the end-system and network, providing a framework for the integration of quality of service control and management mechanisms."*

It is hoped that this approach can help avoid duplication of functions across layers and maximise efficient QoS management.

4 .1 Overview

Comprehensive surveys on QoS architecture research can be found in [Aurrecoechea96] and [Campbell96] Here, we present a short overview over research going that direction. Some of them are based on ATM networks [Lazar94, Ferrari96, Besse94]. Others are based on OSI-RM [ISO95], the Internet [Braden94] or RM-ODP (TINA).

- The HeiProject at IBM's European Networking Centre in Heidelberg has developed a comprehensive QoS model [Volg96]. This model has been designed to handle heterogeneous QoS requirements from different receivers in a multicast group and to support QoS adaptivity via flow-filtering and media scaling.

- The University of Pennsylvania has developed the OMEGA architecture [Nahrstedt95a]. The essence of this architecture is resource reservation and management of end-to-end resources. A QoS brokerage model [Nahrstedt95b] is developed to facilitate the resource management process. This incorporate QoS translation, QoS negotiation and renegotiation. Resource guarantees are negotiated during connection establishment by the QoS broker protocol which is present in both the application and transport subsystems.
- The Integrated Services (int-serv) Group of IETF has defined a comprehensive architecture [Braden94] and a QoS framework used to specify the functionality of Internet system elements which make multiple, dynamically selectable QoS available to applications.
- The Quality of Service Architecture (QoS-A) [Campbell96] is developed at the University of Lancaster. It is a layered architecture of services and mechanisms for quality of service management and control of continous media flows in multiservice networks. Orthogonally to layers the architecture is divided into three planes: The protocol plane, the QoS maintenance plane (which contains a number of layer specific QoS managers), and the flow management plane.
- The COMET group at Colombia University is developing and Extended Integrated Reference Model (XRM) as a modelling framework for control and management of multimedia telecommunications networks [Lazar94].
- The Tenet architecture developed at the University of California at Berkeley [Ferrari96]. It includes a family of protocols which run over an ATM network. The Tenet architecture also support QoS mapping between layers and to a form needed by resource reservation protocols. Media scaling and multicast flows with heterogeneous QoS requirements are also addressed.
- The TINA QoS framework [TINA] is developed for specifying QoS aspects of distributed telecommunications applications within the TINA Computing architecture. In the computational viewpoint QoS parameters are stated declaratively as service attributes. In the engineering viewpoint QoS mechanisms employed by resource managers are considered. Computational bindings are mapped to channels that can be constructed from stubs, binders and protocol-adapters.
- The MASI end-to-end model [Besse94] developed by the CESAME Project at Laboratiure MASI, Universite Pierre et Marie Curie. This includes an architecture for multimedia communications which takes end-to-end QoS support as its primary objective.

- The End System QoS framework [Gopal94] developed at Washington University for providing QoS guarantees within the end-system for networked multimedia applications. The main components are: QoS specification, QoS mapping, QoS enforcement and protocol implementation.
- The OSI QoS framework which is developed by the WG21 [ISO95]. This framework broadly defines terminology and concepts for QoS and provides a model which identifies objects of interest to QoS in open system standards.

5 CONCLUDING REMARKS

In this report we have investigated what are the main concepts and issues in Quality of Service support, focusing at open distributed computing and middleware platforms and how these issues are being addressed in research on QoS support in middleware. QoS is about non-functional properties in general, but research has been centred around communication protocols and multimedia. QoS is an aspect of interaction, and is related to the concept of bindings, but it may also depend on support from object and interface implementations to be instantiated.

In the information viewpoint an important issue is how one provide manageable abstractions for application programmers and end-users. QoS specification should be declarative. Approaches to a complex and heterogeneous parameter space may be standardisation through domains or classes, constructs like regions, layering and mapping. In the computational viewpoint the recognition that bindings should be explicit computational objects is important. In the engineering viewpoint we need to consider mechanisms for provision, control and management of QoS. There may be a specific need to deal with dynamically changing conditions, by using renegotiation or scaling (through filtering or adaptation).

We have looked at some important research in QoS in middleware platforms. This is centred around ODP, ANSA and CORBA architectures. We can conclude that there is a need of support for controlling non-functional aspects as well as functional aspects in middleware. Promising progress is being made in language support for QoS specification as well as engineering support for resource management.

Separation of concern is important. First the separation of functional aspects from non-functional aspects, which cannot always be hidden behind abstractions. Second the separation of declarative QoS requirements from the policies and mechanisms that realise the required QoS. The Open Implementation approach and the principle of reflective programming seems to be promising in exposing implementation issues, separate from the functional interface.

The option of exploiting reflection to tailor implementations to meet Quality of Service requirements therefore seems to be a case for further research, especially how this may be combined with simple declarative QoS specification. More may be done in investigating how aspects of the different models and techniques described here may be combined in practical middleware solutions.

REFERENCES

[Aurrecoechea96]
C. Aurrecoechea, A.T. Campbell, L. Hauw, “A Survey of QoS Architectures” Multimedia Systems Journal, special issue on QoS architecture, 1996.

[Berry88]
Berry, G. Gonthier, “The ESTEREL Synchronous Programming Language: Design, Semantics, Implementation”, INRIA Report No. 842, 1988.

[Besse94]
Besse, L. Dairaine, L. Fedaoui, W. Tawbi, K. Thai, “Towards an Architecture for Distributed Multimedia Application Support”, Proc. International Conference on Multimedia Computing and Systems, Boston May 1994.

[Blair94a]
G.S. Blair, M. Papathomas, G. Coulson, P. Robin, J.B. Stefani, F. Horn, L. Hazard, “Supporting Real-Time Multimedia Behaviour in Open Distributed Systems: An Approach Based on Syncronous Languages”, Proc. ACM Multimedia 1994, San Fransisco, 1994.

[Blair94b]
G.S. Blair, G. Coulson, M. Papathomas, P. Robin, J.B. Stefani, F. Horn, L. Hazard, “A Hybrid Approach to Real-time Synchronisation in Distributed Multimedia Systems”, Lancaster University Report MPG-94-21, 1994.

[Braden94]
Braden, D. Clark, S. Schenker, “Integrated Services in the Internet Architecture: An Overview”, RFC-1633, 1994.

[Campbell96]
A.T. Campbell, “A Quality of Service Architecture”, Ph.D. Thesis, Computing Department, Lancaster University, 1996.

[Coulson92]
G. Coulson, G. S. Blair, N. Davies, N. Williams, “Extensions to ANSA for Multimedia Computing”, Computer Networks and ISDN Systems (25), pp. 305-323, 1992.

[Eliassen95]
F. Eliassen, J.R. Nicol, “Polymorphic Typing for Continuous Media Flows and its Application to QoS Brokerage”, GTE Labs Technical Report, TR-303-07-95-380, July 1995

[Engler95]
D.R. Engler, M.F. Kaashoek, “Exterminate All Operating System Abstractions”, Proc. 5th Workshop on Hot topics in Operating Systems (HOTOS-V), IEEE Press, May 1995.

[Ferrari96]
D. Ferrari, “The Tenet Experience and the Design of Protocols for Integrated Services Internetworks”, Multimedia Systems Journal, 1996.

[Gopal94]
Gopalakrishna, G. Parulkar, “Efficient Quality of Service in Multimedia Computer Operating Systems”, Washington University Report WUCS-TM-94-04, August 1994.

[Hanssen97]
O. Hanssen, “FlexiNet - Extensible Kernel Investigation”, ANSA Phase III draft report APM.2002.00.01, May 1997

[ISO95]
“Quality of Service Framework”, ISO/IEC JTC1/SC21/WG1 N9680, 1995.

[Kiczgales91]
G. Kiczgales, J. J. des Rweres, D. Bobrow, “The art for the metaobject protocol”, MIT Press 1991.

[Lazar94]
A.A. Lazar, S. Bhonsle, K.S. Lim, “A Binding Architecture for Multimedia Networks”, Proc. COST-232 Conf. on Multimedia Transport and Teleservices, Vienna, 1994.

[LiOtway94]
Li, D. Otway, “ANSA Real-Time QoS Extensions”, ANSA draft report APM.1094.00.06.

[Nahrstedt95a]
K. Nahrstedt, J. Smith, “Design, Implementation and Experiences of the OMEGA End-Point Architecture”, University of Pennsylvania Technical Report MS-CIS-95-22, May 1995.

[Nahrstedt95b]
K. Nahrstedt, J. Smith, “The QoS Broker”, IEEE Multimedia, spring 1995.

[Nicolaou91]
C. A. Nicolaou, “A Distributed Architecture for Multimedia Communication Systems”, Ph.D. thesis, Computer Laboratory, University of Cambridge, 1991.

[Plagemann95]
T. Plagemann, A.S. Saetre, V. Goebel, "Application Requirements and QoS negotiation in Multimedia Systems", proc. 2nd Workshop on Protocols for Multimedia Systems, Salzburg, October 1995.

[RM-ODP.1]
ISO/IEC JTC1/SC21/WG7, "Basic reference model of Open Distributed Processing – Part 1: Overview", ISO/IEC DIS 10746-1.

[Schmidt97]
D.C. Schmidt, A. Gokhale, T.H. Harrison, D. Levine, C. Cleeland, "TAO: a High-Performance Endsystem Architecture for Real-time CORBA", RFI response to the OMG Special Interest Group on Real-time CORBA. 1997.

[Stefani92]
J.B. Stefani, L. Hazard, F. Horn, "Computational Model for Distributed Multimedia Applications Based on a Synchronous Programming Language", Computer Communications, pp. 114-128, Vol. 15, No. 2, March 1992.

[Stefani93]
J.B. Stefani, "Computational Aspects of QoS in an Object-Based Distributed Architecture", 3rd International Workshop on Responsive Computer Systems, Lincoln, USA, Sept. 1993.

[TINA]
TINA-C, "The QoS Framework", internal report

[Volg96]
C. Volg, L. Wolf, R. Herrtwich, H. Wittig, "HeiRAT - Quality of Service Management for Distributed Multimedia Systems", Multimedia Systems Journal, 1996.

[Yeadon96]
N. Yeadon, A. Mauthe, F. Garcia, D. Hutchison, "QoS Filters: Addressing the Heterogeneity Gap", proc. Interactive Multimedia Systems and Services (IDMS'96), Berlin, March 1996.

[Zinky97]
J.A. Zinky, D.E. Bakken, R.E. Schantz, "Architectural Support for Quality of Service for CORBA Objects", Theory and Practice of Object Systems (Special Issue on the OMG and CORBA), january 1997.