Design of a middleware for QoS-aware distribution transparent content delivery

G. Fábián
KPN Research
P.O. Box 421
2260 AK
Leidschendam
The Netherlands
g.fabian@kpn.com

A.T. van Halteren
Twente University
Dept. of Computer Science
P.O.box 217
7500 AE Enschede
The Netherlands
a.t.vanhalteren@utwente.nl

M. van de Logt
KPN Research
P.O. Box 421
2260 AK
Leidschendam
The Netherlands
m.vandelogt@kpn.com

F. Stoinski
Humboldt University
Berlin, Dept. of
Computer Science
Rudower Chaussee 25,
12489 Berlin, Germany
stoinski@informatik.hu-berlin.de

Abstract

Developers of distributed multimedia applications face a diversity of multimedia formats, streaming platforms and streaming protocols. Furthermore, support for end-to-end Quality-of-Service (QoS) is a crucial factor for the development of future distributed multimedia systems. This paper discusses the architecture, design and implementation of a QoS-aware middleware platform for content delivery. The platform supports the development of distributed multimedia applications and can deliver content with QoS guarantees. QoS support is offered by means of an agent infrastructure for QoS negotiation and enforcement. Properties of content are represented using a generic content representation model described using the OMG Meta Object Facility (MOF) model. A content delivery framework manages stream paths for content delivery despite differences in streaming protocols and content encoding. The integration of the QoS support, content representation and content delivery framework results in a QoS-aware middleware that enables representation transparent and location transparent delivery of content.

1. Introduction

The goal of the QUality Aware Middleware for Multimedia Delivery (QUAM MD) platform is to provide a software infrastructure that facilitates the development of distributed multimedia applications. These applications typically integrate off-the-shelf streaming platforms and use existing multimedia stores. One of the design goals of the QUAM MD platform is to provide a software infrastructure that can integrate these existing hardware and software system elements, and can easily accommodate future ones. The complexity of distributed multimedia applications and the shortened time-to-market necessitates applying engineering methods when developing these applications. The QUAM MD platform defines a framework for distributed multimedia applications to achieve faster development and to enhance the quality of distributed multimedia systems. In the rest of this section we present the key technical challenges faced when designing the QUAM MD platform and also, we give an overview of related work.

1.1 Technical challenges of content delivery

Multimedia Representation: Current trends in multimedia production and provision show the increase in the number of multimedia encoding schemes and the volume of available digitised multimedia data. To enable efficient multimedia information access and delivery, a content representation model is necessary to provide format independent content management. Such a model must incorporate present and future media formats and structures easily. For better multimedia data management, a representation model must support access of multimedia data in a distribution transparent way, in a similar way as middleware offers distribution transparencies to client-server objects in distributed object systems.

High-Quality Presentation: Today there is a growing demand for high-quality multimedia applications. The quality of a multimedia presentation is however determined by several factors. Consumer tools vary from powerful high-end desktop computers to handheld devices with limited processing power and display capabilities. Also, the connection bandwidth of consumers varies from low bandwidth mobile connections to dedicated high bandwidth connections. A pre-requisite to high-quality content delivery is that content format is matched to the end-user device capabilities and also to the available connection bandwidth. Furthermore, to provide high quality multimedia presentations, the availability of system resources must be guaranteed. In other words, a guaranteed Quality of Service (QoS) is required from the

underlying software and hardware infrastructure. This leads to the next technical challenge.

Quality of Service Aware Infrastructure: Multimedia applications are typically distributed applications that run in a heterogeneous environment. In such environments, a middleware is used to provide location transparency and to shield applications from the diversity of the underlying software and hardware systems. For multimedia applications, there is a new task; the QoS of the underlying systems must also be controlled. This however should not be done directly by the applications themselves, in order to keep them portable, that is, independent of the low level QoS mechanisms. Today, a broad consensus exists in the research community that QoS provisioning is the task of the middleware. To meet this requirement, several initiatives exist that extend existing middleware platforms, to provide QoS provisioning for distributed applications. Applying this capability for real-time multimedia streaming is one of the goals of the QUAM MD platform.

1.2 Related work

There are several research initiatives for end-to-end QoS support in distributed multimedia systems. This overview concentrates on middleware-based solutions that, next to providing QoS support, also define a component framework for stream establishment and control.

Requirements for a multimedia ORB have been specified in the reTINA project [9]. The CORBA A/V Streaming Service specification [1] defines an architectural model for implementing distributed multimedia streaming applications. This architecture defines the entities that play a role in streaming and a set of interfaces for stream establishment and control. By standardizing these entities, the CORBA A/V Streaming Service has set the goal of supporting multiple data transfer protocols and many types of sources and sinks within a single architecture. Another characteristic of the CORBA A/V Streaming specification is that while controlling signals are exchanged via the ORB's GIOP/IIOP path, data transfer takes place outside of the ORB. This is important to be able to optimise data transfer. MULTE-ORB [6] is another QoS-aware middleware. In MULTE, a binding framework has been developed, which is a composite of distributed objects used to connect multiple interfaces. It defines explicit stream bindings, stream interfaces, and flows. Per binding, stream properties can be specified such as QoS requirements and binding structures. Another QoS-aware middleware capable of multimedia streaming is Quartz [8]. In Quartz, QoS concepts are introduced at system and application levels, with configurable mappings between the two. A hierarchy of QoS agents implements the mapping between the different levels of parameters and resource trading. The CATS [2] offers a platform for content composition and delivery. CATS uses a multimedia metadata to describe the content structure with activation and deactivation conditions. CATS supports path set up and release between communication sources and sinks, and can deliver media according to the content schedule in a QoS-aware way using resource adaptation.

2. Overview of the QUAM MD architecture

The technical challenges of content delivery lead to the following design goals for the QUAM MD platform. The middleware platform should not reveal the physical location of content to a multimedia application. Hiding the physical location of content is called *location transparency*. In addition, the platform should deal with the diversity of the streaming protocols and encoding schemes for content. Hiding these aspects from a distributed multimedia application is called *representation transparency*.

To meet these design goals, a unified information model is needed to represent the properties of multimedia as distributed objects. Furthermore, an object-oriented delivery framework is necessary to integrate the many different hardware and software system elements that are part of today's streaming solutions. The solution should be a QoS-aware middleware platform, where QoS negotiation and establishment concerns content streaming. QoS enforcement mechanisms are hidden from applications, and applications remain portable across different systems.

The QUAM MD platform defines a framework and a set of interfaces that can be used to build QoS-aware distributed multimedia applications. Below we outline the functional components and the engineering considerations of the QUAM MD platform. The QUAM MD architecture reuses elements of the architecture of the QoS Provisioning Service (QPS) [11]. QPS is a CORBA service that provides QoS for remote method invocations between a client and server object. In QPS, client applications can express a required QoS level ($Q_{required}$) concerning a single binding, whereas server objects can express an offered QoS level ($Q_{offered}$). For QPS, we have implemented a QoS negotiation process that takes into account $Q_{required}$, $Q_{offered}$, and the available network and computing resources. When successful, the negotiation results in an agreed QoS level (Q_{agreed}), which is then maintained for the lifecycle of the binding. This architecture has been extended in QUAM MD to establish streams with QoS.

A stream is the transfer of content from a producer device to a consumer device. A stream path is terminated by a source and sink stream endpoint. Management and

signalling operations are conveyed by an ORB, whereas, streams flow outside of the ORB. Stream Management objects in the QUAM MD platform provide stream path set up, QoS enforcement and path management functionalities. The platform contains Content Representation objects as well that are the run-time representations of content properties. Figure 1 shows the functional components of QUAM MD and their global interactions.

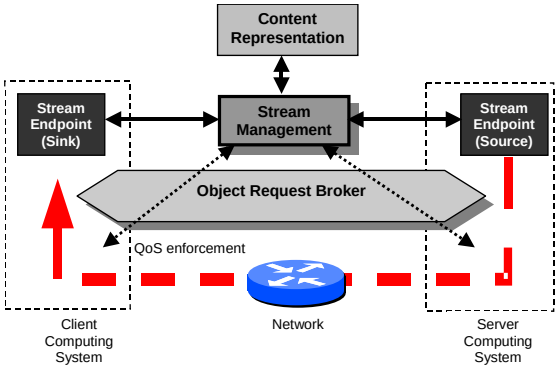

Figure 1: QUAM MD functional architecture

Based on the QoS negotiation concepts of QPS, we have implemented a negotiation process for multimedia content in QUAM MD. Client applications can express their $Q_{required}$ concerning the streaming of content. Again, a matchmaking process follows, which also takes into account that *a)* the same content can be encoded using different encoders, and *b)* different streaming platforms may be used. Successful negotiation results in a Q_{agreed} and a choice for a specific system configuration. When a stream is set up, Q_{agreed} is maintained for the lifetime of the stream. To achieve this, in QUAM MD, we re-use the QoS enforcement mechanisms that were already present for QPS. We use the Resource ReSerVation Protocol (RSVP) [10] to reserve dedicated channels for streaming.

In the rest of the article the QUAM MD platform is explained in more detail. The article is further divided as follows. Section 3 discusses the design of the content representation model. Section 4 presents the QUAM MD support for QoS negotiation and enforcement. In Section 5, the stream management objects are introduced and it is illustrated how these objects collaborate with the content representation and QoS support objects. Finally, in Section 6 we present the conclusions.

3. Models for content

At the core of the QUAM MD design lays a generic information model to describe content. This model uses the OMG Meta-Object Facility (MOF) [4] to express multimedia content descriptions. The MOF is a generic framework to describe and represent meta-data. Meta-data denotes any data that in some sense describes other data. Two mappings of MOF models to external formats have been standardised:

- MOF-IDL-mapping: This mapping generates the IDL-specification for a meta-data service from a MOF-meta-model specification. The resulting service is a repository that can be used to store or manipulate models.
- MOF-XMI (XML based Model Interchange) mapping: This mapping defines rules *a)* to derive an XML Document Type Definition (DTD) from an model and *b)* to represent a model in an XML document that is structured according to this DTD.

The MOF-IDL-mapping enables the automated generation of a meta-data repository that allows CORBA applications to access meta-data at run-time. This mapping has been used for QUAM MD to create a repository that stores content meta-data.

3.1 Content meta-data

This section shows an example of content meta-data. The example concerns a promotional movie to promote the activities of a company. To be able to view the movie with different streaming players, the promotional movie is encoded using QuickTime and Windows Media Format. As a result, two clips are created with the same format parameters, with the only difference being their formats. Each clip is stored on its own content server.

The meta-data of a clip is described using a set of descriptors. A descriptor is a name-value pair that describes a particular aspect of content. An example of a descriptor for a clip is the name `CompressionFormat` with value `QuickTime`, which indicates how the clip has been encoded. Descriptors can be grouped in a description scheme. Such a scheme called `PromoA_Coding` is shown in Figure 2.

```
DescriptionScheme PromoA_Coding realises
MediaCodingType {
     FrameWidth 800;
     FrameHeight 600;
    CompressionFormat QuickTime;
};
```

Figure 2: An example description scheme

A descriptor has a type. For example, in Figure 2 `CompressionFormat` has type `string` and the `FrameHeight` has type `unsigned short`. Descriptor types can be grouped together in a type for a description scheme, which enables type checking of a description scheme. Figure 3 shows the description scheme type `MediaCodingType` that is realised by the description scheme shown in Figure 2.

```
DescriptionSchemeType MediaCodingType{
     unsigned short FrameWidth;
     unsigned short FrameHeight;
      string CompressionFormat;
  };
```

Figure 3: A description scheme type

Different types of description schemes can be used to capture different aspects of the content meta-data. For example, a description scheme for the physical location of the clip can be added. These description schemes can be composed into a new description scheme called `CompanyPromoA` that contains the `PromoA_Coding` and `urlA` schemes shown in Figure 4.

```
DescriptionScheme CompanyPromoA{
 DescriptionScheme PromoA_Coding realises
MediaCodingType{
     FrameWidth 800;
     FrameHeight 600;
    CompressionFormat QuickTime;
    };
 DescriptionScheme urlA realises
MediaLocatorType{
    URL rtsp://serverA/promo.mov;
    };
};
```

Figure 4: A container description scheme

To further represent the alternative locations and encoding of the clips, the container description scheme can be contained in another description scheme called `CompanyPromos` as is shown in Figure 5.

```
DescriptionScheme CompanyPromos{
// 1st alternative
DescriptionScheme CompanyPromoA{
  DescriptionScheme PromoA_Coding{ . . .
};
  DescriptionScheme urlA{ . . . };
};// 2nd alternative
DescriptionScheme CompanyPromoB{
  DescriptionScheme PromoB_Coding realises
MediaCodingType{
     FrameWidth 800;
     FrameHeight 600;
    CompressionFormat WMF;
    };
  DescriptionScheme urlB realises
MediaLocatorType{
    URL mms://serverB/promo.wmv;
    };
 };};
```

Figure 5: A set of alternative description schemes

The nesting of description schemes must be constrained, because some configurations are devoid of logic. For example, it is not allowed to define a descriptor and a description scheme in the same containing description scheme. Therefore, a description scheme can only contain one or more description schemes *or* one or more descriptors.

3.2 The content meta-model

To create and manipulate content meta-data such as presented in the previous section, a meta-model has been developed for content descriptions. The objects instantiated from this model are the content meta-data. The content meta-model is inspired by the MPEG-7 standard [5].

The content meta-model is shown in Figure 6. The modelling entities and their relations as described in the previous section are represented as UML classes and associations. To emphasise that the model is a meta-model the classes have the postfix 'Def'. The model uses the container-contained pattern as found in the CORBA Interface Repository specification, which is a variation on the Composite design pattern [7].

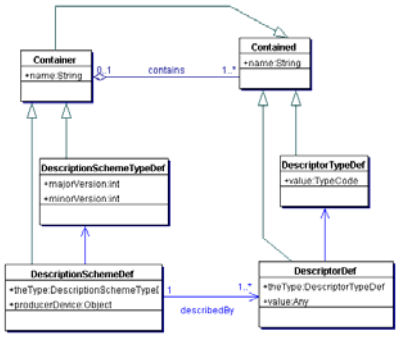

Figure 6: The content meta-model

A `DescriptionSchemeDef` may be contained in another `DescriptionSchemeDef`. As a result a `DescriptionSchemeDef` may have arbitrary levels of containment. However, the level of containment is restricted to three. Such a constraint is expressed using the Object Constraint Language (OCL), and is checked at run-time to prevent inconsistencies. As a result, the content meta-data processed by the QUAM MD platform remains consistent.

Formatted: Bullets and Numbering

4. Content streaming with QoS guarantees

To facilitate content streaming with QoS guarantees, in QUAM MD a QoS control infrastructure has been built. In this section, we describe the QoS dimensions for multimedia, the QoS control infrastructure, and the negotiation model for multimedia.

4.1 QoS Dimensions

The QUAM MD platform concerns the end-user QoS which is the quality of the multimedia presentation that the end-user experiences. These QoS values can be verified by streaming platforms at streaming time. Applications using the QUAM MD platform specify the

required end-user QoS ($Q_{required}$). The offered QoS ($Q_{offered}$) of a clip is determined at encoding time, when a particular codec is selected for encoding with certain parameters. Although, the QoS parameters are somewhat different per streaming platform, in QUAM MD, we have succeeded to settle with a small set of common QoS dimensions. These are shown in Table 1 and Table 2 below.

The QoS dimensions for audio material are audio kind, sample rate and frequency response. The sample rate and the frequency response are much related; therefore, it shall often be the case that only one of these dimensions is specified, probably frequency response. The kind of audio is in fact determined by the raw audio material, which determines the codec choice.

Table 1: Audio QoS Dimensions in QUAM MD

Dimension Name	Values
Audio kind	Voice only, Mono, Stereo, Quad, Dolby
Sample rate	8 kHz, 16 kHz, 32 kHz, 48 kHz (DVD, DAT), 11,025 kHz, 22,5 kHz, 44,1 kHz (CD)
Frequency response	4–10 kHz (Voice only) 5–15 kHz (Mono music or low bandwidth stereo) 15 –20 kHz (Stereo music with high bandwidth) 20-22 kHz (Excellent quality stereo music with high bandwidth)

For video material, the QoS dimensions are motion/clarity factor, the absolute quality value and resolution. The reason for choosing these dimensions is as follows. During encoding information is lost, which is realized either by dropping frames or reducing the amount of information stored per pixel. How this translates to end-user quality depends on the kind of content (frame loss would be less acceptable for a fast motion film than for a more still scenery). The end-user QoS experience would also depend on the techniques used in encoding algorithms to recover lost data. Therefore, by concentrating on the end-user QoS, we chose the two orthogonal QoS dimensions: the motion/clarity factor and an absolute quality value. The three video QoS dimensions are independent of each other.

Table 2: Video QoS Dimensions in QUAM MD

Dimension Names	Values
Motion/quality factor	[0..1] Indicates the preference for better motion (1) or better picture clarity (0).
Absolute quality value	Excellent, Good, Fair, Poor.
Resolution	176x132, 240x180, 320x240, 640x480, 800x600, 1280x720, 1152x900, 1920x1080

QoS specifications are expressed using description schemes. A description scheme that describes the QoS of video material is shown in Figure 7. This description scheme is constructed of two types: `AudioQoSType` and `VideoQoSType`. Description schemes of type `MultimediaQoSType` are stored per clip in the meta-data repository.

```
DescriptionScheme MediaQoS realises
MultimediaQoSType{
 DescriptionScheme AudioQoS realises
AudioQoSType{
   AudioKind Stereo;
   SampleRate 44.1;
   FrequencyResponse 20;
   };
 DescriptionScheme VideoQoS realizes
VideoQoSType{
   MotionClarity 0.5;
   AboluteValue Good;
   FrameWidth 800;
   FrameHeight 600;
   };
};
```

Figure 7: The multimedia QoS description scheme

4.2 QoS Agent Architecture

In QUAM MD, QoS is controlled by a coordinated act of QoS agents. QoS agents represent individual software or hardware entities that play a role in the streaming of multimedia content, and thus influence the end-to-end QoS. A QoS agent that represents such an entity controls the configuration and resource use of the component. The minimum requirement for an entity to be represented by a QoS agent is that it provides a programmable interface for configuration, resource use and status query. In QUAM MD, the following kinds of candidate entities are identified:

- *multimedia devices:* These are hardware devices attached to server or player machines such as network cards, sound cards or graphic cards. It is also possible to consider a complete handheld device as a multimedia device. Furthermore, software devices such as a streaming server or player program instances are also in this category.
- *network resource managers:* These are the known resource manager daemons, such as the RSVP or Differentiated Services (Diffserv) [3] daemons. Network resource managers can reserve a dedicated network channel between devices with a specified bandwidth.

- *processing resource managers:* These are managers that can influence the way a particular process or thread is served by the operating system. Examples are queue managers and schedulers, or other programmable instances that determine the priority of processes.
- *storage devices:* These are devices that may influence the QoS of the multimedia streaming. An example could be a multimedia content storage device, if its throughput can be controlled.

QoS agents are composed hierarchically. On the top of the QoS agent hierarchy is a *master* QoS agent that coordinates and instructs the lower level QoS agents, called the *slave* agents. An arbitrary tree of QoS agents can be built in this way, with a *root* agent on the top. An example can be seen in Figure 8. In this configuration, a desktop computer acts as a multimedia client. QoS agents represent the network card and the streaming player software. One master agent controls these agents. The QoS agent hierarchy is similar on the server side. The network resources are managed by RSVP, and controlled by one network QoS agent. The reason for using only one network agent is that an RSVP channel can be set up between client and server by communicating with the RSVP daemon only on the server side. The root agent controls the client and the server master agents and the network agent.

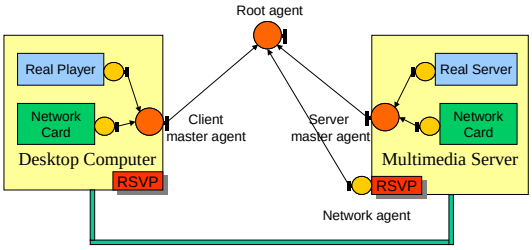

Figure 8: Example QoS agent hierarchy

4.3 QoS negotiation and enforcement

QoS negotiation takes place during stream path set up which is coordinated by the stream management objects as will be explained later in Section 5. During this operation, the task of the QoS agents is to carry out QoS negotiation and reserve resources. The root QoS agent coordinates all these activities.

The input for a QoS negotiation process is a description scheme that may contain several alternative representations of the same content and $Q_{required}$. The negotiation takes place in two steps. First, the `selectContent` operation is called on the root agent. During this operation, the root QoS agent selects those content representations that satisfy $Q_{required}$. This is determined by comparing the values of $Q_{required}$ and $Q_{offered}$ for each QoS dimension. If the value of each QoS dimension is equal or higher ("better") than that of $Q_{required}$, then a representation satisfies $Q_{required}$. The only exception to this is the motion/clarity factor, where value ranges have been introduced. If $Q_{required}$ and $Q_{offered}$ are in the same range, then $Q_{offered}$ satisfies $Q_{required}$.

The second step of the QoS negotiation is the `negotiateQoS` operation. During this operation the root agent gathers information from its slave agents about the availability of system resources. Then, the root agent chooses one content representation and one specific system configuration. In QUAM MD we implement the simple policy of choosing the representation that uses the least resources. This step may use different policies however to determine which content representation is streamed eventually. At the end of a successful negotiation, when a system configuration is determined, the `makeReservations` operation is called and subsequently, resource reservation takes place in a distributed fashion. Each master agent calculates the resources its slave agents must reserve, performs necessary parameter translation and delegates the reservation to its slave agents.

For a content representation selected by the `negotiateQoS` operation, in the next step, all resources are reserved that have been assumed during the encoding of this clip. For example, for a video clip that has been encoded with a 150 Kbps codec, 150 Kbps bandwidth is reserved.

5. Content Delivery Framework

The QUAM MD framework is an object infrastructure to provide the basic functionalities to set up and manage streams using off-the-shelf streaming platforms. This framework shields application developers from the diversity of streaming platforms and streaming protocols. Together with the content representation framework it also provides representation and location transparent content access. The framework uses the QoS agent hierarchy to set up streams with QoS guarantees. This section discusses the design of the content delivery framework and the stream path set up operation.

5.1 Content Delivery Stream Path

We define the content delivery stream path as the path from a content producer device to a content consumer device. The end-to-end path can be composed of several content stream paths as shown in Figure 9.

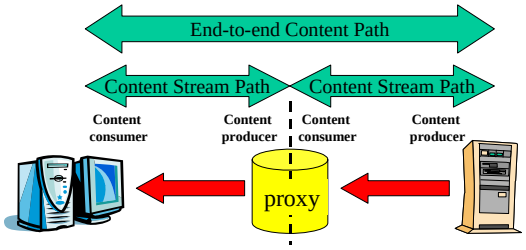

Figure 9: Composed Content Delivery Paths

When setting up a stream path the QUAM MD platform determines which streaming platforms are available on the consumer and on the producer side and selects one to be used for streaming. By abstracting from specific streaming platforms, it is possible to define a uniform control interface for stream paths. This interface provides control functions like suspend, resume and cancel for individual streams. Individual stream path controls can be composed to create an end-to-end stream path control. By doing so, the QUAM MD platform can control and manage the complete stream path.

The QUAM MD framework defines the objects to create, control and manage stream paths. Figure 10 shows an example of stream path objects. In the rest of this section we describe in detail the QUAM MD objects.

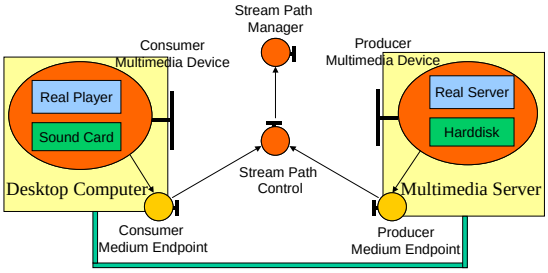

Figure 10: QUAM MD Delivery Framework Stream Representation

The Stream Path Manager is a factory for Stream Path Controls. When an application wants to set up a new stream path, it calls the `createStreamPathControl` operation which is explained further in Section 5.2. The Stream Path Manager uses the Stream Path Control objects to control and manage individual streams. The Stream Path Manager uses the root QoS agent, introduced earlier in Section 4.2, for QoS support operations such as resource reservation. The root QoS agent is associated with the Stream Path Manager using the `setQoSAgent` operation.

Multimedia Devices are representations of actual physical devices. In Figure 10 the producer and consumer Multimedia Devices represent the specific hardware and software components of a multimedia server and desktop computer respectively. They describe device capabilities such as the supported content formats and display size. These capabilities can be queried using the `canSupport` operation of the Multimedia Device interface.

Medium Endpoints represent Multimedia Devices for one specific stream path. They are created by Multimedia Devices when the `createMediumEndpoint` operation is called. Medium Endpoint objects provide real-time stream related information, like the number of frames dropped by a content player or the packets lost when receiving the content.

Stream Path Control objects control individual streams. When created, they query the Multimedia Devices that have to be connected and instruct them to create Medium Endpoints. When a stream is established, the Stream Path Control enables run-time manipulation of the stream and provides monitoring information.

5.2 Stream Path Set Up

The stream path set up is initiated by calling the `createStreamPathControl` operation on the Stream Path Manager as shown in Figure 11. The first parameter of this operation is a description scheme that contains the alternative content representations (see Figure 5) of the same content. The second parameter is the consumer Multimedia Device. The third parameter is the required QoS, also in the form of a description scheme, as explained earlier in Section 4.1.

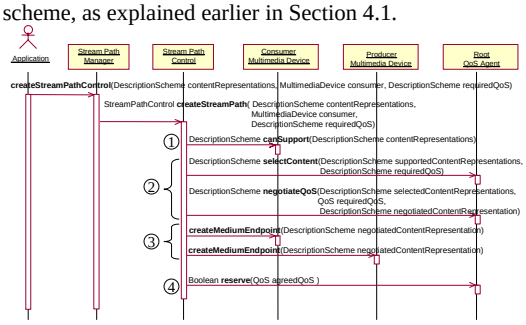

Figure 11: Stream Path Set Up

The remainder of this section discusses the four steps to set up a stream path, as shown in Figure 11.

The first step is the *Multimedia Device Capability Check.* For each content representation in the description scheme, the consumer Multimedia Device is checked whether it is capable of playing back that content. This is done using the `canSupport` operation on the consumer Multimedia Device. This operation returns a DescriptionScheme that contains those content representations that can be played on the consumer Multimedia Device.

The second step is the *Content Representation Selection.* First, the `selectContent` operation is called

on the root QoS agent (see Section 4.3) that selects the content representations that satisfy the required QoS. Second, the `negotiateQoS` operation (see Section 4.3) is called on the root QoS agent. The root QoS agent now determines which content representation will be streamed and the system configuration for the streaming.

Initialisation, the third step, comprises of creation and configuration of the Medium Endpoints for the stream path. The Stream Path Control calls the `createMediumEndpoint` operation on the consumer Multimedia Device and the producer Multimedia Device, which results in a consumer and producer Medium Endpoints that are configured for delivery of presentation of the selected content representation.

Resource Reservation, the final step, is initiated when the Stream Path Control calls `makeReservation`(see Section 4.3) on the RootQoSAgent. The RootQoSAgent reserves the resources needed to deliver the selected content representation with the agreed QoS. Should the reservation fail, the stream path set up continues with step two, removing the initially selected content representation. When no more content representations are available, the stream path set up fails.

QUAM MD also allows stream paths to be set up without QoS enforcement, resulting in a stream path based on a best effort service. In this case step four of the stream path set up is skipped. Also, the `negotiateQoS` operation executes an alternative implementation, with an empty required QoS.

6. Conclusions

In this paper we have presented the QUAM MD platform that is a middleware-based software infrastructure for developing QoS-aware distributed multimedia applications.

In QUAM MD a meta-data based information model has been developed for content representation. A CORBA IDL run-time representation of this information model enables other CORBA objects to access content representation data. As a result the QUAM MD platform offers representation and location transparencies to the developers of distributed multimedia applications.

Similarly to the CORBA A/V Streaming specification, QUAM MD defines an object infrastructure for multimedia delivery. This architecture abstracts from specific streaming platforms and provides an integrated central approach for setting up streams, and for control and management of content delivery. Even in a best effort service environment, the delivery architecture provides substantial benefit by hiding from application developers the heterogeneity of multimedia formats, streaming protocols and streaming platforms in a heterogeneous environment.

The architecture of the platform is based on QoS Provisioning Service (QPS), developed earlier for QoS provisioning of object invocations. The QoS mechanisms available for QPS have been re-used to support QoS for multimedia streaming. This middleware-based approach of QUAM MD ensures that low-level QoS enforcement mechanisms are hidden from applications. This reduces development time and results in more portable applications.

Future work on the platform includes an import utility for MPEG-7 descriptions. Such a utility could benefit from XSLT technology to convert MPEG-7 documents to an XMI representation that complies with our content meta-model. The result of this conversion can then be imported directly into the QUAM MD meta-data repository. Furthermore, the platform should be extended with support for security. We are currently investigating the use of IPSec to create a secure stream path. The QoS agents can be employed to negotiate and enforce such a secure stream path.

7. References

[1] Object Management Group, *Control and Management of A/V Streams specification*. OMG Document telecom/97-05-07 edition.

[2] O. Kath, F. Stoinski, W. Takita, Y. Tsuchiya, "*Middleware Platform Support for Multimedia Content Provision*", Proceedings of EURESCOM Summit 2001: 3G Technologies and Applications, Heidelberg 2001.

[3] M. Carlson, W. Weiss, S. Blake, Z. Wang, D. Black, and E. Davies, *An Architecture for Differentiated Services*, IETF document. RFC 2475, December 1998.

[4] Object Management Group, *Meta Object Facility, Version 1.3*, OMG document ad/99-07-03.

[5] P. Salembier and J.R. Smith, *MPEG-7 multimedia description schemes*, IEEE Transactions on circuits and systems for video technology, Vol. 11, No. 6, June 2001.

[6] T. Plagemann, F. Eliassen, B. Hafskjold, T. Kristensen, R.H. Macdonald, and H.O. Rafaelsen, *Flexible and Extensible QoS Management for Adaptable Middleware*, in Proceedings of International Workshop on Protocols for Multimedia Systems (PROMS 2000), Cracow, Poland, October 2000.

[7] E. Gamma, R. Helm, R. Johnson and J. Vlissides *Design Patters: Elements of Reusable Object-Oriented Software.* Addison-Wesley, Reading, MA. 1995.

[8] F. Siqueira and V. Cahill, *Quartz: A QoS Architecture for Open Systems*, 20th International Conference on Distributed Computing Systems (ICDCS'00), Taipei, Taiwan, April 2000.

[9] Chorus Systems, *Requirements for a Real-Time ORB*, ReTINA, Tech. Report RT/TR-96-8, May 1996.

[10] M. Karsten, J. Schmitt, and R. Steinmetz, *Implementation and Evaluation of the KOM RSVP Engine*, IEEE InfoCom 2001.

[11] A.T. van Halteren, G. Fábián and E. Groeneveld. *Design and Evaluation of a QoS Provisioning Service (QPS)* In Proceedings of Third IFIP International Working Conference on Distributed Applications and Interoperable Systems (DAIS 2001) Krakow, Poland, 2001, pp. 189-200.