

Chapter 6

Towards an Advanced Quality of Service Architecture

Jan deMeer, Armin Eberlein

6.1 Introduction

This chapter addresses questions such as “What a QoS-aware architecture must look like?” or, “What building blocks does a QoS-aware architecture consist of?” in the context of interactive multimedia applications (IMM). As an integrated part of QoS architectures, capabilities must exist which support management and control of functionality and performance of IMMs. For an architecture that is said to be QoS-aware, the IMM service quality provided must be controlled and if necessary, be adapted to previously agreed targets. QoS-awareness operations are to be placed in locations of interest of the considered architectural model that allow access to interfaces, provide negotiation, adaptation and tuning capabilities etc. For the design of QoS architectures there are two basic classes of requirements. The first class of requirements is derived from applications that benefit most from QoS architectures. Interactive multimedia services are the main representatives of this category of applications. The other class of requirements deals with constraints on how to construct a framework of QoS that provides the building blocks, i.e. the concepts, operations and tools, of QoS architectures. Consequently this chapter discusses at first, the properties and features of IMM services which must be taken into account

during the design of an architecture that is QoS-aware, as well as the middleware of a distributed system. Next, architectural building blocks are introduced and their technical aspects with respect to QoS are discussed. Then, recent architectural approaches are presented that claim to be QoS-aware. However, in order to find out whether an architecture is suited for QoS management and control, one needs a kind of conformity criteria. It is a definitive goal of this chapter to present a set of criteria that allows evaluation of architectures with respect to the derived set of QoS conformity criteria. Hence, QoS-conformity comprises architectural concepts and features used to control and manage the quality of interactive multimedia services for the whole time the application is active. Whereas the latter set of criteria addresses *application sensitivity*, the former set addresses *QoS architectural constructivity*.

Section 6.2 discusses IMM requirements representing application sensitivity. Section 6.3 deals with various views of QoS, as provided by the major standardisation bodies like ISO, ITU-T, ETSI and IETF. Section 6.4 contains the descriptions of the basic functions and mechanisms that are needed during the evolutionary phases of a QoS-controlled service. They comprise prediction, resource reservation, negotiation, monitoring, tuning and termination mechanisms. In order to evaluate the presented QoS architectures a new *QoS reference architecture* is introduced in Section 6.5, which identifies structures to deal with forward and backward controlling and which identifies locations or interfaces in the architecture at which QoS-related activities, like observations, tuning or resource actuating are to be done.

The architectural approaches presented in Section 6.6 include an ODP-conformant approach, the CORB-Architecture of the OMG, the Telecommunication Architecture TINA, the QoS-A approach from Lancaster University (U.K.), the HeiProjects of the IBM-Labs in Heidelberg (Germany), the Tenet architecture of the University of California in Berkeley and the Omega architecture of the University of Pennsylvania. The presented architectures are then evaluated against the collected criteria of QoS application-sensitivity and constructivity. From this evaluation an initial set of recommendations of QoS-conformity is derived. The concluding section discusses the benefits and drawbacks of the specified QoS conformity evaluation criteria with respect to applications and architectural design.

6.2 Quality of Service Requirements of IMM services

Interactive multimedia services have not been in existence for very long. However, they have become of great importance in the last few years. Applications, such as video conferencing, video on demand (VoD), and interactive games[12] have been paid a lot of attention due to their potential to raise revenues. However, large-scale deployment of such services will impose very high constraints on virtually all system components, especially the network devices and end-systems. One of the key features of IMM is the vast amount of data that needs to be transmitted at high speed in order to provide a high-quality multimedia service. Depending on the quality of

the video and the compression scheme, a single movie can require 25 GBytes of storage[6]. Very high transmission rates are required to deliver this data in time. Again depending on the quality and the compression scheme, transmission rates of 135 Mbps and more can be necessary. In fact, one of the hardest requirements of IMM services is the in-time delivery of data. A frame that arrives after its deadline is worthless and can drastically reduce the quality perceived by the user. Furthermore, IMM data of many applications has to be delivered as a continuous stream. If too much data is delivered, data will be lost due to buffer overflow; if too little data arrives at the receiver's side, the receiver will experience buffer starvation. It is therefore crucial to look for means that can guarantee a certain quality of service for multimedia services.

IMM applications have to meet certain properties, which have a great influence on the quality of interactive multimedia services:

1. Continuity, because streams of data are delivered to IMM applications
2. Capacity, because large amounts of data are transported by streams
3. Timeliness, because of real-time transportation constraints
4. Integrity, because of presentation constraints

The resources used by IMM services, i.e. networks, switches, channel bandwidths or flow control mechanisms of protocols, must be capable of supporting selected QoS characteristics and mechanisms that help maintain the properties listed above. This means, that certain characteristics and mechanisms which provide *continuity* are to be implemented at given system component interfaces in such a way that e.g. enough bandwidth will be made available during the whole transmission time in order to guarantee continuity of a stream of data. *Capacity* is a property that requires either space for buffering or bandwidth for transportation or a combination of both. To achieve *timeliness* resource control mechanisms must be implemented that avoid traffic jams. *Integrity* requires reliable transmission protocols. Loss or duplications of data (i.e. packets of a stream) must be detected and if possible be corrected. Hence, a critical issue of the design of QoS architectures is the deployment of various kinds of mechanisms to observe and to control stream continuity, buffer capacities, transmission delays and integrity of data.

6.3 Views on Quality of Service

The notion *Quality of Service (QoS)* is a widely used term in standardisation and research efforts. QoS originally emerged in communications to describe characteristics of data transmission. Especially in ATM, QoS is an important issue[19]

The definition introduced by the ODP description 11.2.2 of ISO/IEC IS 10746-2[23] states:

"The notion QoS is a system or object property, and consists of a set of quality requirements on the collective behaviour of one or more objects... Note: QoS is concerned with such characteristics as the rate of information transfer, the latency, the probability of a communication being disrupted, the probability of system failure, the probability of storage failure, etc."

The European Telecommunications Standards Institute (ETSI) describes in its book "ETSI and the Information Society"[24] the QoS notion from the end-user's point of view. Methods for the evaluation of service or system usability and human factors are needed to improve the regular process of development.

"It should be possible to evaluate the characteristics of a system or service as to its task performance. This should be done both qualitatively and quantitatively".

Together, the definitions of ISO and ETSI span the scope of QoS, i.e. comprising characteristics and functions of networks and applications.

The OSI Reference Model[23] describes QoS as being mainly related to speed and reliability of transmission, such as throughput, delay, delay variation (jitter), bit error rate (BER), cell loss rate, and connection establishment failure probability etc. An OSI (N)-QoS subsystem is equipped with three auxiliary QoS management functions, i.e. for policies, interaction constraints and QoS parameters, that allow filtering and observation of QoS characteristics of input and output data to and from the OSI (N)-subsystem. Because of these built-in auxiliary functions the exchange of QoS-related information with functional entities in a higher-layer (N)-subsystem is thus possible. For completeness, it shall be noted that the presented QoS extensions to the OSI RM are documented in a separate standard called *Quality of Service Framework* which has been published as ISO International Standard No. 13236[25] in January 1997.

The *QoS Framework in ODP*[27] is an outcome of the standardisation work of recent years and was expected to become an international standard during the year 1999. This standard is more complete than the IS13236 in the sense that it is based on the object-oriented approach of ODP and can thus easily be applied to IMM services. Straightforward notions like flow (stream), QoS contract and formal approaches to specify QoS constraints have been invented by this standard. The framework of QoS for ODP consists of an entity-relationship-like approach that invents a number of QoS concepts and combines them into certain QoS relationships. For example, QoS parameters contain the values of QoS limits, targets, thresholds or signals. The latter QoS values are to be understood as the quantification of the QoS characteristics. QoS parameters on the other hand are exchanged among QoS management agents that perform dedicated tasks such as monitoring, control or administration. Generally, the QoS management of a system is driven by the QoS characteristics that are specified as user requirements or system policies. A QoS characteristic itself represents on the one hand QoS aspects of the system, service or the resources, but on the other hand the actual behaviour of the application.

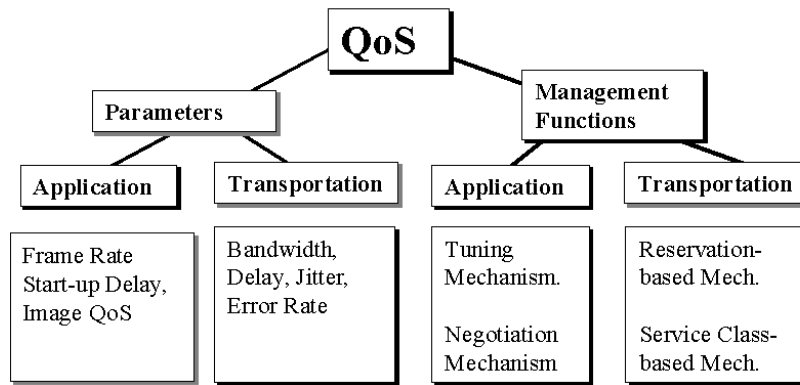


Figure 6.1 Quality of Service framework

Figure 6.1 presents the QoS categories divided into parameters and management functions according to the QoS framework of ODP. Parameters and management functions apply to IMM services as well as to transportation services. The application-related parameters mainly describe presentation characteristics, e.g. image size and resolution, frame rate, start-up delay etc. The transportation parameters describe the most common network characteristics, i.e. bandwidth, delay, jitter and transmission error rate. Compared to the parameters, the QoS management functions are less obvious but more important with respect to the anticipated QoS capabilities. Management functions used by applications differ from those one that are used by transportation protocols. Transportation QoS management is principally not capable to exploit the advantages of feed-back QoS control mechanisms for network-wide communications because transportation resources are normally shared by many applications. If there is no dedicated reservation possible, reservations will be done using statistical mean values. Hence peak rates might not violate QoS contracts, but will still lead to congestion and data loss. Transportation protocols react to congestion by applying forward control mechanisms like dropping of transportation units or by queueing remaining transportation requests. These mechanisms are called forward control mechanisms, because traffic control is only performed after congestion has been detected. Backward control mechanisms however are able to avoid congestions per se. Hence applications can act prior to congestion. This is possible because application resources will not be shared with other applications as it is the case for transportation resources which have to be shared by many users. Thus, on the application level, there is no statistical sharing of resources between several applications. In other words, forward control is performed in direction of data flow, whereas backward control feeds information backwards, i.e. in the opposite direction of a flow of data. As it is outlined in more detail in Section 6.5, the combination of both control principles leads to a two-level QoS control architecture, i.e. the division into an application and a networking layer.

Nevertheless, QoS management functions can be distinguished according to their policies. One policy is to establish control mechanisms along the transportation path or along the connection. Another policy is to establish control per message. All resource reservation mechanisms like those used by ATM or RSVP[9] fall into the first category. Message-based, sometimes also referred to as connection-less control is provided by the so-called Differentiated Services[10]. Messages are uniquely identified during their flow through the network and handled according to the service class to which the message stream belongs. Each unique identifier contained in each message represents a certain service class. This control policy is therefore also referred to as being service class oriented.

Non-standardised but very useful categorisation frameworks of QoS have been provided by Guo[7], Nahrstedt[14,15] and Vogel[20]. Guo[7] distinguishes between application QoS and network QoS parameters. The application QoS parameters specify image quality and size, frame rate and startup delay. The network QoS addresses bandwidth, delay, jitter and error rates. A more comprehensive QoS framework is given by Nahrstedt[15]. It defines five different levels of QoS. The user QoS is the quality of the service as it is perceived by the user. The application QoS is specified in terms of media quality (e.g., data-unit rate and end-to-end delay) and media relations (such as synchronisation). The system QoS describes communication and operating system requirements, such as task processing time, ordered delivery of data, error-recovery mechanisms and scheduling mechanisms. *Network QoS* addresses issues such as latency, bandwidth and jitter. Device QoS parameters specify timing and throughput demands for media data units. Vogel[20] defines Quality of Service as

“..the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application”.

There are five categories of QoS parameters:

- *performance-oriented parameters* are end-to-end delay and bit rate;
- *format-oriented parameters* are video resolution, frame rate, storage format and compression scheme;
- a *synchronisation-oriented QoS parameter* is, for example, the skew between the beginning of audio and video sequences;
- *cost-oriented parameters* are related to connection and data transmission charges and copyright fees;
- *user-oriented parameters* describe the subjective image and sound quality.

It can be said that standardised and non-standardised frameworks mainly concentrate on defining relationships between QoS parameters and on their classification. The dynamic co-operation that exists between network and application components for the control of various qualities of several multimedia services has been less intensively studied by current frameworks. The contribution

of this chapter is thus devoted to identify mechanisms that handle end-to-end QoS constraints and that can be integrated into a unified architecture.

6.4 Quality of Service Management Functions

For the purpose of classifying QoS management functions, the QoS framework of ISO[25] has invented certain stages in the evolution of quality-controlled services. These phases describe the following service evolution activities:

1. A-priori service seizing applies prediction mechanisms of expected QoS characteristics in order to establish a certain QoS context.
2. Preparation of service establishment requires resource reservation mechanisms for predicted QoS contexts.
3. Execution of service establishment requires negotiation mechanisms in order to agree upon QoS requirements.
4. Operation of services requires mechanisms that ‘honour’ previously negotiated QoS agreements.
5. Termination of services requires mechanisms by which all reserved resources, services and connections are freed or reset.

In accordance with this phased approach, the QoS framework of ISO[25] provides guidelines for the specification of QoS. So-called specifiers consist of the unique name of QoS characteristics, their definition of purpose, their quantified parameter values, their derivation rules from non-quantified notions to quantified values, their specialisation rules from generic to specialised, as well as usable QoS and optional specifiers for further information.

This is suitable for users and their applications who need to be able to specify the QoS required to provide an interactive multimedia service with the desired quality. It is essential that the users can specify the QoS requirements using terms they can understand. A user can understand terms like resolution, image size, colour depth, but will find it difficult to specify jitter or cell-loss rate. However, the latter terms are very meaningful at the network level. This means, the user’s QoS specification is declarative in nature, i.e. the specification merely states what is required rather than how this is to be achieved.

6.4.1 Prediction

By determining the actual system load and QoS limits, a QoS context is calculated by which the requested service can most probably be provided without degradation of quality. An example of a typical prediction mechanism is the admission control function. Admission control is responsible for comparing the QoS requirements arising from a new service request with the resources available in the system. The acceptance of a new request depends on the scheduling mechanisms, on the characteristics of the traffic and on the available resources. A set of tests is run to

determine if the new service request can be supported without violating the guarantees given to already existing sessions.

6.4.2 Resource Reservation

Resource reservation and negotiation mechanisms are quite similar. The different notions are possibly due to their different context of application. Resource reservation is a notion used in the networking domain and allows the reservation of all resources that are necessary to achieve a compulsory QoS context (see explanations below). Negotiation on the other hand tries to reach agreement on application QoS characteristics. However, there is no agreement without resource reservation on the application and transportation layers.

Due to the stringent timing-constraints for the delivery of data in multimedia applications, resource reservation is commonly used to provide a compulsory QoS. A resource reservation protocol firstly establishes a path through the network and then reserves resources (e.g., CPU, memory, bandwidth, etc.). This can be done using two approaches. The pessimistic approach reserves resources based on the worst-case traffic, which means that the system is under-utilised, but a *guaranteed QoS* is provided with a very high probability. The optimistic approach allocates resources based on the average characteristics. This leads to high resource utilisation but overload situations still can occur. Hence compulsory services can never provide service guarantees.

6.4.3 Negotiation

Negotiation is a mechanism that interprets the complex semantics of the QoS parameters of the possible different participant's roles in an application. Roles of participants are called initiator, responder and arbiter. The arbiter role is played by the network that interconnects initiator and responder. QoS parameters and characteristics normally have an $m:n$ relationship. Within this relationship negotiation must reach an agreement on, for example, the *highest quality attainable* (HQA) of an operating target with best effort QoS, or the *controlled highest quality* (CHQ) on an upper limit with compulsory QoS, or the *lowest quality acceptable* (LQA) on a lower limit with guaranteed QoS. Since for best effort QoS there is no assurance, no monitoring and no remedy activities, best effort service is only applicable when fixed parameter values of targets and thresholds do not exist. *Compulsory QoS* [25] on the contrary provides monitoring services but is even not able to provide a guaranteed QoS. However, when QoS degrades below an agreed minimum level, the service can be aborted or the messages transmitted can be dropped. In order to guarantee QoS, services are not allowed to be aborted because of QoS violations of other applications. Instead new resources must be added to the service to maintain the guaranteed level of QoS.

In the literature, compulsory or guaranteed services are sometimes also referred to as *predicted* or *deterministic* service respectively. The deterministic service offers hard QoS guarantees to all packets belonging to a stream. The predicted service that

is based on statistical bounds provides weaker QoS guarantees than the deterministic service, but stronger ones than the best-effort service.

Negotiation occurs also between vertical architectural layers within one system. This kind of negotiation is called *QoS mapping*[6]. The lower layers need to support the QoS expected by the higher layers. This kind of negotiation takes place during call setup. The mapping process, sometimes also referred to as QoS translation[7], will take place between several layers in a hierarchy. Each layer will request a certain QoS from its adjacent lower layer. However, it is to be expected that there is not always a one-to-one mapping possible. If a direct mapping is not possible, the appropriate QoS parameters will need to be negotiated. This negotiation between two adjacent layers in one end system is sometimes referred to as layer-to-layer negotiation.

However, for already established calls, re-negotiation of the QoS parameters might become necessary as well. For instance, during a medical tele-consultation, the transmission of high-quality X-ray images requires a better QoS (e.g., higher bandwidth), which can result in network congestion unless initially negotiated QoS parameters of a less stringent application can be reduced after re-negotiation. This requires a system to continuously monitor the actual QoS and to use re-negotiation mechanisms to adapt the QoS parameters or resources to the user demands or to the actual network load. Re-negotiation of the initially agreed QoS context might be initiated by the user or the underlying system. Depending on the QoS context re-negotiation may allow the user to request a higher or lower level of quality. The user can, for example, ask for colour quality while the current session is in 'black & white' mode, or can reduce the QoS requirements in order to save costs.

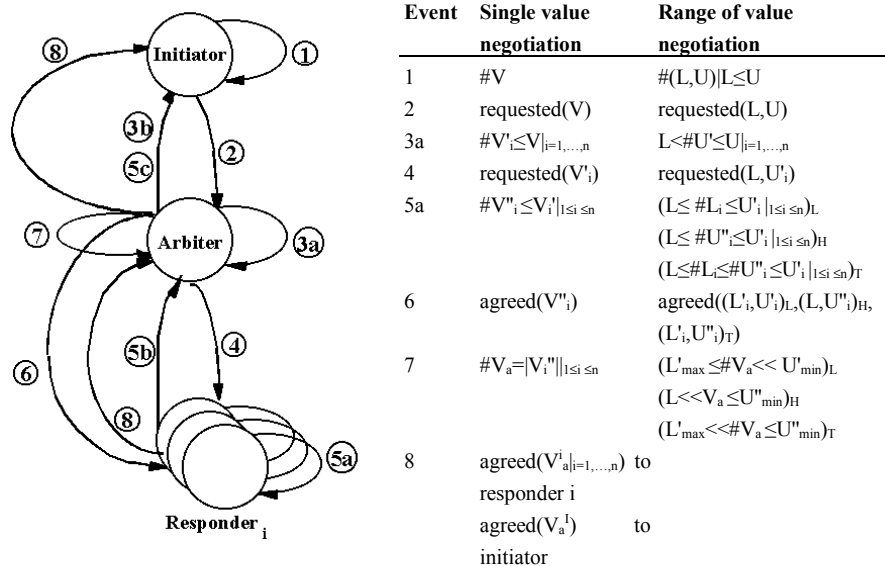


Figure 6.2 1:n-multicasting negotiation

System-initiated re-negotiation is usually caused by a severe lack of resources in cases when automatic QoS adaptation is not sufficient to resolve the problem.

Figure 6.2 presents the basic principles of a negotiation algorithm. It shows a $1:n$ -party negotiation algorithm, which is called multicasting negotiation. It shows the three basic negotiation roles, i.e. *initiator*, *arbiter* and *responder* which are adopted by the parties involved. In a $1:n$ -party negotiation scenario the initiator and arbiter roles are each adopted by one party only. However, the responder role can be adopted up to n times by different players. It is important to realise, that the initiator and responder roles do not determine the sender or the receiver of the data streams for which the players want to negotiate resources or QoS characteristics. The initiator role, for example can be adopted by either the sender or by the receiver. This is not true for the arbiter role. The arbiter role must be adopted by a player which is next to the initiator in the chain of streaming parties, and which is in-between of the initiator and the set of responders. Thus, the arbiter role can be played by the sender in co-operation with its initiator role but also by the streaming channel or network that is between the initiator and the responders. The latter case is assumed to be the normal case.

The initiator role is characterised by taking the initiative for the negotiation. The initiator of course knows for himself what are the quality and resource it can minimally provide for the service. Hence he is able to provide initial values of quality parameters or initial requests of resources needed. It is possible to negotiate either a single target, or the lower or upper limits of a range. In case the initiator wants to negotiate the lower limit, the upper limit or both at the same time, he selects an appropriate range of quality values with the bounds ($\#U$, $\#L$). This range is then sent to the arbiter player at the channel. If he cannot guarantee the requested lower quality or does not have sufficient resources the player rejects the initiator's request and negotiation is finished. When he accepts, he is allowed to adjust the boundaries according to his own needs. The channel that is assumed to play the arbiter role must at least provide a quality that fulfils the conditions determined by the lower bound of the initiator. However, he might not be able to fulfill the conditions determined by the upper bound. So the channel may reduce the requested range by lowering its upper bound only, i.e. by choosing ($L < \#U' \leq U$). It must be noted that this selection can be done on an individual basis for each participating multicasting party. Hence, there can be up to N selections for the new quality values $\#U'_i$, $i=1, \dots, N$ that are now transmitted to the N waiting responder players. On receiving, each responder in turn checks the modified requested range according to his individual capabilities. If the responder behaves out of the requested range, he has to reject the request. Otherwise he will accept; but he is allowed to modify both bounds. Therefore, each individual responder ($1 < i \leq N$) can suggest a change for the lower bound by choosing ($L \leq \#L'_i \leq U$), and for the upper bound by choosing ($L \leq \#U'_i \leq U$). These selections will individually be returned and collected by the arbiter player. The arbiter player then looks at the N responder returns and chooses the maximum of the lower bounds and the minimum value of the upper bounds. The maximum of the lower bound must obviously be less than any upper bound chosen by the responder players. Finally, the arbiter returns the chosen lower and upper

bounds to the initiator as initially requested and the negotiation is successfully finished.

6.4.4 Monitoring

QoS monitoring is an important activity that is required by many other QoS functions, such as QoS policing and QoS re-negotiation. It presumes that a QoS measuring function is available. Basically, the task of monitoring is, to detect and notify any QoS violations. A QoS violation has occurred when the measured value of a QoS parameter does no longer meet the value that was originally agreed. In this case the QoS management system must be informed.

6.4.5 Tuning

For QoS management to be effective there must be remedy mechanisms. If a user exceeds the initially agreed amount of system resources, e.g. a user sends 3 Mbps although only 2 Mbps were initially agreed, appropriate action must be taken. Possible remedy actions include adaptation of the initially agreed QoS context, or the tuning of the performance of the used transportation paths. Remedy actions are necessary in order to protect other users who keep within their agreed QoS context. The actions taken can vary depending on the available resources. For example, the user's violation can simply be ignored, the user could be charged a fine, or the traffic could be re-shaped corresponding to the agreed QoS.

Otherwise, the QoS parameters that have been agreed upon during the negotiation phase might need to be adjusted due to changes in the environment. It

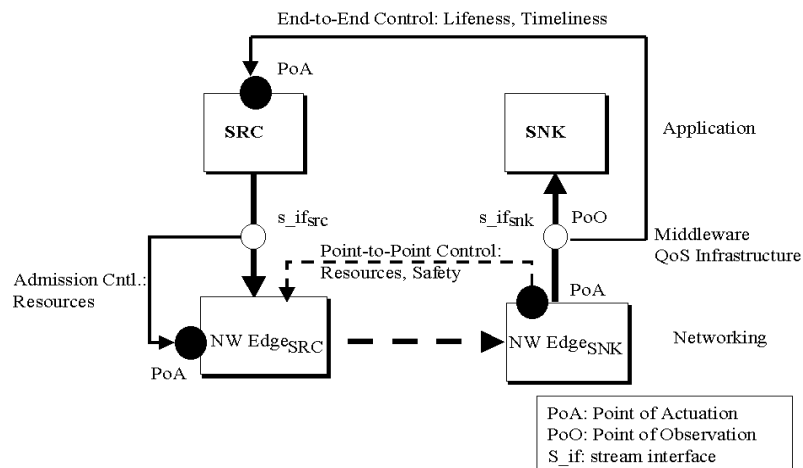


Figure 6.3 QoS management architecture

might not be possible to further maintain the agreed QoS and so, a graceful QoS degradation has become necessary. Lowering the provided QoS in case of unexpected resource shortage is more desirable than the complete abortion of the service.

Tuning is a mechanism that keeps a system variable $y(t)$ between an upper and a lower threshold, e.g. (UTH , LTH). Each threshold is controlled by a separate discriminator. A signal will be generated, e.g. to steer the source of $y(t)$, if either the upper or lower threshold value is violated.

6.4.6 Termination

After the termination of a service, the reserved resources need to be freed again. This means that all system components that were involved in the service provision, have to be notified and asked to free the resources and to update their amount of available resources. This has to be done in a reliable way otherwise resources will be wasted.

6.5 Quality of Service Management Architecture

In this section we develop criteria by which we can identify locations to place active QoS management elements and mechanisms such that they will support QoS system-wide. Basically, we assume that a QoS-aware system architecture is constructed in such a way that a certain set of applications can be run and the performance of the application's service quality can be managed by built-in QoS management functions. We have chosen a three-jar architecture that separates QoS management functions from service providing and service applying functions. Hence, there is the application jar, the networking jar and as a third jar (the middleware) there is the infrastructure of testing, monitoring, tuning, negotiating, terminating etc. in order to do QoS management (see Figure 6.3).

The set of anticipated applications providing guaranteed service qualities that shall be supported by a system architecture that supports QoS, is defined as the class of interactive multimedia (IMM) services. In Section 6.2, the characteristics of IMM have been introduced by a 4-dimensional property space, comprising continuity, capacity, timeliness, and integrity. Continuity requires the support of streams at the application-network interface. High capacities require high bandwidths from the network and high storing capacities from end systems. Timeliness implies short turn-around delays of actions triggered. To provide integrity and safe transmission facilities synchronization mechanisms must be built-in features of the QoS architecture.

The mechanisms and interfaces of the 4-dimensional property space of the IMM service class will dynamically be applied by instances of an application. The mechanisms and interfaces provided by the system architecture must hence dynamically be managed by an additional built-in control policy. The controlling policy employs different mechanisms to deal with safety using point-to-point flow control mechanisms, or, to deal with QoS adaptation using end-to-end tuning

mechanisms, or, to deal with resource management using admission control mechanisms. For example, the Internet provides admission control functions in combination with remedy operations like packet dropping or transmission request queuing in case of overloaded resources.

The architectural principle of flow-control and tuning is feedback and the architectural principle of admission control is feed-forwarding. Both links connect a *Point of Observation* (PoO) with a *Point of Actuation* (PoA) as outlined in Figure 6.3, but with the difference that in the feedback case the PoA steers a source of the controlled flow and in the feedforward case the PoA steers remedy activities at a network's edge. The difference comes from the observation that admission control does not prevent applications from the misuse of resources. Hence, a forward controlling policy must include remedy operations which are applied after the misuse has been observed. In contrast, backward controlling policies operate on specific targets of one or more critical variables observed, such that misuse cannot occur at all. Since these control policies serve for different purposes, a QoS-aware architecture has to support both kinds of control. The control mechanisms are applied at different levels of abstraction, because of their different purposes and of their 'direction of effect'. Direction of effect means a cause-effect relationship. The cause for controlling occurs first at the point of observation and finally triggers an action at the point of actuation. This means, we can identify 'horizontal' and 'vertical' control mechanisms. As it is depicted in Figure 6.3, the admission control mechanism is vertical and the flow control and tuning mechanisms are both horizontal. Admission control is said to take a vertical effect because it is normally located at the edges of networks and is thus not an operational element of a horizontal flow-control mechanism of a protocol. More details can be found in [29].

Any QoS architecture that will be discussed in the following sections provides an individual sub-structure that is transparent to the proposed QoS management architecture as shown in Figure 6.3. Hence, the structure of an architecture is not a matter of evaluation but of their built-in policies and capabilities for managing and controlling QoS. The capabilities and policies provided by the architectures must fit into the above mentioned 4-dimensional property space of IMMs. This means that continuity requires a tight connectivity among the interfaces of various components. Tight connectivity requires that interfaces are able to handle multimedia flows without disruptions. The speed of interconnected interfaces may however synchronously vary over time. The working conditions at the interfaces must be negotiable, i.e. resources will be reserved prior to the start of the service and freed at its termination. The second dimension, capacity, requires access control mechanisms such as admission control or point-to-point flow control mechanisms in order to avoid resource congestion. Mechanisms that are suited to handle the timeliness are mechanisms that permanently keep control of the interaction between end-to-end clients. For instance, if a degradation of speed is observed, the feedback mechanism may assign more bandwidth to the component that is lacking resources. The requirements of the fourth dimension of integrity can be fulfilled by mechanisms that provide safety and liveness qualities. Safety can be achieved by simple

retransmission mechanisms but liveness needs advanced feedback control mechanism.

T. Walter et al.[31] evaluates system architectures from the QoS testing point of view. Since testing is restricted to probing, observing and monitoring activities the structure of the suggested "QoS test architecture" is very simple. Monitors, i.e. testers are attached to the distributed components of the system observed. The behaviour of QoS variables can only be evaluated correctly if the behaviour of the embedding network is predictable during testing. By this philosophy, QoS in a critical state can only be evaluated if the embedding network is forced to perform a predefined behaviour, i.e. a specified test case that instructs the network as well. Certain similarities between a QoS test architecture and a QoS management architecture can be manifested. The tester is a built-in component in the recursive structure of a valid QoS management architecture as suggested in this chapter. The monitor or the tester respectively provides a link between the point of observation at the component to the generating "source" of the behaviour that is expected at the PoO. Thus the testing structure is very similar to the structure of the QoS management architecture with its feedback and feed-forward links with the only difference that the monitor cannot be removed from the system because this would delete the adaptivity property.

Hence, the architecture of a system must be designed such that it is "testable", which means being observable. Furthermore, QoS-aware systems must also be controllable from outside via the points of actuation. In other words, observability and controllability can be expressed in terms of continuous functions. These functions represent flows which behaviours change over time. In QoS-aware

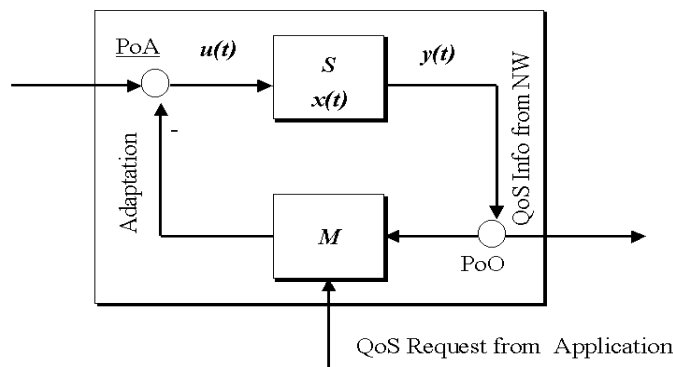


Figure 6.4 Continuous QoS model

systems the internal state is defined by a continuous state function $x(t)$, the egress behaviour, e.g. the egress flow at the edge of a network or a component's interface, is defined by the continuous service function $y(t)$. The tuning capabilities are defined by the steering function $u(t)$ (see Figure 6.4).

A system or a component S is observable, if the service $y(t)$ comprises all internal system states, that are represented by $x(t)$, which means S is testable (observable) by inspecting $y(t)$. S is said to be completely observable, iff the state $x(t)$ can uniquely be determined from $y(t)$ in a finite period of time.

A system or a component S is controllable by monitor M , if any internal state of $x(t)$ can be triggered by a given steering function $u(t)$, which means $x(t)$ is adjustable by modifying $u(t)$. S is said to be completely controllable, if $u(t)$ can translate any state $x(t_0)$ into any other state $x(t_1)$.

Due to design constraints, some of the continuous variables of the system's or component's state $x(t)$ cannot be determined from observing $y(t)$. In that case a so-called observer must be constructed that estimates the state value of a continuous variable from observing $y(t)$. The condition that such an observer can be designed is called the observability of the system[32].

C. Aurrecochea et al.[1] investigates QoS architectures with the aim of summarizing and comparing capabilities and mechanisms of QoS provision, QoS control and QoS management of various approaches in a tabular form. This table provides an interesting overview of the extent to which QoS mechanisms have already been implemented. With respect to the functions listed the most complete architectures are QoS-A, the Heidelberg Architecture and the Tenet Architecture. On the other hand, the authors of [1] have concluded that

“...it is still too early to decide which approach is more suitable for future QoS architectures given the need to support both, high-end (e.g. tele-surgery and time-critical applications) and low-end (e.g. video conferencing and audio tools) multimedia applications.”

The reference architecture suggested in this chapter does not provide a final answer to the question for a suitable QoS-architecture that has initially been raised. However, the answer offered here goes beyond the pure enumeration of QoS mechanisms and parameters. In order to achieve end-to-end control, mechanisms are required that span all components between source and sink of an interaction. This is preferably the case for feedback control mechanisms, because they operate by relating effects directly back to their causes. Normally, causes are observed at the sink, which are several components away from the source. Sources get informed by derived control information from observations. Contrarily, feed-forward mechanisms observe violations from normal behaviour and inform the next node about the violation. There, remedy processes are activated in order to correct observed malfunctions.

Control structures can be adopted by the application and the networking layer in order to provide stream and resource control capabilities over point-to-point and end-to-end distances. Feedback control mechanisms have a so-called tuning effect on the observed streaming behaviour. This effect enables designers to construct

architectures with built-in mechanisms that provide some kind of guaranteed quality of service to users.

6.6 Quality of Service Architectural Approaches

During the last few years a considerable amount of research has been done in the area of quality of service for distributed multimedia services. This research has been conducted by academia as well as by industry. Most of the research has been within the scope of an individual architectural layer, such as the transport or network layer, the operating system, etc. By far less research has been done in the area of comprehensive end-to-end QoS features supporting multimedia applications. The following paragraphs give brief introductions into some of the research projects that address QoS architectures.

6.6.1 Communication Object Request Broker Architecture (CORBA)

The early work of the Object Management Group (OMG) was published in the so-called QoS Green Paper[17]. The basic idea of OMG was to extend the ISO ODP-RM[23] in order to make QoS characteristics and functions suitable for object-based distributed systems.

Both organisations, OMG and ISO, agreed to work on QoS in close collaboration, in order to adapt standards and specifications to new or changed product requirements as quickly as possible. Recent achievements on the issue of QoS can thus most rapidly be adopted by new products. Items of work of the OMG which are related to QoS are considered in answers to OMG's RfP on the Messaging Service[13]. An actual specification has recently been developed within the OMG that covers the three topics of asynchronous method invocations, of transporting requests asynchronously to replies, and of Quality of Service. The specification defines a possible integration of QoS characteristics and functions into CORBA that consists of a set of policies to represent QoS characteristics. Unfortunately, it differs from that one that is used by ISO for defining QoS in the ODP-RM[23].

The OMG model distinguishes three levels of specifying and handling of QoS characteristics. The lowest level corresponds to the *ORB level* at which QoS characteristics for the ORB can be defined by using a default specification option. The next higher level is the *thread level*. Threads describe certain client-server relationships. At this level, it is allowed to override QoS characteristics from the ORB level. The highest level is the *object level*, at which QoS characteristics are processed in the order of priorities. QoS characteristics specified at the thread or at the ORB level are allowed to be overridden at this level.

When a server gets deployed, a set of policies concerning the handling of QoS characteristics is deployed as well. At the same time, when a server becomes accessible after having exported its reference, policy functions are added. In turn, by importing the reference, the client provides to the ORB a validation capability. Therefore, the ORB Core can check the QoS characteristics of the client with respect to the criteria exported by the server.

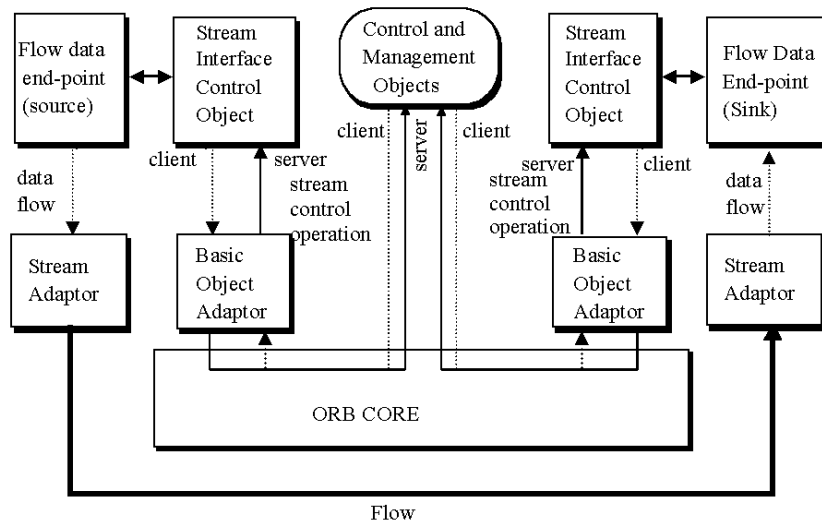


Figure 6.5 CORB architecture

Basically, OMG's approach of specifying QoS using abstract policies is still very generic. However, it is possible to specify certain QoS characteristics in conjunction with the following CORBA messaging features:

- Rebind Support
- Synchronisation Scope
- Request and Reply Priority
- Request and Reply Timeout
- Routing
- Queue Ordering

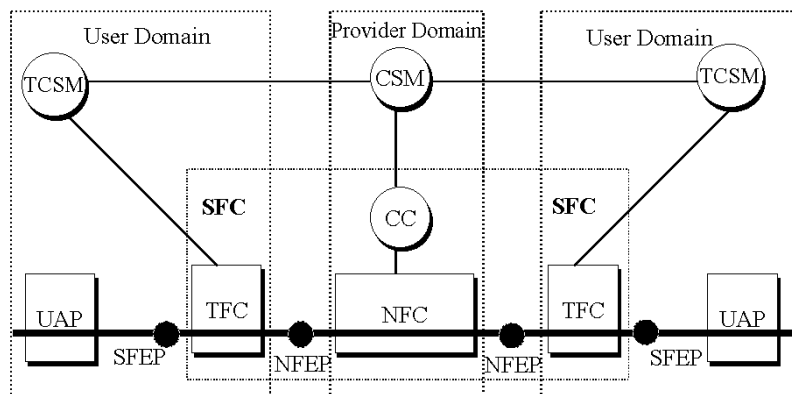
Figure 6.5 shows the principles of control of the CORB Architecture. Client and server control functions are separately handled from data flow control functions. At each data flow end-point there exists a server object that controls the flow of data. The server objects at the data flow end-points interact with a central controlling object. This structure of control is similar to the control structure of TINA (see Section 6.6.2). Both architectures handle data and control paths separately.

6.6.2 Telecommunications Information Networking Architecture (TINA)

The Telecommunication Information Networking Architecture (TINA) of the European TINA Consortium is described in detail in [11] and [23]. Architecturally, TINA is separated into the Native Computing and Communication Environment (NCCE), the Distributed Processing Environment (DPE) and into the Application Layer (AL). The NCCE contains the operating systems and the communication

protocol stacks. It is composed of a set of heterogeneous interconnected computing nodes. The DPE hides the heterogeneity and distributive nature of the NCCE. It thus minimises the knowledge required by the Application Layer about NCCE technology. The Application Layer contains management capabilities for the resource sub-layer and the service sub-layer. The service sub-layer comprises the specific part of the telecommunication service, e.g. a video-on-demand service, and additionally some generic procedures for the association of users to a service invocation. More importantly, the same sub-layer also contains the service session manager which deals with negotiation and control of service resources. In cooperation with the service session manager, the network resource manager provides connection configuration capabilities.

The components that provide services for setting-up and for managing connections are captured by the so-called TINA Network Resource Architecture (TNRA), which is outlined in Figure 6.6. TNRA is the structure of the resource sub-layer described above. At the top level of the TNRA there is the Connection Session Manager (CSM). The CSM provides service-independent interfaces about the control of End-to-end Stream Flow Connections (SFC). A SFC is separated from its user application components (UAP) by its Stream Flow End-points (SFEP). CSM shares control of a SFC with its local Terminal Communication Session Manager (TCSM) from the associated user domains. The part of the SFC which is controlled by the local TCSMs is called Terminal Flow Connection (TFC) and bridges the gap between the SFEP at the user application and the Network Flow End-point (NFEP) at the communication network. Both end-points SFEP and NFEP are either



Legend: TCSM: Terminal Communication Session Manager,
 CSM: Communication Session Manager,
 CC: Connection Coordinator, UAP: User Application,
 N/S/T FC: Network/Stream/Terminal Flow Connection,
 N/S FEP: Network/Stream Flow End Point

Figure 6.6 TIN-Architecture

application-independent or network-independent end-points respectively, i.e. they provide transparency of the service or technology. The inter-connections between NF end-points is provided by Network Flow Connections (NFC). NFCs are controlled network internally by Connection Coordinators (CC). SFC and TFC are logical representations of a connection, whereas the NFC is their physical counterpart.

The TIN-Architecture provides the necessary flexibility to get information from resources and from service components used by an application. The management components are optimally distributed in order to control local resources and to cooperate with remote management components, i.e. to feed-back or to feed-forward steering information, such that end-to-end control of QoS can be achieved.

6.6.3 Lancaster Architecture (QoS-A)

The Quality of Service Architecture (QoS-A)[3] has been developed at Lancaster University, UK and research is now continuing in collaboration with Columbia University, USA. The architecture consists of a number of layers and planes (see Figure 6.7). The highest layer consists of a distributed application platform augmented with services to provide multimedia communications and QoS specification. The task of the orchestration layer is to control jitter and regulate the rate for continuous media streams. This layer is supported by the transport layer which contains various configurable QoS services and mechanisms. The network layer is very closely linked with the transport layer and supported by the lowest two layers, the data-link layer and the physical layer.

QoS management is done in three vertical planes. The protocol plane separates control and data because of their different QoS requirements. The QoS maintenance plane contains several layer-specific QoS managers, each one being responsible for the monitoring and maintenance of their associated protocol entities. The flow-management plane is responsible for flow establishment, which includes admission control, QoS-based routing and resource reservation, QoS mapping (i.e., the translation of QoS parameters between layers), and QoS scaling (which constitutes QoS filtering and adaptation).

QoS-A is a very comprehensive framework that addresses many aspects of QoS provision, control and management. It can provide deterministic, statistical and best-effort service guarantees. In general, it can be said that its main focus is on the end systems rather than on the network. Flow synchronisation, flow scheduling, flow control and resource allocation in the network require further attention.

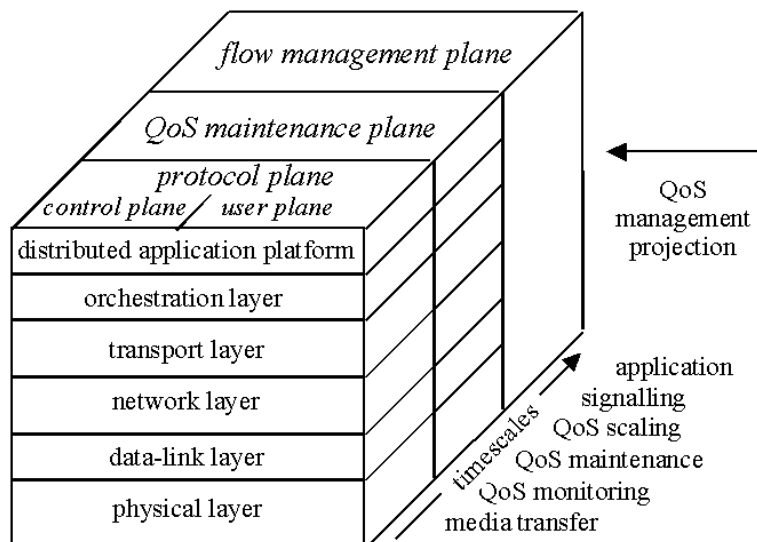


Figure 6.7 Lancaster Architecture

6.6.4 Heidelberg Architecture

The HeiProjects at IBM's European Networking Center in Heidelberg, Germany, have achieved a comprehensive framework for providing QoS guarantees in the end-systems and the network. The projects include the development of HeiTS (Heidelberg Transport System) for the transport of continuous media streams across the network[22] and HeiRAT (Heidelberg Resource Administration Technique) that provides a well-defined QoS for this transport[21].

HeiTS is designed for the communication of continuous media data. Its purpose is to exchange digital audio and video with QoS guarantees. This can be done in a one-to-one or multicast fashion. It uses HeiTP (Heidelberg Transport Protocol) as transport protocol, ST-II as network protocol and HeiDL as data-link layer protocol[8]. Although HeiTS uses the old-fashioned ST-II protocol for resource reservation, it can be adapted for RSVP[3].

HeiTS has several support mechanisms. As can be seen in Figure 6.8, it provides support for buffer management with the HeiBMS (Heidelberg Buffer Management System) and for platform portability with the HeiOSS (Heidelberg Operating System Shield). However, HeiRAT is the most important subsystem. It manages all resources that are critical for the processing and transmission of continuous media data: CPU time, bandwidth and memory. The QoS values are given in terms of maximum end-to-end delay, minimum throughput required and reliability class. The latter defines how the loss of data is treated. Furthermore, an application can specify

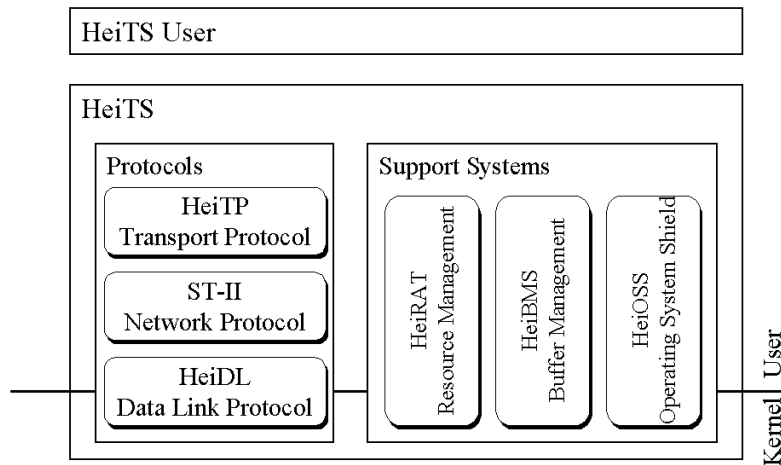


Figure 6.8 Heidelberg architecture

an interval from desired to worst-acceptable QoS values. In order to offer guaranteed or statistical QoS, QoS routing is employed to find the best path through the network.

A big advantage of the HeiProjects is their scalability[5]. In the case of MPEG, the continuous media stream can be divided into substreams. The stream that contains the important I-frames is transmitted over a guaranteed connection. The P- and B-frames are less important and can be transmitted over a best-effort connection. However, this division into substreams requires good synchronisation, which is still a weak point in this architecture.

6.6.5 Tenet Architecture

The Tenet architecture[15] has been developed at the University of California at Berkeley and consists of a family of protocols that run over networks which can provide guaranteed-performance services, such as FDDI or ATM. The user specifies the desired QoS in terms of delay, jitter and loss rate together with traffic characterisation parameters.

The protocol suite defines five protocols (see Figure 6.9). At the network level the Real-Time Internet Protocol (RTIP) delivers packets to meet the channels' real-time requirements. It is connection-oriented and performs rate control, jitter control and scheduling based on the QoS parameters of each connection. It provides an unreliable, simplex, guaranteed-performance, in-order packet delivery service.

Two protocols are defined for the transport layer, the Real-Time Message Transport Protocol (RMTP) and the Continuous Media Transport Protocol (CMTP). RMTP is concerned with message-based real-time transport between end-points. It

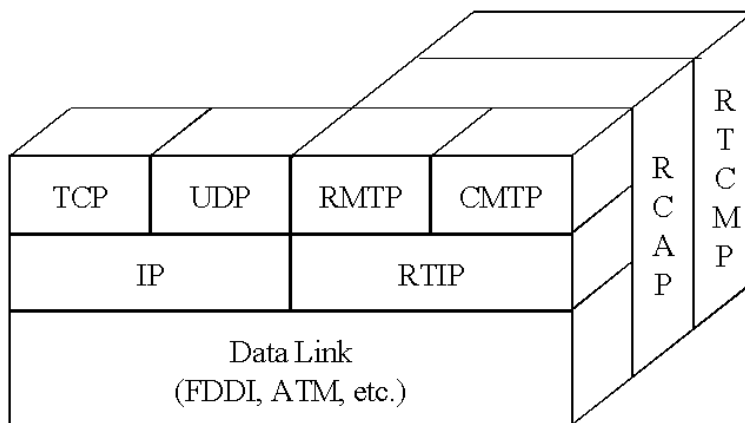


Figure 6.9 Tenet architecture

is unreliable and depends on RCAP to manage connections so that there is no congestion and on RTIP to provide rate-based flow control. One of the main services provided by RMTP is fragmentation and reassembly. CMTP is concerned with the transport of continuous media and offers a stream-based interface for isochronous applications.

The remaining protocols are control protocols. The Real-Time Channel Administration Protocol (RCAP) provides the signalling and control services in the Tenet suite. Its main functions are the setup and teardown of real-time channels in a heterogeneous internetwork and to provide status inquiries about already established channels. Finally, the Real-Time Control Message Protocol (RTCMP) is there to detect and manage error conditions. In the event of a network failure it computes a new route and allocates resources before redirecting a data stream.

Compared to the aforementioned QoS architectures, Tenet is much more focussing on the network rather than the end systems. In order to provide comprehensive end-to-end QoS, a more detailed consideration of the end systems would be necessary.

6.6.6 Omega Architecture

Omega[15] was developed at the University of Pennsylvania. It is built on a network that provides bounds on delay and errors and that can meet bandwidth demands. The architecture concentrates on the provision of QoS in the end points.

The system can be partitioned into two parts, the communication model and the resource model. The communication model (see Figure 6.10) contains the application subsystem layer that provides functions such as call management, as well as rate control, fragmentation, I/O functions, etc. which are the core of the Real-Time Application Protocol (RTAP). The transport subsystem layer provides

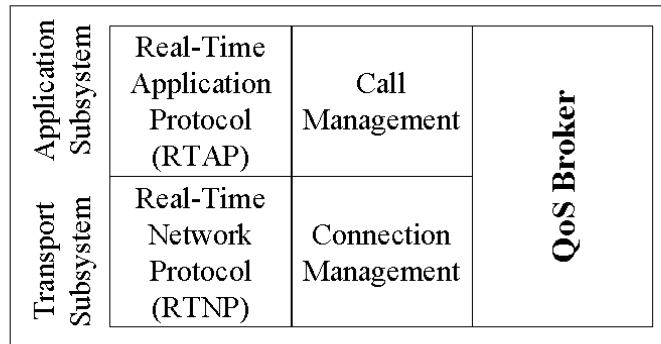


Figure 6.10 Omega architecture

connection management, FEC, timing failure detection, etc, which form the core of the Real-Time Network Protocol (RTNP). The service guarantees provided to the applications are negotiated by the so-called QoS Broker[16] during call establishment. The broker is responsible for the management of local and global end-point resources.

The resource model addresses the management of multimedia devices, CPU scheduling, memory allocation, and network resources. The resources in each of these domains (application, operating system, network) are described with appropriate QoS parameters. The application specifies its resource requirements with high-level (application) QoS parameters (such as frame rate and resolution). The operating system provides system resources (such as CPU slots, buffers and processing). The network finally provides network resources (such as bandwidth and packet size).

The essence of OMEGA is resource reservation and management of end-to-end resources. The architecture includes the application layer and allows the re-negotiation of the initially negotiated QoS parameters. However, it does not address QoS filtering or flow synchronisation.

6.7 Advances in Quality of Service Architectures

Without the possibility to retrieve appropriate information from system components QoS management cannot be performed. The previously invented QoS architecture, as sketched in Figure 6.3, shows access points and signalling paths which are necessary for providing reflective or adaptive QoS management respectively in the application layer and on the networking layer. Obviously, the mechanisms in the transport layer are resource-oriented, and the mechanisms in the application layer are service-oriented. Resource-oriented mechanisms are e.g. the point-to-point flow control or the admission control mechanism. These mechanisms can be found in the Lancaster Architecture, the Heidelberg Architecture and the Tenet Architecture.

CORBA, which can also be classified as being resource-oriented, has a very low level of visibility of its resources and is thus not open enough for introducing resource control mechanisms suitable for QoS-awareness. In contrast TINA, which is object-oriented, provides a very fine-grained structure on the application level. It further structures the interaction activities between application entities into phases of access, transportation and termination. It is thus possible for the application layer to assign QoS tasks to some of the TINA management objects and to identify access points at object interfaces and signalling paths between managing objects.

One of the most important features of QoS-aware architectures is the openness of their structures. Openness provides the necessary visibility of internal behaviour. Visibility is required for the various activities to enable QoS control, like controlling, filtering, shaping, monitoring etc. More precisely, openness is subdivided into observability and controllability as presented in Section 6.5. Whereas observability defines the constraints between the internal state variables and the outputs, controllability defines the relationship between the input and the internal state of a component or a system. To take these constraints into account a system designer must know which of the state variables are stringent to QoS.

Control requires decision procedures. Decision procedures interpret observations and generate actuation signals for the adaptation of resources or sources. Thus, decision procedures are policy-driven and may change according to traffic or service characteristics. Therefore, the following design constraints can be derived:

1. to identify openness constraints for a known QoS policy in terms of observability and controllability.
2. to identify continuous variables that are stringent to the QoS policy to be achieved. Define their relationships to input and output of the system.
3. to architecturally separate control functions from service functions. Define a clear interface between the control plane and the service or network plane.

The structure of the advanced QoS-aware middleware fulfills the three constraints introduced above, by which QoS control is achieved. This kind of middleware plays the role of a mediator between the application layer and the networking layer. The sensors are attached to the interfaces of components coping with transportation resources. These components are found at the edges of the network, i.e. routers, shapers etc. Actuators are either plugged to application interfaces or to resource management interfaces of the networking layer. Placing actuators depends on the adaptation policy applied. For example, the adaptation policy adapts the behaviour of an application to the observed traffic situation. Alternatively, the resources can be rearranged in a way that the same quality of service is maintained even after the observation of scarce resources. The third class of agents of the QoS-aware middleware are the discriminator agents. Discriminators may reside not only as part of the middleware but also in the various domains of the application and of the network edges. Thus discrimination looks like a distributed process that provides the decision making task of the adopted QoS policy. During

execution, the decision making process refers to the negotiated values, thresholds, limits etc, that are captured by the QoS contract. The QoS-aware middleware is thus capable of providing the adaptation infrastructure to the QoS management, that include signalling between sensor, actuators and discriminators, the access to QoS contracts from any location, and the roaming of decision-making agents, i.e. the discriminators. This approach achieves a clear distinction between QoS control, i.e. management function and service function.

6.8 Conclusions

This chapter has introduced quality of service mechanisms and architectures for interactive multimedia services. It first listed some of the issues in QoS management functions and then surveyed the most common architectures that claim to be QoS-aware. Research in recent years has shown that it is not enough to provide guarantees with respect to network resources. The sole reservation of bandwidth in the network cannot ensure high quality of IMM services. Although these guarantees are essential, they need to be combined with service guarantees with respect to end-systems. The scheduling of multimedia streams at the end-systems is an important factor. Furthermore, QoS guarantees need to be provided on an application-to-application basis. Resource management at the end-systems as well as in the network needs to be related to QoS parameters given by applications or users.

Multimedia communication is very dynamic in nature. Unfortunately, most reservation schemes and QoS architectures are still quite static. What is required are more adaptable structures, similar to the suggested QoS reference architecture, that can respond to QoS violations and network or device failures. However, the diversity of networks and applications makes it very difficult to implement such a comprehensive and reflective system structure as is required for the deployment of high-quality interactive multimedia services on a large scale.

References

- [1] Aurrecoechea, C., Campbell, A.T. and L. Hauw, (1998) "A Survey of QoS Architectures", ACM/Springer Verlag Multimedia Systems Journal, Special Issue on QoS Architecture, Vol. 6 No. 3, pg. 138-151, May 1998. [a former version can be found in: "A Review of QoS Architectures", Proceedings of the 4. International IFIP Workshop on Quality of Service IWQoS96, Paris, March 6-8, 1996.]
- [2] Banerjæ A., Ferrari D., Mah B.A., Moran M., Verma D.C. and Zhang H. (1996) "The Tenet Real-Time Protocol Suite: Design, Implementation and Experiences", IEEE/ACM Transactions on Networking, 4 (1), pp. 1-10.
- [3] Campbell A., Coulson G. and Hutchison D. (1994) "A Quality of Service Architecture", ACM Computer Communication Review, 24 (2), pp. 6-27.
- [4] OMG (1998) "CORBA Messaging, Joint Revised Submission", OMG TC Document, OMG, May 18, 1998, <ftp://ftp.omg.org/pub/docs/orbos/98-05-05.pdf>
- [5] Delgrossi L., Halstrick C., Hehmann D., Herrtwich R.G., Krone O., Sandvoss J. and Vogt C. (1993) "Media Scaling for Audiovisual Communication with the Heidelberg

- Transport System”, Proceedings of the ACM Conference on Multimedia (Multimedia'93), Anaheim, CA.
- [6] Eberlein A. (2001) “Interactive Multimedia: A Review”, International Journal of High Performance Computer Graphics, Multimedia and Visualization 1 (1), pp. 9-29.
 - [7] Guo X. and Pattinson C. (1997) “Quality of Service Requirements for Multimedia Communications”, Proceedings of the Time and the Web Symposium, UK.
 - [8] Huard J.-F. and Lazar A.A. (1997) “On QoS Mapping in Multimedia Networks”, Proceedings of the Twenty-First Annual International Computer Software & Applications Conference, Washington, D.C.
 - [9] R. Bradsen ISI, L.Zhang UCLA (1997) “Resource Reservation Protocol (RSVP)”, Network Working Group [www.ietf.org/ RFC#2209](http://www.ietf.org/RFC#2209)
 - [10] Y. Berner et al. (1998) “A Framework for Differentiated Services”, Internet Draft draft-ietf-diffserv-framework-01.txt
 - [11] E.Koerner (1998), “Methods and Elements for the Construction of Collaborated Services in TINA”, These de doctorat, University de Liege, ISSN 0075-93333
 - [12] Little T.D.C. and Venkatesh D. (1994) “Prospects for Interactive Video-on-Demand”, IEEE Multimedia, 1 (3), pp. 14-24.
 - [13] OMG(1996) “Messaging Service RfP”, OMG, November 1996, <ftp://ftp.omg.org/pub/docs/orbos/1996/96-03-16.pdf>
 - [14] Nahrstedt K. and Steinmetz R. (1995) “Resource Management in Networked Multimedia Systems, Computer”, 28 (5), pp. 52-63.
 - [15] Nahrstedt K. and Smith J.M. (1996) “Design, Implementation and Experiences of the OMEGA End-Point Architecture”, IEEE Journal on Selected Areas in Communications, 14 (7), pp. 1263-1279.
 - [16] Nahrstedt K. and Smith J.M. (1995) “The QoS Broker”, IEEE Multimedia, 2 (1), pp. 53-67
 - [17] OMG (1997) “Quality of Service (QoS)” OMG Green Paper, Working Draft, Version 0.4a, OMG, June 1997, <ftp://ftp.omg.org/pub/docs/ormsc/97-06-04.pdf>.
 - [18] Svend Frølund and Jari Koistinen (1998) “Quality of Service Aware Distributed Object Systems”, Hewlett-Packard, April, 1998.
 - [19] Tanenbaum A.S. (1996) “Computer Networks”, Prentice Hall.
 - [20] Vogel A., Kerhervé B., von Bochmann G. and Gecsei J. (1995) “Distributed Multimedia and QoS: A Survey, IEEE Multimedia”, 2 (2), pp. 10-19.
 - [21] Vogt C., Wolf L.C., Herrtwich R.G. and Wittig H. (1998) “HeiRAT – Quality of Service Management for Distributed Multimedia Systems”, to appear in ACM Multimedia Systems Journal - Special Issue on QoS Systems.
 - [22] Wolf L.C. and Herrtwich R.G. (1994) “The System Architecture of the Heidelberg Transport System”, ACM Operating Systems Review, 28 (2), pp. 51-64.
 - [23] ISO (1995) IS10746-2 “IT ODP-RM Open Distributed Processing Part 2: Descriptive Model”
 - [24] ETSI (1995) ETSI and the Information Society – “State of Art 1995”
 - [25] ISO (1997) ISO/IEC ITU-T Recommendation IS 13236 / X.641, “Information Technology – Quality of Service – Framework”, December 1997.
 - [26] ISO (1998) ISO/IEC ITU-T Recommendation TR 13243 / X.642, “Information Technology – Quality of Service - Guide to methods and mechanisms”, September 1998.
 - [27] ISO (1999) CD15935 (1999) “IT ODP-RM – Quality of Service”, ISO/IEC JTC1/SC7 N1996

- [28] TINA-C (1995) “Computational Modelling Concepts”, Archive Label TB_A2.HC.012.1.2.94, February 1995
- [29] J.B. deMeer (1999) “On the construction of Reflexive System Architectures”, Middleware Workshop RM2000, 7.4.2k N.Y.C.
- [30] I.Foster, A.Roy, V.Sander (2000) A Quality of Service Architecture that combines Resource Reservation and Application Adaptation. IWQoS2000 Proceedings pp. 181-188
- [31] T.Walter, I.Schieferdecker, J.Grabowski (1998) Test Architectures for Distributed Systems – State of the Art and Beyond. IWTCS 1998 Proceedings pp. 149-174
- [32] B.C.Kuo 1995 Automatic Control Systems, Prentice Hall, ISBN 0-13-304759-8

Glossary of Architectural Concepts

BER	Bit Error Rate
CC	Connection Coordinator
CORBA	Common Object Request Broker Architecture
CHQ	Controlled Highest Quality (Negotiation term)
CMTF	Continuous Media Transfer Protocol
CSM	Connection Session Manager
DPE	Distributed Processing Environment
HQA	Highest Quality Attainable (Negotiation term)
IMM	Interactive Multimedia
LQA	Lowest Quality Acceptable
Mbps	Mega Bits per Second
MPEG	Moving Pictures (Definition) Expert Group
NCCE	Native Computing and Communication Environment
NFC	Network Flow Connection
NFEP	Network Flow End-point
ODP	(ISO) Open Distributed Processing
OSI	(ISO) Open Systems Interconnection (Reference Model)
PoA	Point of Actuation
PoO	Point of Observation
QoS	(ISO) Quality of Service
RT	Real Time
RCAP	RT Channel Administration Protocol
RTTP	RT Transport Protocol
RTAP	RT Application Protocol
RTIP	RT Internet Protocol
RMTP	RT Message Transfer Protocol
RSVP	Resource Reservation Protocol
RTCMP	RT Control Message Protocol
RTNP	RT Network Protocol
SFC	Stream Flow Connection
SFEP	Stream Flow End-point
S_IF	Stream Interface
TCSM	Terminal Communication Session Manager

TFC Terminal Flow Connection
UAP User Application Component

Glossary of Standardisation Institutions

ETSI European Telecommunication Standards Institute
IETF Internet Engineering Task Force
ISO International Standardisation Organisation
ITU-T International Telecommunication Union –Telecommunication Standardization Sector
OMG Object Management Group
TINA-C Telecommunications Information Networking Architecture Consortium