ODP View on Quality of Service for Open Distributed Multimedia Environments

Peter Leydekkers
Email: P.Leydekkers@research.kpn.com
KPN Research
P.O.Box 15000
9700 CD Groningen - The Netherlands

Valérie Gay
Email: Valerie.Gay@masi.ibp.fr
Université Paris VI, Laboratoire MASI,
4, place Jussieu,
75252 Paris Cedex 05 - France

Abstract: Quality of Service (QoS) requires standardization to achieve some consistency in terminology, models and mechanisms to operate successfully in an open distributed environment. This paper embraces this standardization effort and addresses the QoS issue from an ODP perspective. The objective of this paper is to relate the multimedia QoS concepts found in the literature and ISO-QoS framework to the ODP architecture. In the computational viewpoint it details the ODP "Environment Contract" concept using the TINA Object Definition Language to express QoS characteristics for computational objects. It also gives typing rules adapted to stream interface characteristics. In the engineering viewpoint, it refines the "stream interface reference" concept to incorporate QoS specifications that are used for the exchange of QoS information in an open distributed environment. Channel establishment is then described according to the QoS phases described in the ISO-QoS Framework. Finally, an example is used to illustrate the mapping of engineering QoS parameters onto ATM.

Keywords: QoS, Distributed Multimedia Systems, RM-ODP, TINA-ODL.

1. Introduction

The majority of new applications, especially those destined for deployment in an open distributed environment as defined by RM-ODP will include multimedia as a key element. This is especially true now that multimedia is easier to provide due to the convergence of telecommunications, computing and entertainment, as well as the increasing power of personal computers. These factors demand that distributed (telecommunications) architectures have to deal with multimedia services that have stringent QoS requirements.

The QoS issue has received quite some attention the last few years resulting in many papers, and an ISO standardization effort to align terminology, models and QoS mechanisms for open telecommunication environments. Standardisation is necessary for open distributed environments where multiple stakeholders play a role, such as, end-users, network providers and service providers. The ISO Reference Model of Open Distributed Processing (RM-ODP) is developed to provide a general architecture for open distributed environments. But the question arises how well does RM-ODP deal with QoS? If RM-ODP is intended to be the overall architecture for open distributed processing it needs to incorporate QoS appropriately.

The objective of this paper is to study this question and we approach QoS from an ODP perspective. It relates multimedia QoS concepts found in the literature, and ISO-QoS framework to the ODP viewpoints and viewpoint languages to express QoS aspects. It discusses how certain multimedia aspects can be modelled using the computational and engineering viewpoint languages since the most important concepts and structures for distributed applications are found in these viewpoints. The paper will show that RM-ODP is a suitable framework to describe QoS but it needs several extensions to deal with QoS adequately.

The paper is organised as follows:

Section 2 relates the different levels of QoS commonly found in the literature to the ODP viewpoints to position this paper. Section 3 focuses then on QoS in the computational viewpoint and we detail the ODP "Environment Contract" concept using the TINA Object Definition Language to express QoS characteristics for computational objects. This section also gives typing rules adapted to stream interfaces. In the engineering viewpoint (section 4), we refine the "stream interface reference" concept to incorporate QoS specifications that are used for the exchange of QoS information in an open distributed environment. Channel establishment is then described following the QoS phases described in the ISO-QoS Framework. Section 5 illustrates the

In the proceedings of IWQoS'96 - Paris - March 96

mapping of general QoS characteristics onto ATM technology. The conclusion presents some on-going activities with respect to standardisation to include the proposals described in this paper.

2. Multimedia QoS specification and the ODP viewpoints

The perceived quality is an important aspect for continuous data exchange. The QoS concept is used to describe the quality of perception. QoS is used in this paper according to [17] and is defined as: "*QoS represents the set of those quantitative and qualitative characteristics of a distributed multimedia system necessary to achieve the required functionality of an application*". Quantitative characteristics can be evaluated in terms of concrete measures such as bits/s, task processing time etc. [12]. Qualitative characteristics specify the expected services offered by the system such as interstream synchronisation, ordered delivery of data, scheduling mechanisms [12]. This definition of QoS is more specific than the RM-ODP definition of QoS [4] which defines QoS as "*a set of qualities related to the collective behaviour of one or more object*". The ODP definition is more related to objects whereas Vogel's definition [17] is more general and not only restricted to objects that make up a distributed system.

QoS parameters can be identified at different levels of abstraction. From end-user perspective the QoS parameters are of different nature (more qualitative oriented and global) than those at system level (more quantitative oriented and detailed). In the literature, QoS is used at different levels of abstraction and it is sometimes difficult to find correspondences in terminology and the right abstraction level. A relation with a standardized framework like RM-ODP [2] [3] or ISO-QoS framework [4] is desirable to achieve some consistency. We will relate the different levels of QoS commonly found in the literature to the ODP viewpoints to allow better positioning of this paper.

RM-ODP defines 5 viewpoints used to describe a system from different perspectives [3]. Each viewpoint description represents a different abstraction of the system of concern. The *enterprise* viewpoint specification focuses on the requirements, purpose and policies that apply to the system. The *information* viewpoint is concerned with the information that needs to be stored and processed in the system. A computational specification is concerned with the description of the system as a set of interacting objects. It focuses on the objects, interfaces and describes how distributed applications are structured in a distributed transparent way. The *engineering* viewpoint specification focuses on the infrastructure required to support distributed processing. Finally the *technology* viewpoint specification describes suitable technologies to support the implementation of the system.

Table 1 shows the different QoS levels organised along the ODP viewpoints.

<table>
<tr><th>ODP Viewpoint</th><th colspan="2">QoS level</th><th colspan="2">Characteristics</th><th colspan="2">Examples</th></tr>
<tr><td>Enterprise</td><td colspan="2">Subjective QoS</td><td colspan="2">End-user oriented QoS requirements (not formal and subjective)</td><td colspan="2">"Audio must be along the video"; "Video should be similar to television-quality"</td></tr>
<tr><td>Information</td><td colspan="2">Objective QoS</td><td colspan="2">Precise statement of QoS requirements derived from the subjective QoS specification. Often application independent</td><td colspan="2">25 fps (PAL), 30 fps (NTSC)
Aspect ratio of 4:3 (PAL TV Format)
Telephone Quality: 8 kHz</td></tr>
<tr><td>Computational</td><td colspan="2">Application QoS</td><td colspan="2">Describe the QoS requirements for the applications often specified in terms of media quality and media relations</td><td colspan="2">Lip synchronisation between audio and video can be expressed in Ms. (e.g. synchronisation skew ± 80Ms)</td></tr>
<tr><td>Engineering</td><td>System QoS</td><td>Network QoS</td><td>Describe the QoS requirements from the operating system</td><td>Describe the QoS requirements from the network</td><td>Buffersize,
Operations/s,
Memory (Mb)</td><td>Throughput (Mbit/s),
Jitter (Ms),
Delay (Ms)</td></tr>
<tr><td>Technology</td><td colspan="2">QoS properties</td><td colspan="2">Describe the QoS characteristics of devices, operating system and network technologies. QoS properties.</td><td colspan="2">Video device (PAL format),
Audio device (μ-law)
Cell loss rate for ATM,</td></tr>
</table>

Table 1: QoS levels related to ODP viewpoints

The *Subjective QoS* deals with end-users expressing QoS requirements that are different for each end-user and depend on the users' perception of the received information flows. These subjective QoS requirements can be expressed in an *enterprise* QoS specification. An enterprise viewpoint description expresses QoS as perceived or required by the end-user using parameters that describe QoS from the end-user perspective.

The way end-user expresses QoS requirements such as "audio is most important in a video conference" or "naturalness for audio" can often be related to a certain level of quality. This is often referred to as *Objective QoS* that expresses QoS requirements in QoS parameters and associated values [16]. Objective QoS specifications can be related to an *information QoS* specification of a system. Objective QoS descriptions contain specifications of the structure of the information objects that define the information type, quality attributes and rules stating the relationships between the information objects (e.g. synchronisation between audio and video information objects).

The *computational* specification identifies the computational objects involved in the system under design. QoS related to computational objects is often expressed in *Application QoS* describing the QoS requirements related to the objects that interact. It is often expressed in terms of media quality and media relations. Media quality includes source/sink characteristics such as media data unit rate and transmission characteristics such as end-to-end delay [9].

The *engineering* viewpoint does not hide the distribution transparency and shows the collection of (computer) systems and networks that together provide the distributed processing environment. *System QoS* and *network QoS* specifications both relate to the engineering viewpoint. System QoS describes the QoS requirements of the operating system of the involved (computer) systems whereas the Network QoS describes the QoS requirements related to the network.

The *technology* viewpoint specification addresses the technology solutions applied to implement the system. The technology used specifies the *QoS properties* of the various components (hardware/software) used to implement the system.

As stated in [3], the viewpoint specifications are not independent and there should be some correspondence and consistency between the viewpoint specifications. Applied to QoS, an interesting issue is how to relate a certain viewpoint QoS specification to another viewpoint QoS specification. This mapping is not obvious since it is in general not a one-to-one mapping.

A possible translation from an enterprise QoS specification (subjective QoS) to an information QoS specification (objective QoS) can be achieved using a method called “Quality Query by Example” [13]. This method shows end-users' examples of varying quality, such as images of different size, resolution and colour depth, or speech of telephone or CD quality from which he/she may select one. The example corresponds to a certain QoS specification (objective QoS). An example representing Television quality corresponds to 25 frames/s (PAL) and a ratio of 4:3.

The relation between computational and engineering QoS specifications mainly depends on how a computational specification is mapped onto an engineering configuration and is often a choice of the application designer. In general, a QoS characteristic expressed in the computational QoS specification influences one or more QoS parameters in the engineering QoS specification. For example, a frame rate interval of 20-25 fps expressed in a computational application QoS specification may influence both the engineering network QoS specification (e.g., throughput (bits/s)) and the system QoS specification (processing capacity (instructions/s)).

In this paper we focus on the computational and engineering viewpoint, and refine several ODP concepts to express QoS. The expression of QoS in the RM-ODP framework is still an open issue and subject of standardisation [7] [8].

3. QoS in the Computational viewpoint

In the computational viewpoint, multimedia oriented applications are modelled in terms of objects interacting via interfaces. Each object is described by its behaviour, its environment contract and by its interfaces. RM-ODP defines *stream interfaces* which are used by computational objects that exchange continuous data (e.g., voice, video). The stream interfaces are composed of a signature, a behaviour and an environment contract.

QoS properties are specified a static way using *object environment contract* and *interface environment contract*, see figure 1. The *environment contract* [1] [3] expresses the requirements and obligations which have to be fulfilled and it addresses issues such as the performance, availability, and security of the objects and indications of behaviour which invalidates the contract. The object environment contract is defined for the object as a whole, including all the interfaces it supports. The interface environment contract is specific to each interface. The values specified in the object environment contract apply to all interfaces and might constrain interface environment contracts. For example, statements about the availability of an object influence the availability of all the interfaces it supports.

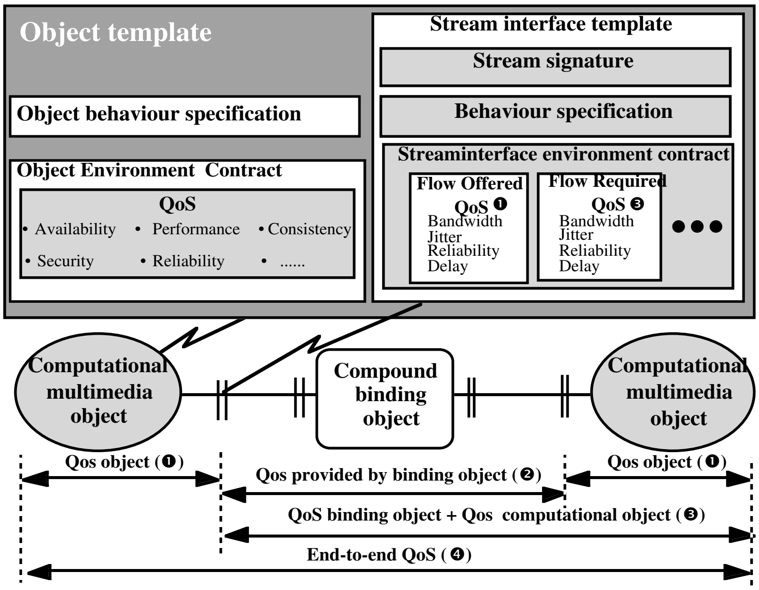


Figure 1: Computational concepts used for the specification of QoS

Interactions between computational interfaces are only possible if a *binding* (i.e. some communication path) has been established between them [3]. QoS is handled in a dynamic way by a compound binding object that is in charge of this binding. Compound binding abstracts over multiplex mechanisms like a video bridge or a coding exchanger.

QoS specifications in the various environment contracts contribute to the end-to-end QoS that is desired for the multimedia objects involved. The end-to-end QoS (❹) is composed of the QoS provided by the multimedia objects (❶) and the QoS provided by the compound stream binding object (❷). The QoS offered by a computational object is determined during processing and (internal) transportation inside the computational object. The parameter values are expressed in the *QoS offered contract* for each flow involved in the binding. The *required QoS contract* is similar to the offered QoS contract but differs for the actual parameter values. In general the values in the QoS offered contract are higher than the corresponding values in the required QoS contract. This is because the processing and transportation overhead introduced by the other multimedia objects and binding objects respectively have a negative effect on the parameter values (❸). The QoS specified for the compound binding object specifies values experienced during processing and transportation by the binding object (❸). In this case, delay will specify the processing and transportation delay between the two stream interfaces that are bound by this compound binding object. The environment contract specification for the stream interfaces in the binding object should correspond to the environment contract specification of the bounded stream interfaces.

A *control interface* on the *compound binding object* might allow manipulation of engineering flow synchronisation mechanisms as part of the provision of lip-synch. The ways QoS is controlled and monitored by the binding object control interfaces are outside the scope of this paper and have been discussed in [14] [15].

An important action in the binding process is the *typing* of stream interfaces (including matching of the stream interface environment contracts) involved in the binding. Flows from a number of producers may be combined to provide a composite flow to a single consumer. For example, a video flow from one source and an audio flow from another might be combined into a single television flow as image and associated commentary. Typing will be discussed in detail in section 3.2.

3.1 Static specification of QoS in the environment contract

In the computational viewpoint QoS requirements and offerings are described declaratively in the environment contract. It abstracts from the engineering (and technology) mechanisms that have to provide and guarantee these QoS statements. The QoS is specified in terms of media quality and media relations. Thus, the application QoS parameter-value pairs are closely related to the parameters defined for the stream signatures. For example, a QoS requirement expressing the throughput offered by a stream interface should be expressed in the number of frames/s (fps) that will be offered by the stream interface flow and *not* in Mbit/s.

The ISO QoS Framework defines a set of general QoS characteristics that are of importance to communications and processing systems. The QoS characteristics in the QoS Framework are very general and need refinement to be useful in specifying QoS constraints for a particular application. Since real-time multimedia applications are time-critical we focus on those QoS characteristics that express the timeliness character. Table 2 shows how the general QoS characteristics (first column) are refined for audio/video QoS characteristics (second column).

QoS Framework	ODP Environment Contract	Stream interface environment contract (example)
Time delay characteristic (minute, sec, msec etc.)	Jitter (variation in frames, samples) Delay (frames, samples)	Jitter (+/- 1 frame, +/- 100 samples) Delay (+/- 3 frames, +/- 500 samples)
Throughput characteristic	Application information throughput (fps, samples/s)	Throughput (25 fps, 8000 samples/s)

Table 2: QoS Framework characteristics related to the ODP environment contract for multimedia QoS

An important aspect related to the QoS characteristics described in the environment contract is a description of the *guarantee level* provided by a computational object at its interfaces for each QoS parameter-value pair. An agreed guarantee level should be obeyed by the object (and its interfaces). Three classes of service commitment are distinguished [4]: *Compulsory, Statistical reliable (or Guaranteed service*) and *Best effort service*. A guarantee level can be assigned to each parameter in an environment contract, or a certain guarantee level can be reserved for the interface or object as a whole. The approach followed here allows for a flexible service commitment specification for each parameter. This feature is desirable if, for instance, an object supports a stream interface for the exchange of real-time data flows. This might require a guaranteed level for the throughput but a best effort guarantee for reliability.

An environment contract can be expressed in an implementation independent way using TINA's Object Definition Language (TINA-ODL) [10] [11] which is an extension of the OMG-IDL. The syntax description for the environment contract specification is defined in [10]. Based on this syntax, an example is shown in table 3 how QoS declarations can be specified for a stream interface supporting video. It specifies the Video QoS describing the throughput, jitter and delay characteristics. Depending on the guarantee level different values will be provided.

```
struct VideoQoS {
        union Throughput switch (Guarantee)
        { // expressed in frames per seconds
         case Statistical: float Mean;
         case Compulsory: float Peak;
         case BestEffort: struct Interval { float min; float maxd; };
};
        union Jitter switch (Guarantee)
        { // the time variation a frame gets through a service
         case Statistical: float Mean;
         case Compulsory: float Peak;
         case BestEffort: struct Interval {float min; float maxd; };
};
        union Delay switch (Guarantee)
        { // delay
         case Statistical: float Mean;
         case Compulsory: float Maximum;
         case BestEffort: struct Interval {float min; float maxd;}
};
```

Table 3: TINA-ODL example specification of a stream interface environment contract

The TINA-ODL syntax for environment contract is very flexible and allows various ways to express the QoS constraints on interfaces and objects. This flexibility is necessary since the environment contract has to accommodate many different types of objects operating in different circumstances and thus requiring various QoS specifications. The QoS statements made in the computational specification should be fulfilled by the underlying mechanism. Thus the computational expression for a certain jitter boundary for video frames with a compulsory guarantee requires for special engineering solutions to meet that requirement.

The QoS environment contract specification allows for a flexible QoS negotiation between all computational interfaces involved. QoS contracts can be accomplished that meet each computational object involved. However, if the established QoS agreements differ much between the computational objects, it implies that the compound binding object (figure 1) has to match all these QoS differences. This might require complicated engineering mechanisms which is not always feasible. For example if a producer sends video frames at 25 fps and the consumer only accepts 20 fps the binding object needs additional mechanisms to mach the difference in frame rate.

3.2 QoS typing

Typing rules apply when we need to bind two or more stream interfaces (producer/consumer). Typing stream interfaces takes into account the flow types, causality of the flows, and the environment contract (including stream QoS). Stream interface typing is especially important for the binding object that is responsible for the binding of two or more stream interfaces. This paper proposes the following stream interface typing rules:

Two stream interface can bind if

1. They have a compatible behaviour, compatible environment constraints, and
2. It exists two flows to bind having compatible information type, compatible flow QoS and an opposite causality.
3. Two flows have compatible QoS, if the QoS required by the consumer flow is less than or equal to the QoS offered by the producer flow.

The constraints to bind two stream interface types (TSt1 and TSt2) can be expressed more formally as illustrated in table 4.

Tst	is a type of stream interface	C(TSt)	is a class of interfaces satisfying TSt
Sst	is a stream interface specification	$\approx$	compatible with
f	is a flow	st	is a stream
B	Behavior	E	environment constraints (including stream QoS)
I	information type	RQoS	flow Required QoS
Ca	Causality (consumer or producer)	OQoS	flow Offered QoS

SSt = {B, E, {(Ii,Cai), i>0}}

PossibleBind (TSt1, TSt2) $\equiv$ ($\forall$st1, st2 : s1 $\in$ C(TSt1) $\wedge$ s2 $\in$ C(TSt2):

1. B(s1) $\approx$ B(s2) $\wedge$ E(s1) $\approx$ E(s2) $\wedge$
2. ($\exists$f1, f2 : f1 $\in$ s1 $\wedge$ f2 $\in$ s2 | I(f1) $\approx$ I(f2) $\wedge$ Ca(f1) $\neq$ Ca(f2)) $\wedge$
3. IF CA(f1) = "producer" THEN RQoS(f2) $\leq$ OQoS(f1) ELSE RQoS(f1) $\leq$ OQoS(f2)))

Table 4: Formal rules for typing of two stream interface type

In general, the stream typing process can be divided into two steps. First, the identification of the type equivalence (different causality, compatible behaviour) between the primitive flows in the stream signatures. Second, the evaluation of the stream environment contracts and of the flow QoS compatibility.

4. QoS in the Engineering viewpoint

The ODP engineering viewpoint focuses on the infrastructure and mechanisms to support distributed processing. In this viewpoint the functions and mechanisms are identified to manage the QoS of a distributed system. A distributed system in this viewpoint is a collection of networked (computer) systems.

The QoS framework identifies a set of QoS functions that perform activities such as the establishment of QoS for a set of QoS characteristics or monitoring of the observed values of QoS. Many of these QoS functions need information about the QoS characteristics of the involved objects as expressed in the computational environment contract. This implies that there must be some general means to pass QoS information from one system to another system. In distributed systems, information about object instances is passed using interface references. Currently, most distributed systems only support operational interfaces, and for the exchange of QoS information no explicit space is reserved in the interface reference.

Stream interfaces are not yet implemented in distributed processing environments like CORBA [21] [22] and especially stream interfaces have stringent QoS requirements that need to be fulfilled to obtain useful interaction. We will focus in this section on the stream interface reference and show how the computational environment contract can be included in this stream interface reference. Furthermore, it shows how stream interface references can be passed in an ODP compliant environment. We will not discuss the QoS mechanisms that use the information in the interface reference to perform a certain QoS management function.

4.1 ODP Interface reference

Interface references are unambiguous identifiers for the interface they reference. In an ODP system it is assumed that the nucleus is responsible to create an unambiguous reference. Given such a reference it is possible to discover the type of the interface, and a communication address at which binding to it can be initiated. It also contains other information about the expected behaviour of stubs, binder and protocol objects within the channel. The interface reference conveys complex information which should be interpretable throughout the ODP system.

If the content of an interface reference becomes too large for convenient storage or transmission, then all or part of the content can be held in a directory and replaced in the interface reference by a small key which can be used to access that content from the repository when required. The objects passing such interface references must have some other means of determining the repository at which the content is held. For example, in some systems, this might be some well-known interface.

In [4] an initial attempt is made to define the structure of an interface reference which is represented in Backus Naur Form (BNF) in table 5.

<interf-ref> ::= <null> | <direct reference> | <non-interpreted-reference>
<direct-reference> ::= <interf-type> <channel-class> <location-info> <relocation-info> <additional-info>
<non-interpreted-reference> ::= <interpreter-reference> <opaque-info>

Table 5: BNF description of interface reference structure

Focussing on the `direct reference` part, the `channel-class` field provides information about the stub, binder and protocol objects used in the channel local to the referenced interface. This information is needed to configure the remote end of the channel using equivalent stub, binder and protocol objects. The channel class also identifies the supporting objects required for the channel (e.g., interceptor).

The `location-info` field provides the necessary information for the binding to the location of the interface at the time it was created. The information includes network and sufficiently unique node-specific addressing. An interface reference may include a set of different pieces of addressing information, corresponding to different access paths. Location information will, in general, be context sensitive, because of this association with possible access paths. The format of the location information will depend on the channel class in the interface reference.

The `relocation-info` field identifies a relocator object that manages a repository of locations for interfaces, and this information is used when an interface changes of location. The `additional-info` field provides a place-holder for functions not directly related to binding, e.g. interface reference tracking

The `non-interpreted-reference` is used in systems crossing different interface reference domains where different naming policies apply (e.g., CORBA object reference [19] versus ANSA interface reference [20]). The `interpreter-reference` field identifies an object able to replace the `opaque-info` part of a `non-interpreted-reference` with a `direct reference.` (e.g., a translator object that is able to convert a CORBA object reference to an ANSA interface reference).

RM-ODP interface reference	ORB inter-operability object reference
Interface type	Interface type
Channel template	Communication protocol information
Additional information	Additional information to support other functions
Channel template Location information Relocation information Additional information	ORB interoperability information (IOR)

Table 6: Correspondence between ODP and OMG references

The ODP parameters identified in table 5 correspond to the OMG object reference parameters [19] as shown in table 6 [7] [22]. However, both ODP and OMG lack the explicit support for QoS specification in the reference structure. Section 4.2 proposes a syntax for stream interface references that provides explicit support for the QoS specification.

4.2 QoS in a stream interface reference

A stream interface reference has the same general characteristics than the operational one. Its real-time multimedia characteristics appear mainly in the

- computational interface type that it implements, including the QoS specification (i.e., QoS offered/ QoS required),
- channel configuration of stubs, binders, protocol objects and,
- possible interceptors needed that have real-time characteristics.

A refinement of the BNF definition for interface references specialized for stream interfaces using TINA-ODL syntax is shown in table 7.

```
<interf-type> ::= <stream-interf-typ>|<operational-interf-type>|<signal-interf-type>

<stream-interf-type> ::= <stream-interf-ref-name> {<flow-description> ";"}+

<stream-interf-ref-name> ::= <string_type>

<flow-description> ::= <flow-name> <direction> <flow-type> <flow-QoS>

<flow-name> ::= <string_type>

<direction> ::= "producer"|"consumer"

<flow-type> ::= <type_spec>

<flow-QoS> ::= <QoS_attr_type> <QoS_attr_name>

<QoS_attr_type> ::= <simple_type_spec>

<QoS_attr_name> ::= <simple_declarator>
```

Table 7: Refinement of the BNF definition for stream interface reference

The `flow-description` contains information about each flow in the stream interface that reflects the computational stream interface specification. It describes the name of the flow (`flow-name`), the direction of the flow (`direction`), the type of flow (`flow-type`) which can be for instance audio or video, and the associated QoS characteristics (`flow-QoS`).

The information in the interface reference is needed, for instance, to determine whether two flows can be bound using the rules outlined in section 3.2. Focussing on the flow-QoS, a possible C-structure is presented in table 8 based on a compulsory QoS agreement which implies that the values for the QoS parameters indicate the highest quality.

```
struct { // computational oriented parameters
        long        sample-rate;     // the sample rate for audio
        long        frame-rate;      // the frame rate for video
        ...
        // engineering oriented parameters
        long        throughput;      // bytes/s, max. throughput of the flow
        long        jitter;          // delay variation (msec)
        long        error-rate;      // magnitude order (10exp-)
        boolean     flow-control;    // indicates if flow control is supported
        long        loss-rate;       // magnitude order (10exp-)
        long        delay;           // (msec)
        long        average-rate;    // bandwidth (bits/sec)
        ...
        };
```

Table 8: Example C-structure for QoS description

4.3 Channel establishment for "Compulsory QoS" in an ODP environment

The ISO QoS framework describes several general QoS negotiation mechanisms which can be applied to distributed systems. We will focus on one QoS negotiation mechanism called "Compulsory QoS", and apply this mechanism to an ODP compliant configuration (figure 3). We use the stream interface reference as the means to exchange QoS information between the nodes.

The QoS framework defines three phases for QoS activity [4] which can be applied to ODP systems as follows:

- In the ***prediction phase*** (engineering) objects will make QoS enquiries to predict aspects of system behavior so that the engineering objects can initiate QoS mechanisms appropriately. Enquiries could include, for example, the current loading of the node or previous levels of QoS achieved.
- In the ***establishment phase*** the engineering objects express requirements for QoS (as specified in the computational environment contract), enter into negotiations or re-negotiations, make agreements on the QoS to be delivered and offered. Furthermore, the actions to be performed if QoS degrades, and initiate mechanisms that will be needed during the operational phase.
- In the ***operational phase*** the agreed QoS will be monitored and maintained and if violated, appropriate actions will be taken.

The actions to be performed in the different QoS phases can be applied to the channel establishment between two or more stream interfaces residing on different nodes. Channel establishment in ODP is described in general terms and this section identifies the set of operations specific for the channel establishment for stream interfaces.

Engineering Configuration

The computational multimedia object of figure 1 is represented as:

- a *Basic Engineering Object* (BEO) and,
- associated several engineering objects (e.g., audio/video devices that produce and consume the actual flows) and,
- a *channel half*, i.e., stub, binder and protocol object that deals with the continuous data flows (figure 2).

The correspondence between a computational object with stream interfaces and its engineering equivalent is discussed in [9].

The *node* contains several other engineering objects needed to support distributed processing [3]. The *resource manager* located in the *nucleus* plays an important role for the determination of the load in the node and whether the requested QoS characteristics in the computational environment contract can be honoured.

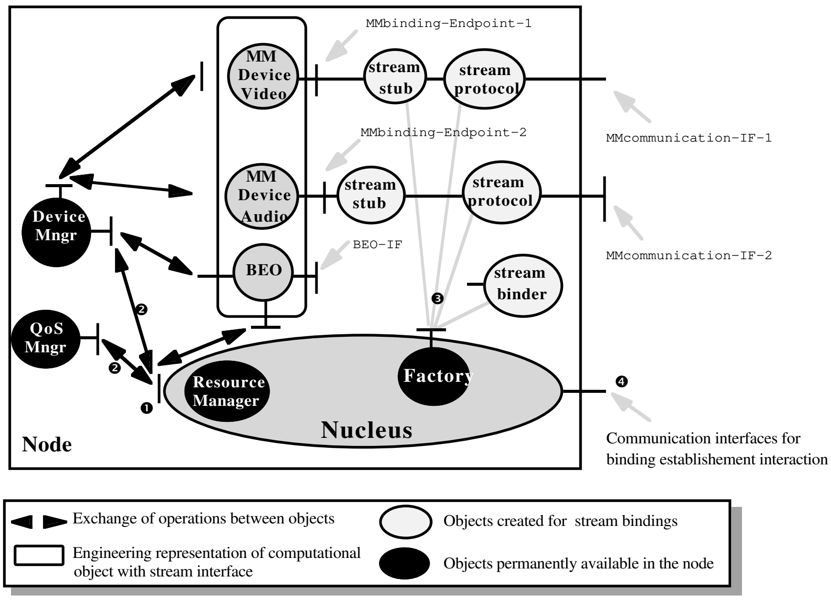


Figure 2: Configuration of engineering objects in a node

Using the configuration of figure 2, a possible scenario for channel establishment between two BEOs residing in different nodes can be as follows:

One of the BEO invokes a `bind()` operation on its local nucleus. The parameters in the bind-operation are (amongst others):

- BEO's interface reference (figure 2, `BEO-If`);
- Interface reference of the remote BEOs (list of remote `BEO-If`)[1];
- List of stream interface references of Multimedia devices associated to the BEO (`MMBinding-Endpoint1, MMBinding-Endpoint2`)[2].

The nuclei (local and remote), that receive the bind request, determine whether the requested binding can be instantiated. Related to the QoS phases [6] they will enter the *prediction* and *establishment QoS* phase. Afterwards, they will return the result of the bind request to the BEOs involved in the stream interface binding.

QoS Prediction phase

Before establishing the connection and instantiating all the objects necessary for the exchange of continuous data flows, a QoS allocation test needs to be performed to determine whether the necessary resources for the binding can be allocated. The allocation test will be performed by the nodes, and underlying networks involved in the binding.

For each node involved, the nucleus will invoke a QoS allocation test operation (figure 2, ❶) on its Resource Manager to determine whether the QoS characteristics can be met. The resource manager has an overview of

---

1We assume in this scenario that the interface references of the remote BEO are obtained via a trader or previous interactions.

2It is also assumed that upon deployment of the Computational MMobject the BEO knows the stream interface references of its associated MMdevices.

the current load of the node, and has the mechanisms to determine whether the requested QoS characteristics as described in the environment contract can be accepted. This test is made in order not to violate established QoS agreements of other processes that are active, or to overload its processor. Interaction between the Device manager and QoS manager will occur to verify the allocation request (figure 2, ❷). The result of the allocation test contains a description of the values of the QoS parameters for which allocation can be guaranteed.

The result of the allocation tests will be analysed either by one of the nucleus or by a third party. The QoS establishment phase is entered when the allocation test has been passed successfully.

QoS Establishment phase

Communication between objects located in different nodes is possible if the necessary objects are instantiated, the QoS between the parties agreed, and the stream interfaces (i.e., `MMcommunication-IF`) are bound.

The `resource-allocation` operation allocates the resources and instantiates the stub, binder and protocol objects in the nodes (figure 2, ❸). For a “compulsory QoS” agreement, the resource manager will allocate the appropriate resources such as the MM devices and a certain processing capacity for the BEO to guarantee 100% availability.

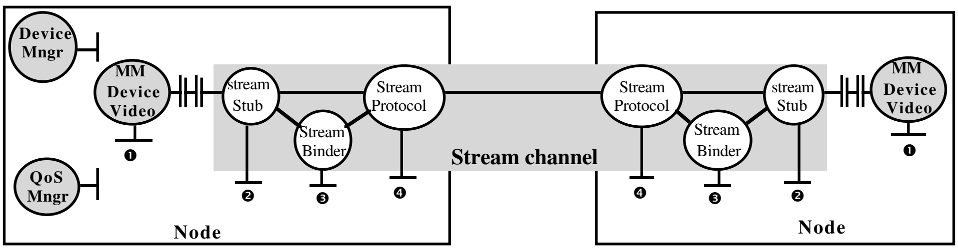
QoS negotiation between nodes (a third party may be involved as well) is possible, and depending on the guarantee level various QoS mechanisms are possible [4]. In a compulsory QoS scenario neither the provider (third party) nor the responding user may decrease the quality as supplied by the initiating user. However, the QoS compulsory value may be increased by the responding user to a value that does not exceed a certain bound.

In an ODP system the initiating user communicates directly to the responding user using the communication interface (figure 2, ❹). In this scenario a third party provider (e.g. network operator) is not directly involved in the negotiation process but indirectly through the protocol object and its associated QoS characteristics that have to be met in a compulsory QoS agreement.

If QoS agreement has been reached the responding nuclei send their `MMcommunication-IF` references to the initiating user (or a third party) that will perform the binding. The `Bind()` operation will establish the binding between all `MMcommunication-IF` interfaces. The parameter `ConnectionGraph` in the `Bind()` operation describes precisely which flows described in the `MMcommunication-IF` need to be inter-connected [11].

QoS operational phase

In this phase QoS monitoring, QoS maintenance and/or QoS enquiry is performed [4]. Figure 3 shows the objects that are involved when a channel has been established between two stream interfaces. Local QoS monitoring can be performed by the *QoS manager* object available in each node. The QoS manager can invoke QoS operations on the various channel objects (figure 3, ❷❸❹) to adjust the QoS values to maintain the QoS agreement. During the operational phase the QoS manager receives periodically (e.g. every 500msec) the actual QoS values from the relevant objects and calculates the required system performance for the negotiated QoS. It calculates the difference between the required system performance and the measured system performance and invokes the appropriate action (e.g. figure 3, ❷❹) to achieve the required system performance. The precise mechanism for QoS tuning are outside the scope of this paper.



❶ create(), dispose(),connect(), disconnect(), start(), stop(), open(), close()
❷ increase-buffer-size(),QoS-change(),no-buffer-space(),data-drop-out(),adjust-compression-rate()
❸ change-channel-configuration(), (+ operations to access the relocator object)
❹ adjust-target-QoS(), (+ operations to access the directory)

Figure 3: QoS operations related to objects in the channel

5. Mapping on ATM QoS parameters

The previous sections related QoS to the ODP framework which resulted in (abstract) models such as figure 3. An interesting issue is how to relate these ODP models and QoS to a certain technology. This section is an initial attempt to apply both the ISO-QoS and ODP standard to ATM. We focus on the stream protocol object and relate the rather abstract QoS characteristics described in [4] to some (more concrete) ATM QoS parameters.

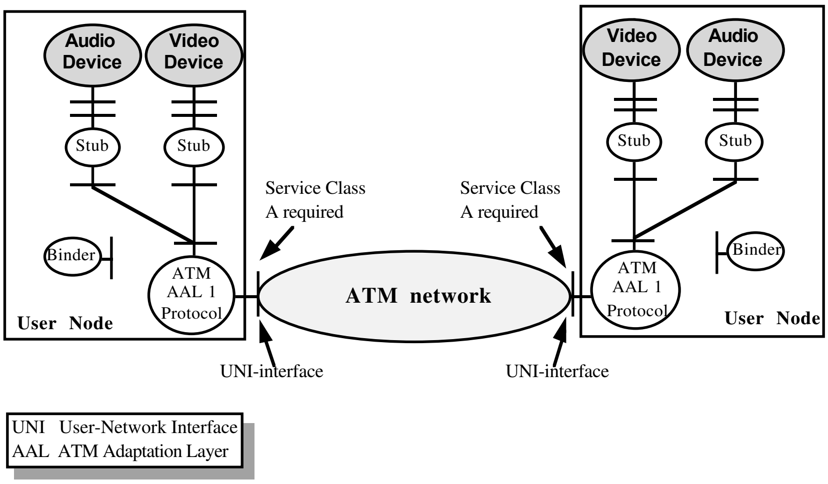


Figure 4: Example of ATM AAL1 protocol object

Figure 4 depicts a partial (ODP) channel showing an end user node supporting the ATM AAL1 protocol. For the support of Constant Bit-Rate (CBR) audio and video, the so-called *Service class A* is the appropriate QoS class [23] [24]. Service class A is connection oriented and a strong timing relation between the source and sink is required but where the bit rate is constant. QoS parameters in this class consist of cell transfer delay, cell delay variation and cell loss ratio. Data passed from the audio and video device in figure 4 is passed to the ATM AAL 1 protocol at the source at fixed intervals and must be passed to the sink at the same rate.

If the user requests for a certain service class (e.g. service class A) this implies that certain ATM QoS parameters are set for the AAL1 protocol (table 9, right column).

QoS characteristics (QoS-framework)	ATM QoS parameters
Time related characteristic (time delay)	Cell interarrival time Cell Delay Variation (delay jitter) Cell Transfer Delay
Capacity related characteristics (communication throughput)	Cell Transfer Capability Mean Cell Rate
Guarantee level	Best effort class (unspecified QoS Class) Statistical/compulsory class (specified QoS class)
Delay	Cell Transfer Delay Mean Cell Transfer delay
Channel availability (the proportion of time that the communications channel is available)	Priority Level (loss priority level) Cell Loss probability
Accuracy (Error related parameters)	Cell Sequence Integrity Cell loss Ratio Cell Error Ratio Cell Misinsertion Rate Severely-Errored Cell Block Ratio Cell-insertion rate (number of cells in a certain time interval which are inserted (from other connections) in a connection. Bit-error probability (in the information field)

Table 9: Translation of QoS parameters onto ATM QoS parameters

Table 9 shows that the more general QoS characteristics (left column) as defined in [4] can be refined for ATM QoS parameters (right column). The ATM parameters apply to cell streams only and are intended to characterise ATM connections that can be set up between two nodes as shown in figure 4.

6. Remarks

This paper discussed ODP computational and engineering issues to integrate quality of service in distributed multimedia environments. These issues are of interest for both industrial consortia (OMG, TINA-C, DAVIC) and standardization groups (ISO, ITU) that need to deal with multimedia applications that have stringent QoS requirements [7], [8], [21].

We used the ISO QoS framework [4] as a basis but due to its general nature we encountered difficulty to relate the (rather) abstract QoS characteristics to concrete QoS parameters as those defined for instance by ATM. We believe that the QoS standardization framework could be improved if translations are made to various network technologies.

The ODP framework provide concepts (e.g. environment contract, stream interface) to specify non-functional characteristics of objects that deal with continuous data flows. Currently the environment contract concept in ODP is not detailed and serves merely as a placeholder for QoS specifications. The ideas presented in this paper will be proposed to the ODP standardization group as input to the ISO/IEC SC21 meeting. In ISO/IEC SC21 two new topics deal with the issues for which this paper provides some solutions:

1. *"QoS and ODP"* [8]. With respect to this question, our paper provides some answers to investigate how the concepts as defined in the QoS framework [4] can be applied to ODP. It also proposes typing rules for stream interfaces. Also OMG is involved in this area and some concepts presented in this paper have been proposed as a white paper to OMG's Telecommunication Special Interest Group [22].
2. *"ODP interface references and binding"* [5]. This work will become a standard that specifies the information content of an interface reference (both stream and operational interfaces). This paper proposes some suggestions how and where to express QoS in the interface reference. It also relates the QoS phases described in the QoS framework [4] to the ODP binding process.

We believe that consistency in terminology, models, and mechanisms for QoS developed in co-operation by de-facto and de-jure standardization is an important step to an efficient and successful way to achieve openness for open distributed multimedia applications.

7. References

[1] Reference Model of Open Distributed Processing, Overview (DIS), ITU/T X.901-ISO 107046-1, 1995.

[2] Reference Model of Open Distributed Processing, Foundations (IS), ITU/T X.902-ISO107046-2, 1995.

[3] Reference Model of Open Distributed Processing, Architecture(IS), ITU/T X.903-ISO 107046-3, 1995.

[4] Quality of Service - Framework, ISO/IEC CD 13236.2, October 95.

[5] Quality of Service - Methods and Mechanism, Working draft #3 ISO/IEC JTC1/SC21/N 9681, November 1995.

[6] "Information Technology-Quality of Service-Guide to Methods and Mechanisms", Proposed Draft Technical Report, ISO/IEC JTC1/SC21/N1022, November 1995.

[7] "ODP interface References and Binding ", Working draft, ISO/IEC JTC 1/SC21 N???, SC21/WG7 Interim Meeting, Kobe, Japan, November 1995.

[8] "A note on issues on QoS and ODP", ISO/IEC JTC 1/SC21, November 1995.

[9] A.T. van Halteren, P. Leydekkers and H.B. Korte, "Specification and Realisation of Stream Interfaces for the TINA-DPE", KPN Research, Proceedings of TINA-95, February 1995.

[10] "TINA Object Definition Language (TINA-ODL) Manual", Document No. TR_NM.002_1.3_95, TINA Consortium, Public, June 1995.

[11] "Network Resource Information Model Specification", No. TB_LR.0012.1_95, TINA Consortium, December 1994.

[12] K. Nahrstedt and R. Steinmetz, "Resource Management in Networked Multimedia Systems", IEEE Computer, May 1995.

[13] A. Vogel, G. v. Bochmann, R. Dssouli, J. Gecsei, A. Hafid and B. Kerheve, "On QoS Negotiations in Distributed Multimedia Applications", technical report TR-891, University of Montreal, 1993.

[14] V. Gay, P. Leydekkers and R. Huis in 't Veld, "Specification of Multiparty Audio and Video Interaction Based on the Reference Model of Open Distributed Processing", Computer Networks and ISDN Systems- Special issue on ODP, Volume 27-Number 8, pp.1247-1262, July 1995.

[15] P. Leydekkers, V. Gay and L.J.N. Franken, "A Computational and Engineering View on Open Distributed Real-time MultiMedia Exchange", In Lecture Notes in Computer Science - Number 1018 - Springer Verlag, ISBN: 3-540-60647-5, 1995.

[16] J.T. Schuilenga, "QoS Monitoring and Modelling of Continuous Streams in a Distributed Environment", M.Sc. thesis KPN Research/ University of Twente, April 1995.
[17] A. Vogel, B. Kerhervé, G.v.Bochmann and J.Gecsei, "Distributed Multimedia and QoS - A Survey-", IEEE Multimedia, Vol2, No2, (Summer 1995), pp.10-19.
[19] "CORBA 2.0/Interoperability - Universal Networked Objects", OMG Document number 95.3.10, March 20, 1995.
[20] Y. Hoffner, "Interoperability and Distributed Platform Design", APM.1507.01, ANSA Phase III, October 1995.
[21] K. Raymond, "Streams and QoS: a White Paper", Contribution to OMG Telecommunications Special Interest Group (TELSIG), December 1995,
[22] P. Leydekkers and M. Mikael Jørgensen, "Stream Interfaces", TINA-C, Contribution to OMG Telecommunications Special Interest Group (TELSIG), December 1995.
[23] P.J. Kuhn, "Quality of Service in ATM Networks", Slides, 1995.
[24] L.G. Cuthbert and J.C. Sapanel, "ATM The Broadband Telecommunications Solution", IEEE Telecommunications Series 29, ISBN 0 85296 815 9, 1993.