

Issues for Distributed Multimedia System

Essay in course “DIF8914 Distributed Information System”

Gu Mingyang, November, 2002

IDI, Norwegian University of Science and Technology, 7491 Trondheim, NORWAY

mingyang.Gu@idi.ntnu.no

Abstract

Multimedia applications generate and consume continuous streams of data in real time. They contain large quantities of audio, video and other time-based data elements, and the timely processing and delivery of the individual data elements is essential. In distributed system, data transmission is pre-requisite. So the main topic in distributed multimedia system is how to transfer multimedia data within the demanded quality. This paper discusses the requirements imposed by multimedia computing, and then provides two framework models to meet some of these requirements.

1 Introduction

Multimedia applications generate and consume continuous streams of data in real time. They contain large quantities of audio, video and other time-based data elements, and the timely processing and delivery of the individual data elements is essential. In distributed system, data transmission is pre-requisite. So the main topic in distributed multimedia system is how to transfer multimedia data within the demanded quality.

The existing standards and platforms about the distributed system, such as RM-ODP, CORBA and DCE, mainly focus on the discrete data transmission. The introduction of multimedia computing puts a large number of new requirements on distributed system.

Firstly, the distributed multimedia system should be able to provide support for continuous media types, such as audio, video and animation. The introduction of such continuous media data to distributed systems demands the need for continuous data transfers over relatively long periods of time. For example, playing a video from a remote website implies that the timeliness of such media transmission must be maintained in the course of the continuous media presentation.

The second requirement of distributed multimedia applications is the need for sophisticated quality of service (QoS) management. In most traditional computing environments, requests for a particular service are either met or ignored. But in multimedia system, there are more contents, which can be classified into static QoS management and dynamic QoS management.

Another requirement of distributed multimedia applications is the need for a rich set of real-time synchronization mechanisms about continuous media transmission. Such real-time synchronizations can be divided into two categories: intra-media synchronization and inter-media synchronization.

A further requirement is to support multiparty communications. Many distributed multimedia applications are concerned with interactions between dispersed groups of users, for example, a remote conference application. So it is important for distributed multimedia system to support multiparty communication.

In order to meet such requirement, some new frameworks appear. In this paper, we will introduce two framework models.

In the first model, we model the stream as binding object. And this model will establish a middleware platform to resolve user-oriented QoS using such binding objects.

The second model is based on the idea that the new differentiated or integrated services respond to the needs of new applications, which means the framework provides a number of services and satisfies the requirements of different applications through selecting or integrating these services. This framework provides a guaranteed end-to-end QoS in an IPv6 differentiated services environment.

The essay is structured as follows. Section 2 presents some terms mainly about the multimedia definition. Section 3 describes the four requirements in detail imposed by multimedia computing. Section 4 introduces two framework models trying to meet such requirements. Conclusions are presented in section 5.

2 Terminology and Related Topics

In this section, two concepts, such as media, multimedia, will be introduced. And then the characteristics of multimedia will be described.

Definition Media:

The term media refers to the storage, transmission, interchange, presentation, representation and perception of different information type (data types) such as text, graphics, voice, audio and video.

According to different application goals, the definition emphasizes different aspects of media. In the case of the definition of multimedia, the representation media is focused on.

Definition Multimedia:

The term multimedia is used to denote the property of handling a variety of representation media in an integrated manner.

Representation media is related to how information is described (represented) in an abstract form, for use within an electronic system. For example, for the data of text, we can present it to user using ASCII characters, grey-scale graphics or colorful graphics. In this example, different representation types are used to present the same content.

In order to store or transfer multimedia data, two types of media ought to be classified: continuous media and discrete media. Continuous media types are those with an implied temporal dimension: items of data must be presented according to particular real-time constraints for a particular length of time. For example, audio, video and animation belong to continuous media type. On the contrary, the discrete media types have no relation to time limitation. Examples of discrete media types are text and graphic.

It is reasonable to fully integrate the discrete media types and continuous media types, but to support the representation of continuous media type will need considerable demands on the underlying technologies. Continuous media types may be represented in either digital or analogue format. In this paper, all the discussion will focus on the digital continuous media.

A distributed system is designed to support the development of applications and services which have a physical architecture consisting of multiple, autonomous processing elements that do not share primary memory but cooperate by sending asynchronous messages over a communication network.

There are some standards and platforms developed to support distributed system on traditional data type. But handling multimedia data on distributed system throw considerable requirements on the design and development of distributed system.

3 Requirements

The introduction of multimedia adds some significant requirements to the developers of distributed system platforms. We can discuss such requirements in four categories:

3.1 Support for continuous media

The first requirement of multimedia is the need to provide support for continuous media types, such as audio, video and animation. The introduction of such continuous media data to distributed systems demand the need for continuous data transfers over relatively long periods of time. For example, playing a video from a remote website implies that the timeliness of such media transmission must be maintained in the course of the continuous media presentation.

Simple streams and complex streams

We can see the continuous multimedia as stream when transferring through the distributed system. On closer examination, stream interaction is a general concept covering a number of different styles. It is possible to identify two broad classes of stream interaction.

- *Simple streams*

A simple stream consists of a single flow of data where the data is of a single continuous media type, such as a single flow of audio or video data.

- *Complex streams*

A complex stream consists of several flows of data where each flow has a designated and potentially distinct media type. For example, a complex stream could consist of an audio flow and a video flow or two separate audio flows.

Complex streams introduce more complexity into the distributed systems than simple streams. Of course, complex streams can be constructed from individual streams. And the individual flows of a complex stream can be transmitted down separate connections. But as a whole, some more complex works must be considered, such as separating and integrating individual flows at the ends of transmission.

Programming models and system supports for continuous media

Existing programming models and system platforms for distributed computing are normally based on discrete interactions:

- *Asynchronous or synchronous message passing*
- *Remote procedure calls*

- *Object invocation*

The first two models are associated with client-server computing and the latter paradigm with object-oriented models. However, they do not fit for handling or transferring continuous media which will last a long period and have concrete temporal demands. We can therefore identify a need to provide explicit programming models and system supports for continuous media computing.

3.2 Quality of service management

The second requirement of distributed multimedia applications is the need for sophisticated quality of service management. In most traditional computing environments, requests for a particular service are either met or ignored. But in multimedia system, there are more contents.

Quality of service management encompasses a number of different functions, and we can classify them into two types of aspects: static aspects and dynamic aspects.

Static aspects

Static QoS management functions are carried out when a given service is initially established. The goal of these functions is to ensure that the appropriate steps are taken to attain the desired quality of service.

- *QoS specification*

The QoS specification refers to the creation of the QoS contract using an appropriate means to express the QoS requirements. For example, the QoS contract could state the concrete demands on different measurement dimensions such as the timeliness, volume and reliability.

- *QoS negotiation*

The QoS negotiation refers to achieving an agreement on the QoS contract between all involved parties. This function will focus on establishing the concrete quality of service for each of the involved components and ensuring that the whole quality of service can satisfy the acceptable bounds defined in the contract.

- *Admission control*

In order to ensure whether or not the system can provide desired QoS, we usually carry out an admission control test. The admission control test will determine if the system can deliver the required service at that precise moment. If the test is passed, the system will then guarantee that the quality of service can be met. The admission test puts some concrete demands on different resources such as memory and the network, so it is necessary to combine the admission test with resource reservation.

- *Resource reservation*

The resource reservation is used to guarantee the desired service level to be met by reserving resources to that concrete service, such as network, memory and processor.

Dynamic aspects

Dynamic QoS management functions are used to monitor and control the run-time quality of service. The goal of these functions is to ensure that the appropriate steps are taken to maintain the desired quality of service.

- *QoS monitoring*

This function is used to monitor the level of service being offered by the involved components and report any problems. Here, the user should specify the granularity of the monitoring, for example, how often should the monitoring function execute, every second or every hour?

- *QoS policing*

Besides monitoring the system components to maintain the level of service, we should ensure that the users of the service are adhering to the contract. This function is carried out by QoS policing. For example, the QoS contract demands the users to send videos 30 frames per second in a communication. It is important to ensure that the sending speeds are within the limitation.

- *QoS maintenance*

QoS maintenance is concerned with actions that can be taken to ensure that the level of service is maintained in a concrete process of service. For example, if a decline of QoS is detected, the service can ask for more resources from the system in order to keep the level of service. Usually, such actions are enough to deal with minor fluctuations of quality of service. If the contract of service is clearly broken, we will have to use QoS renegotiation function.

- *QoS renegotiation*

If the contract of service is apparently broken down, it becomes necessary to notice the user of the service and to start a renegotiation of the quality of service. The user at this stage may decide to make a new contract or abort this service.

3.3 Real-time synchronization

A further requirement of distributed multimedia applications is the need for a rich set of real-time synchronization mechanisms about continuous media transmission. Such real-time synchronization can be divided into two categories: intra-media synchronization and inter-media synchronization.

Intra-media synchronization

Intra-media synchronization refers to the maintenance of real-time constraints across a single continuous media connection. For example, in video transmission, this type of synchronization is used to ensure the video is received with required throughput, jitter and latency. We can use figure 1 to illustrate these terms.

Throughput of a continuous media transmission is decided by the value of average interval between frames which indicate the frames transmission speed. Jitter refers to the difference between an individual interval and the average interval. Latency refers to the time between the sending and the receiving.

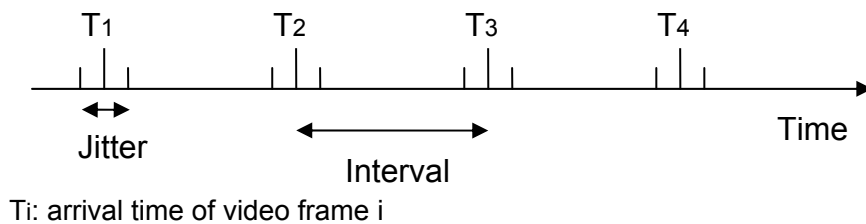


Figure 1. Intra-media synchronization

Inter-media synchronization

Inter-media synchronization is more complex and concerned with arbitrary different media types. We can discuss some examples here which include the synchronization between audio and video channel and the synchronization between text subtitle and video sequences. The first example illustrates an inter-media synchronization between two continuous media types, and the last one is between a continuous media type and a discrete media type.

3.4 Multiparty communications

Many distributed multimedia applications are concerned with interactions between dispersed groups of users, for example, a remote conference application. So the final requirement of the distributed multimedia system is the need to support multiparty communication.

Programming models and system supports

In order to support multiparty communications, we need new programming models. Such models should support different styles of multicasts such as $1 \rightarrow N$, $N \rightarrow 1$, $M \rightarrow N$. In addition, such models should provide some functions to manage the meeting groups including the functions to create and destroy groups, join and leave groups. Also, it is necessary to provide underlying system support for multiparty communications. For example, without system support, the demanded bandwidth will be excessive. We can lessen the bandwidth through constructing the multicast graphs through splitting and merging functions which definitely need the system supports.

Impact on QoS management

In multicast communications, different receivers may require different qualities of service, so it adds considerably complexity to quality of service management. We can see it more clearly through figure 2.

Impact on synchronization

Multiparty communication also adds complexity to synchronization in general. It is important to be able to support a variety of policies for ordering of data delivery, for example, real-time ordering, causal ordering, attribute ordering and partial ordering.

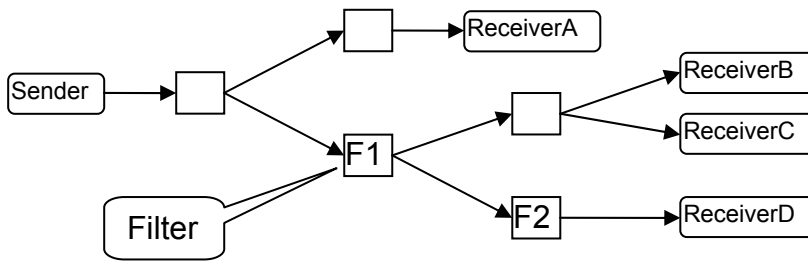


Figure 2. Multicast communication

Sender sends video at 30 frames per second (full color)

ReceiverA receives video at 30 frames per second (full color)

ReceiverB and ReceiverC receives video at 10 frames per second (full color)

ReceiverD receives video at 10 frames per second (grey-scale)

4 Framework Models

There are some standards and platforms for distributed systems such as ISO's Reference Model for Open Distributed Processing (RM-ODP), OMG's Common Object Request Broker Architecture (CORBA), and Open Group's Distributed Computing Environment (DCE) and so on. But almost all of the standards and platforms are designed for discrete data transmissions, and they do not fit for distributed multimedia system, especially for continuous media transmission. Here, I will introduce two framework models, which try to resolve some requirements discussed above.

4.1 A QoS Framework for Streaming

In distributed multimedia system, the multimedia data is transferred as stream. The designers model a stream as binding object. In this framework, we focus on establishing a middleware platform to resolve user-oriented QoS using such binding objects.

Object Model

Figure 3 shows the object model we use to structure the framework. Each object encapsulates state and behavior and can expose operational and streaming interfaces to other objects.

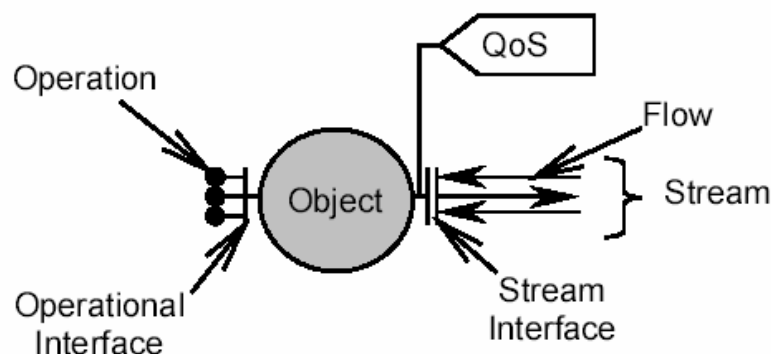


Figure 3. An object with operational and streaming interfaces

Operational interfaces allow client objects to invoke computational services onto the object that exposes the interface (acting as a server object). Streaming interfaces allow objects to exchange one or more flows of continuous information (audio or video information).

QoS-aware Middleware Platform

We propose a platform that is capable to associate a QoS with an object's streaming interface and to control this QoS through the same or another object's operational interface.

The application objects, such as cameras, speakers, files, are endpoints of audio and video stream. The application layer contains agent objects that act as a service object for the end-user. Agents invoke the services of our platform to bind endpoint objects and to control the QoS of local endpoint objects. A binding object allows endpoint objects in the application layer to exchange a stream of multimedia information.

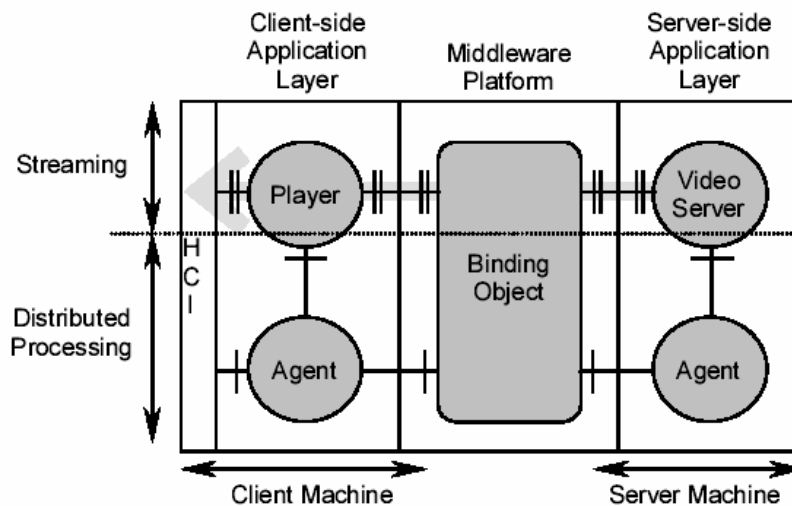


Figure 4. Binding object interconnecting two application objects

Figure 4 show an example, a player user sees the video files stored in the remote video database. The video server produces an audio-video stream that flows to the player object via the binding object. The player user consumes the stream that the binding produces. The two agent objects use the operational interfaces of the player, the video server and the binding object to control the QoS of the streams that these objects produce.

The Platform's Internals

The binding object of Figure 4 may encapsulate a large number and a variety of resources and it needs to perform complex QoS control activities. We will further decompose the platform into two horizontal parts (client and server) and three vertical planes (data transfer plane, QoS control plane and QoS management plane), which is shown in figure 5. Across these three planes we distinguish between a middleware layer and the Distributed Resource Platform (DRP) layer.

- *Data Transfer*

The data transfer plane includes the objects that are capable of forwarding the data units of a multimedia stream.

An object in the data transfer plane of the middleware layer encapsulates resources that perform transport-independent as well as transport dependent stream processing. Examples

of the former are encoders and multiplexers; an example of the latter is a packager that can adapt an MPEG encoded stream for transmission over UDP.

The objects in the data transfer plane of the DRP encapsulate the distributed resources (IP routers, bridges) that provide end-to-end connectivity.

- *QoS Control and Management*

The objects in the QoS control and management planes of the middleware layer and the DRP layer govern the QoS of the stream flowing through the data transfer plane.

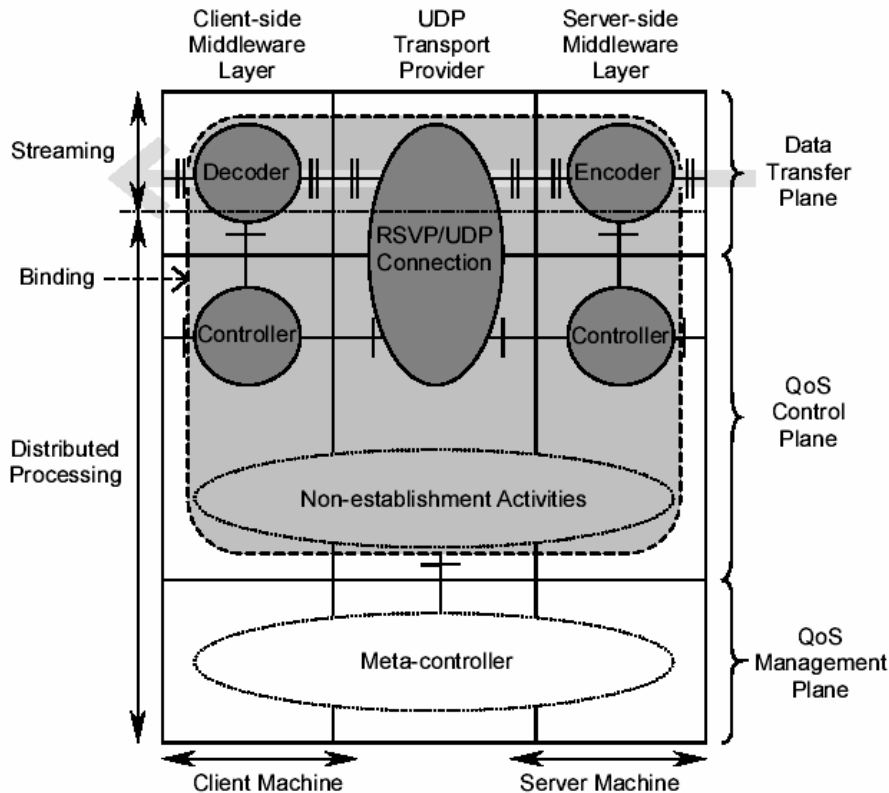


Figure 5. Two-dimensional version of the QoS framework

Each binding object encapsulates a set of objects in the QoS control plane of the middleware layer and the DRP layer. These objects control the QoS of the binding's streaming interfaces during its lifetime. We propose that objects in the QoS control plane are responsible for establishing a QoS for a binding. The establishment of QoS typically involves the negotiation of an acceptable QoS followed by the reservation and initialization of objects in the data transfer plane to fulfill the negotiated QoS. Other activities that can be found in the QoS control plane involve predicting a binding's current and near-future QoS, keeping a binding's current QoS in line with the negotiated QoS, and releasing a binding and its resources.

The objects in the QoS management plane of the middleware layer and the DRP layer are not part of a particular binding. Rather, they can be considered part of every binding because their activities transcend the lifetime of individual bindings. Objects in the QoS management plane for instance take care of fault management and statistics collection.

You can see this model more detailed in [7].

4.2 A Framework Model Based on IPv6 Environment

The main idea in this model is that the new differentiated or integrated services respond to the needs of new applications, which means the framework provides a number of services and satisfies the different requirements of different applications through selecting or integrating these services. This framework provides a guaranteed end-to-end QoS in an IPv6 differentiated services environment.

End-to-end level

In this framework, the communications exchanged within a distributed system can be decomposed into several data flows each one requiring its own specific QoS via a consistent API (Application Programming Interface) offering parameters and primitives for a diverse set of necessary services. In figure 6, the application layer software is allowed to establish one or many end-to-end communication channel, each channel can:

- Unicast or multicast
- Dedicated to the transfer of a single flow of application data
- Able to offer a specific QoS

Besides the API, three other modules are defined in this framework:

- The first one provides multiple transport layer possibilities, such as TCP, UDP;
- The second one implements the mechanisms linked to the utilization of QoS services at the IP layer;
- The third one associate a given transport channel with a given IP QoS service.

Then the transport layer function calls are translated to the new API and the detailed requirements about the end-to-end QoS are translated to generic parameters understood by the API. Finally, in addition to QoS parameters, an application must specify four service parameters:

- The first one characterizes the traffic generated by the application sender;
- The second one designates which transport protocol to use (UDP, TCP);
- The third one designates the IP layer's QoS management desired by the application;
- The final parameter identifies the address, either unicast or multicast, of a set of destination applications.

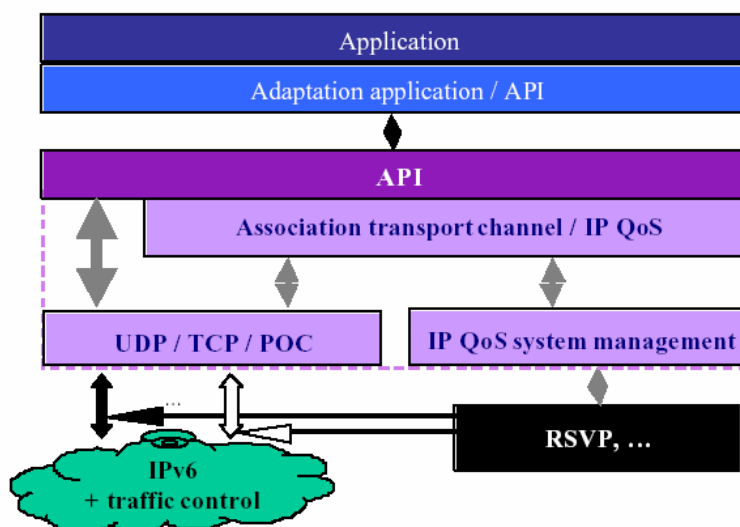


Figure 6. The end-to-end communication system

Network level

QoS functions performed at the network level can be divided in two categories: those related to the data path and those related to the control path:

- On the data path, QoS functions are applied by routers at the packet level in order to provide different levels of service.
- On the control path, QoS functions concern routers configuration and act to enforce the QoS provided.

Three services have been defined at the IP level:

- GS (Guaranteed Service) is used for data flows having strong constraints in both delay and reliability;
- AS (Assured Service) is appropriate for responsive flows having no strong constraints in terms of delay, but requiring a minimum average bandwidth;
- BE (Best Effort) service offers no QoS guarantees.

Input interface of edge router

This is the first interface encountered by a packet when it enters the network. Its logical structure is shown in Figure 7.

This interface is in charge of:

- Packets classification, which is based on information from the IPv6 header;
- Measuring AS and GS flows to determine whether they are in or out of profile;
- Shaping GS packets and dropping them if necessary;
- Marking AS and GS packets with the appropriate Different service code point (DSCP);
- Marking AS packets with the precedence due to their being in or out profile;
- Marking BE packets to prevent them from entering the network with the DSCP of another service class.

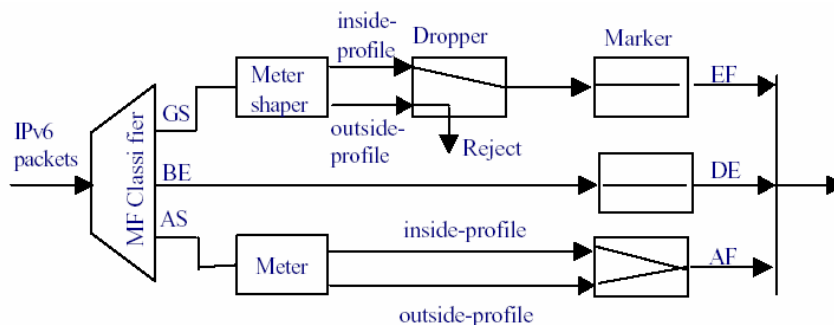


Figure 7. Input interface structure

Output interface of all routers

In this model, all routers must implement a set of forwarding behaviors called Per Hop Behavior (PHB). These behaviors are implemented through scheduling. They are integrated in the output interface of each router (Figure 8).

Two additional points must be mentioned: the Behavior Aggregate classifier which classifies packets according to their DSCP, and the rate control at the output of core routers that is necessary to avoid congestion.

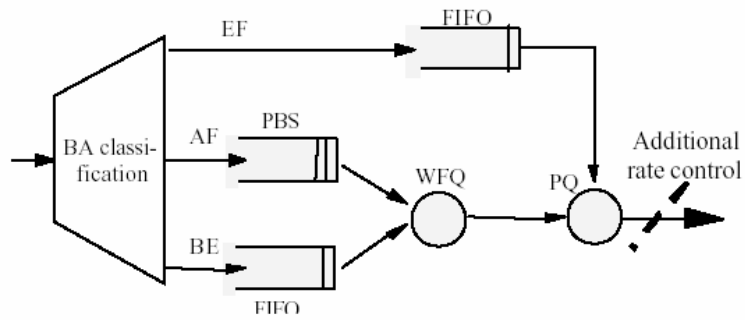


Figure 8. Output interface structure

FIFO: First-in, First-out queue

PBS: Partial Buffer Sharing

WFQ: Weighted Fair Queuing

PQ: Priority Queuing

You can see this model more detailed in [2].

5 Conclusions

This essay has examined the requirements of distributed multimedia system in some depth. Particularly, the following areas have been looked at in detail: support for continuous media, quality of service management, real-time synchronization and multiparty communication. This essay also introduces two framework models trying to solve such requirements. To be frank, these frameworks are just for testing. In order to fully meet such requirements, we should develop a new architecture and some new technologies.

References

- [1]. “Design and Implementation of a QoS-Aware Replication Mechanism for a Distributed Multimedia System Giwon On”, **Jens Schmitt, Ralf Steinmetz**, Lecture Notes in Computer Science _ Interactive Distributed Multimedia System,, 2001, Vol. 2158, Pp. 38-49
- [2]. “Conception, Implementation, and Evaluation of a QoS-Based Architecture for an IP Environment Supporting Differentiated Services”, **Fabien Garcia, Christophe Chassot, André Lozes, Michel Diaz, Pascal Anelli, Emmanuel Lochin**, Lecture Notes in Computer Science _ Interactive Distributed Multimedia System, 2001, Vol. 2158, Pp. 86-98
- [3]. “A QoS-Control Architecture for Object Middleware”, **Lodewijk Bergmans, Aart van Halteren, Luís Ferreira Pires, Marten van Sinderen, Mehmet Aksit**, Lecture Notes in Computer Science _ Interactive Distributed Multimedia System,, 2000, Vol. 1905, Pp. 117-131
- [4]. “QoS Management Middleware: A Separable, Reusable Solution”, **Denise J. Ecklund, Vera Goebel, Thomas Plagemann, Earl F. Ecklund Jr., Carsten Griwodz, Jan Øyvind Aagedal, Ketil Lund, Arne-Jørgen Berre**, Lecture Notes in Computer Science _ Interactive Distributed Multimedia System,, 2001, Vol. 2158, Pp. 124-137
- [5]. “An Architecture for a Scalable Video-on-Demand Server Network with Quality-of-Service Guarantees”, **Lars O. Burchard, Reinhard Lüling**, Lecture Notes in Computer Science _ Interactive Distributed Multimedia System,, 2000, Vol. 1905, Pp. 132-143
- [6]. “Augmented Reliable Multicast CORBA Event Service (ARMS): A QoS-Adaptive Middleware”, **João Orvalho, Fernando Boavida**, Lecture Notes in Computer Science _ Interactive Distributed Multimedia System,, 2000, Vol. 1905, Pp. 144-157
- [7]. “Middleware Support for Media Streaming Establishment Driven by User-Oriented QoS Requirements”, **Cristian Hesselman, Ing Widya, Aart van Halteren, Bart Nieuwenhuis**, Lecture Notes in Computer Science _ Interactive Distributed Multimedia System,, 2000, Vol. 1905, Pp. 158-171
- [8]. “Open distributed processing and multimedia”, **Gordon S Blair, Jean-Bernard Stefani**, British Library Cataloguing-in-Publication Data, 1997, Pp.1 – 86, (chapter 1-3)