Centrum voor Wiskunde en Informatica
REPORT*RAPPORT*

Network Quality of Service for Multimedia Presentation Generation Systems

F.J. Cornelissen

Information Systems (INS)

INS-R0106 May 31, 2001

Report INS-R0106
ISSN 1386-3681

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

Network Quality of Service for Multimedia Presentation Generation Systems

Frank Cornelissen
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

ABSTRACT
Quality of Service (QoS) protocols and architectures play an important role in distributed multimedia systems since multimedia presentations need to be played for the user without stalls, necessitating guaranteed response times. However in the Internet no other QoS than best–effort is given. This report investigates some QoS architectures and protocols designed to overcome these limitations and looks at the applications of these protocols for multimedia systems.

Given the recent research on (semi–)automatically generated multimedia presentations, this report investigates the opportunities to gear the generated multimedia presentations toward the specific network conditions of the client requesting a document, and dynamically adapting the presentation as network conditions change.

1998 ACM Computing Classification System: C.2.2, H.5.1, I.7
Keywords and Phrases: Multimedia, QoS, network protocols, presentation generation
Note: The research reported here has been carried out under the Dynamo and RTIPA projects.

1. Introduction

Multimedia presentations are compositions of individual media objects such as a video, image, text, audio etc. that are displayed to the user in a coordinated way. The most prominent difference between hypertext and multimedia is the addition of the time axis. This requires the synchronization of media items. For example, a presentation that contains a video and wants to show some text when some specific event occurs in the video will have to show the text at the specific time-point of the event in the video.

This coordination requirement presents some interesting problems for multimedia systems. For the coordination to work, it must predict when the media items should be played (which can be determined from the presentation layout) and make sure these items are available in time for display. Within a distributed multimedia system, as described in figure 2, this poses extra requirements on the network.

To create a multimedia system for a general public, the Internet can be used. In this network, the bandwidth varies drastically: a user in the same building will have a high bandwidth and low latency connection, but a user on another continent will have a low–bandwidth, high–latency connection. Given that the protocols used on (most of) the Internet are "best–effort" protocols, no assumptions about any of the timing of the delivery can be made.

To display the multimedia presentation perfectly, a player could load all the media–items referenced in the presentation, and only start displaying it when all items are completely available. However, this will imply a large load–time for the presentation, which the user will not appreciate. This also puts significant requirements on the client which should then be able to buffer all the items which might not be feasible on a device such as a hand-held. Therefore the multimedia system needs to load the media elements of the presentation on demand. In order to meet the timing requirements of the presentation the system needs to make some assumptions about the network.

The arguments for not loading all media–items in advance can also apply to large single media–items, such as (longer) video and audio items. Otherwise the user will still have to wait a long time for the video to start. These items should be streamed to the user: divided into small

packets which the client can display individually thus avoiding a high latency. Another reason for streaming media elements is the viewing of live events: one should not have to wait for the event to be over to view it.

To create multimedia presentations for a broad audience is difficult. The users will have different platforms with different screen sizes, network connections etc., and have different information needs. To write a multimedia presentation for each of these different combinations is a large amount of work. A possible solution to this is to (semi–)automatically generate the presentations based on the requirements from the specific user requesting the document. The generation system can then handle the screen size of the specific user, the platform of the user, and the network connection between the user and the presentation generation server. This might mean not sending certain types of information which the platform cannot handle but sending alternatives, or for the video example showing a "lighter" version of this video when the current network path cannot handle the standard full bandwidth video at this specific point in time.

This report describes several QoS architectures and protocols, and their application to multimedia presentation generation systems. In the next section some protocols and architectures for QoS will be described. In section 3 two currently available multimedia standards will be described. In section 4 the QoS requirements and applications of these multimedia standards will be described, followed by a discussion and future work section.

2. QoS architectures and protocols

Quality of Service (QoS) of a specific system describes how that system will function with respect to a set of parameters. End–to–end QoS is usually defined to be between two applications. To provide end–to–end QoS (which defines more than best effort) not only does the network need to comply with some requirements (based on the QoS parameters and values), but the operating system and machine hosting the system need to as well, since the operating system might be too busy to alert the application in question to be able to meet a deadline. However, this report will focus on the network QoS, given that this is the limiting factor for timely showing presentations in the Internet.

Parameters relevant for network QoS are bandwidth, jitter and latency. Bandwidth is the amount of data that can be pushed through the network and is expressed in bits per second. Latency is the time period from when the sender puts the bits on the wire to the point in time when the receiver receives the bits. Jitter is the amount of variance in the latency. These parameters are all important to determine the time it requires to get a media item. The bandwidth determines how big a stream usage can be, e.g. it does not make sense to try and put a 1Mb/s stream into a network which only has a bandwidth of 0.5Mbps. The latency determines the latest possible point in time to start the request for a specific media item in order for the user to not experience a stall. Jitter determines how much data the client needs to buffer of a stream in order not to pause during playback while waiting for data. This report will mostly focus on the bandwidth parameter since this seems to be the most important parameter for multimedia transport.

In an ideal world there would be an end-to-end QoS mechanism that would always have guaranteed latency, bandwidth and jitter. However, this is not true in real life large scale networks such as the Internet, which has always been a best–effort network with no guarantees whatsoever about the parameters above. However, several efforts such as DiffServ are under-way to extend the QoS in the general Internet. On the other side of the spectrum lie protocols that, given the best–effort nature of the Internet, try to be more intelligent about using the available bandwidth and dynamically adapt to changes in the bandwidth. An example of this kind of protocol is RTP, the Real Time Protocol (see [13]).

2.1 Differentiated Services architecture

The Differentiated Services Architecture (DiffServ, [2]) is a way to provide differentiated network services. This allows a network operator to handle different types of data streams differently. Every packet inside a DiffServ domain is marked with a DiffServ code point (DSCP) which determines the behavior for the network elements to wards this packet. The DSCP indicates a certain Per Hop Behavior (PHB), which defines the behavior for all the network elements the packet traverses.

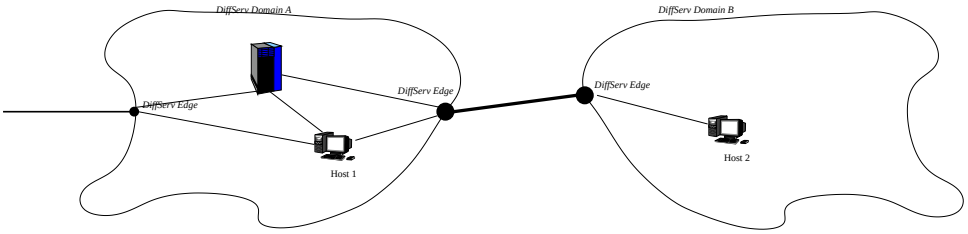

Figure 1: DiffServ domains. The splines indicate DiffServ domains. At the edge of the DiffServ domains there are edge routers (indicated by black points) which tag each packet with a DSCP. A domain is usually under one single authority of administration.

The marking of the packet is done at the edges off the DiffServ domain (see figure 1). An edge can be the host from which the packet originates (a workstation sending the packet) or the gateways to other networks (which may or may not be DiffServ enabled). The edges also need to make sure that the stream(s) they are sending out conform to the network and PHB limitations such as bandwidth. They can do this by shaping the traffic they send out by dropping or delaying packets sent out in the DiffServ domain. This will make sure the bandwidth used by these streams will not interfere with the policies set within the DiffServ domain.

DiffServ tries to provide network users with a model of QoS without putting a huge burden on core network devices. This is done by not looking at individual data streams from applications but instead aggregating these streams into classes. Thus the core network infrastructure only has to look at the DSCP in each IP packet to make a decision with respect to the QoS for this packet. For example, if it specifies that the packet is part of a guaranteed stream, then it will always be forwarded. Packets which do not belong to such a stream may be dropped by the router when congestion may otherwise occur. All this behavior is defined by the PHB indicated by the DSCP.

The result of this aggregation is that the network core infrastructure does not look at individual end–to–end flows (individual flows between applications, also called micro–flows), but instead to aggregations of these flows. The PHB's which build on top of the DiffServ architecture only work on these aggregations of the individual end–to–end flows and not on the micro–flows themselves.

The burden of the QoS handling is put on the in– and out–gateways and routers of the differentiated services network. The in–interface can be a host from which a data stream originates or a gateway to another network which doesn't use the DiffServ architecture. Here, the decision is made as to which PHB the incoming packet should receive, and the regulation of the packet streams is performed. Regulation means dropping packets which are above a certain agreed upon rate in case of expedited forwarding, or shaping packets in a node to smooth the rate of packet flow. This regulation is based on the state of the stream (for example the packet rate, which should be metered) and assigned PHB.

Two sets of DiffServ code points have been defined for the expedited forwarding per hop behavior and for the assured forwarding per hop behavior. Expedited forwarding is meant to provide a virtual leased line service, with guaranteed bandwidth, while assured forwarding allows one to prioritize packets. Each of these per hop behaviors will be described in some more detail.

Expedited forwarding The expedited forwarding per hop behavior [11] implements a virtual leased line, with the properties of low loss, latency, jitter and an assured bandwidth. This realizes an end–to–end connection between two nodes. This is accomplished by the following requirements:

- Each node has a well defined minimal departure rate that can be guaranteed under all load circumstances. This defines the load the core network can handle when faced with (severe) congestion.
- Arrival rate of packets at any node is always less than the minimal departure rate of that node. Given the above requirement, this means that all packets arriving at any node can

always be forwarded by that node.

The expedited forwarding PHB ensures the departure rate, while the network boundary controllers ensure the arrival rate through policing (dropping packets if they are out of profile) and shaping (delaying).

To implement this PHB on a broader network, a lot of information needs to be known about the intermediary network. This means that these types of virtual leased lines will not often be setup, since it requires a large amount of bookkeeping to ensure the above requirements are met, especially when the virtual line uses more than one domain of authority. Looking at figure 1, an expedited forwarding connection between the server in domain A and a host in domain B would require settings on both the edge servers of each domain. When a connection crosses more domains this would require settings in each domain on the edges.

Assured Forwarding Assured forwarding [7] allows discrimination between the probability of the dropping of the packet. Packets with a high drop precedence will be dropped with a higher probability than packets with a low drop precedence. It allows packets to be put in 4 classes (there is a possibility to have more classes in a DiffServ domain, but 4 classes are predefined by [7]) each with 3 drop precedence levels. Each class in each DS node is assigned a certain amount of forwarding resources (buffer space and bandwidth). The level of forwarding assurance for an IP packet depends on, in order of importance, the forwarding resources assigned to the assured forwarding class of this packet, the current load of the assured forwarding class and, in the case of congestion, the drop precedence of the packet.

Example service given in the RFC is the "Olympic service" with gold, silver and bronze service classes. Packets are assigned to these three classes so that packets in the gold class experience lighter load and thus have a greater probability for timely forwarding than packets assigned to the silver class. The same relations would hold between the silver and the bronze class.

In this PHB the translation that happens when a connection crosses the borders of authority domains is unknown to an application. It might be that the 2nd domain will translate the high priority packets from the first domain as normal packets in its network. There seems to be no automated way of discovering this at this time.

2.2 RSVP protocol

The Resource reSerVation Protocol, RSVP [6, 8], is a protocol for reserving bandwidth for micro flows. RSVP provides receiver-initiated setup of resource reservations for multicast or unicast data flows, with good scaling and robustness properties.

The RSVP protocol is used by a host to request specific qualities of service from the network for particular application data streams or flows. RSVP is also used by routers to deliver quality-of-service (QoS) requests to all nodes along the path(s) of the flows and to establish and maintain state to provide the requested service. RSVP requests will generally result in resources being reserved in each node along the data path.

RSVP requests resources for simplex flows, i.e., it requests resources in only one direction. Therefore, RSVP treats a sender as logically distinct from a receiver, although the same application process may act as both a sender and a receiver at the same time. RSVP operates on top of IPv4 or IPv6, occupying the place of a transport protocol in the protocol stack. However, RSVP does not transport application data but is rather an Internet control protocol. Like the implementations of routing and management protocols, an implementation of RSVP will typically execute in the background, not in the data forwarding path.

RSVP is not itself a routing protocol; RSVP is designed to operate with current and future unicast and multicast routing protocols. An RSVP process consults the local routing database(s) to obtain routes. In the multicast case, for example, a host requests to join a multicast group and then sends RSVP messages to reserve resources along the delivery path(s) of that group. Routing protocols determine where packets get forwarded; RSVP is only concerned with the QoS of those packets that are forwarded in accordance with routing.

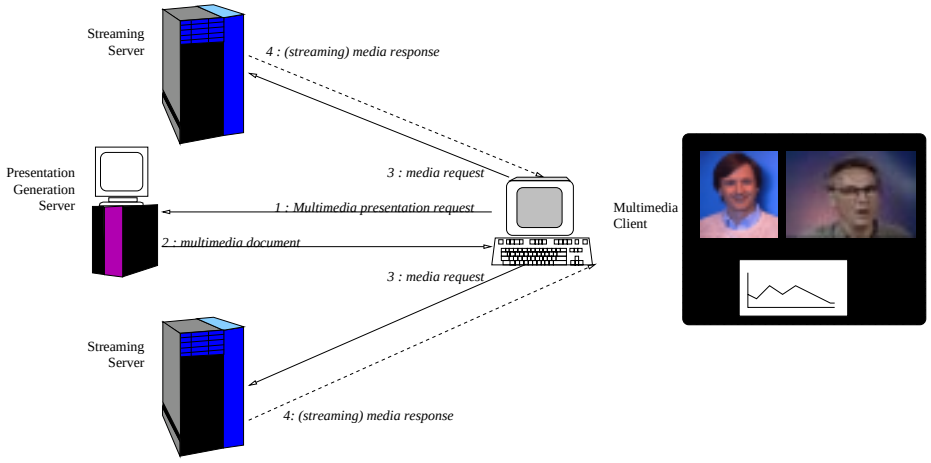

Figure 2: General model for distributed multimedia systems. A client makes a request (1) to the presentation generation server, which processes it and replies with a multimedia document (2). This is then interpreted by the client, and it will request the media items contained in the document from the streaming servers (3), who will reply with the requested streams (4). The resulting presentation is shown on the right

2.3 RTP

The Realtime Transport Protocol, RTP [13], provides end–to–end network transport functions suitable for streaming media data. The data can be sent to a single host (unicast) or multiple hosts at the same time (multicast). It does not address resource reservation nor guarantees (timely) delivery. The data transfer protocol, RTP, is augmented by a control protocol (RTCP, realtime control protocol) which enables the monitoring of data delivery.

RTP is used on top of other protocols, such as UDP. In itself it has no notion of connection nor does it provide any reliability features. It is typically implemented as part of an application, not by the operating system kernel. It consists of two parts, the data part which transfers the payload, and a control part.

The RTP data packets contain a 12 byte header followed by the payload, which can be a video frame, set of audio samples etc. The header includes a payload type indicating the kind of data contained in the packet (e.g. JPEG video, MP3 audio, etc), a timestamp (32 bits), and a sequence number to allow ordering and loss detection of RTP packets.

The control protocol, RTCP, serves several different purposes. It supports QoS monitoring and congestion control. The sender(s) send the number of RTP packets and bytes sent, so the client can estimate the actual data rate. Receivers send reports for all sources from which they have heard, with the highest sequence number received, number of packets lost, a measure of inter–arrival jitter and timestamps needed to compute an estimate of the round–trip delay between sender and the receiver issuing the report. It also supports intermedia synchronization, in that the reports issued by the sender give an indication of the real time (wall clock) and the corresponding RTP timestamp. This allows the synchronization of different media, for example lip–synching of audio and video.

3. Multimedia standards

Several standards for describing multimedia presentations are available and are currently being used over the Internet. Among those are SMIL 2.0 (see [16]) and MPEG4 (see [9]) which we describe in the following sections.

3.1 SMIL 2.0

Synchronized Multimedia Integration Language (SMIL 2.0) is a multimedia language which allows the author to layout media items along spatial and temporal dimensions. Interactivity within this presentation model can be represented through events (such as user input, for example) and links.

The SMIL 2.0 language is a media integration language. It specifies the spatial–temporal layout of the media objects, as well as the user interaction. In contrast with MPEG4, it does not specify a transport model.

The multimedia presentation can be authored to be independent of several system attributes such as screen size, platform capabilities and bandwidth. This is accomplished through the use of the switch statement, which selects one of the structures below it based on the test of the switch. Thus a video could be selected for a high bandwidth connection, but for a low–bandwidth dial–up client a (set of) still images might be shown. The SMIL 2.0 standard does not state how the bit-rate should be resolved, in practice the SMIL 2.0 players tend to implement it as a static property of the system.

Another feature of the SMIL 2.0 language is that it allows the author of the presentation to indicate that a certain media–item should be loaded through the use of the "prefetch" element. This feature allows an author (human or semi-automated) to determine which items will most likely be played next in the presentation and so instruct the player to preload the appropriate items. This mechanism allows parts of the presentation to be buffered in order for better interactivity for the presentation in the context of high latency networks.

3.2 MPEG4

MPEG-4 is a standard providing core technologies for efficient storage, transmission and manipulation of video data in multimedia environments. MPEG-4 considers scenes as compositions of audio-visual objects, supports hybrid coding of natural video and 2D/3D graphics in a common context (e.g. virtual 3D worlds) and provides advanced system and interoperability capabilities.

MPEG4 specifies a system for the communication of interactive audio visual scenes. It specifies the coded (binary) representation of media objects (such as images, audio and video), the spatial–temporal positioning of these objects and their behavior in response to interactions and the data stream management layer. The transport layer of MPEG4 describes a multiplexed representation of the individual media streams. MPEG-4 version 1, formally called ISO/IEC 14496 [9], has been available as an international standard since December 1998.

The aim of MPEG-4 is to provide a set of technologies to satisfy the needs of authors, service providers and end users alike, by avoiding the emergence of a multitude of proprietary, non-interworking formats and players. The standard should be used to allow the development of systems, which can be configured for a vast number of applications (among others, real time communications, surveillance and mobile multimedia). This is achieved by providing standardized ways to:

- interact with the material based on encoding units of aural, visual or audio-visual content, called "media objects". These media objects can be of natural or synthetic origin; this means they could be recorded with a camera or microphone, or generated with a computer;
- interact with the content, based on the description of the composition of the objects to create compound media objects that form audio-visual scenes; in a way one can understand the composition of audio-visual MPEG-4 objects as the representation of the real world, where spatial and temporal relations between objects allow the user to interact with these objects in a way similar to everyday usage;
- integrate different data types, allowing the harmonization of natural and synthetic objects, the usage of 2D and 3D, mono and stereo video or multiview video, and mono, stereo and multi channel audio, etc.;
- multiplex and synchronize the data associated with media objects, so that they can be transported over network channels providing a QoS;

- interact with the audio-visual scene generated at the receiver's end, e.g. to manipulate some characteristics of an object, to access selected parts of a scene, to remove an object from one scene and to integrate it into another, etc..

To meet the swift technological progress within multimedia, MPEG-4 developed the Syntactic Description Language (MSDL). This approach not only aimed at the integration of new algorithms through defined tools, but also that at any time new tools, techniques, and concepts could be adopted which should have provided improved or new functionality. In other words MSDL was the way to guarantee the flexibility of the standard, preventing eventual obsolescence and narrowness of scope. However, MSDL did not become part of the standard and was replaced by the Binary Format for Scene Description (BIFS), which filled a limited part of the role.

The major extensions of MPEG-4 over MPEG-2, with respect to network support, are:

- A standard functionality, e.g. synchronization of audio and video, modi for short delay and usage via networks;
- Content based manipulation of a bitstream without transcoding;
- Efficient coding of several streams at the same time (e.g. stereo video or several views of the same event, etc.);
- An improved random access on parts of an audio-visual sequence. Like its predecessors, MPEG-4 is concerned with streams. Since MPEG-4 subdivides audio-visual content into objects, the standardized characteristics of a stream are concerned with multiplexing, demultiplexing and the synchronization of multiple streams.

MPEG-4 allows meta-data describing their content to be attached to objects. Users of the standard can use this Object Content Information (OCI) datastream to send textual information along with MPEG-4 content.

4. Application of Network QoS for multimedia applications

This section describes what a multimedia application could do with more information from the network in a realistic large scale network such as the Internet in which no end-to-end quality of service can be guaranteed. However, (some of) the techniques described in section 2 provide something more than the current best–effort delivery of the Internet.

Next generation web systems generate multimedia presentations for specific clients (such as described in [15]) based on the user preferences, display device capabilities and a high level description of the message to be conveyed to the user. The high level description is translated into a presentation by selecting the media–objects which fit the description. The spatial and temporal layout of the media–items is also determined by this process. The generation process can also take into account the network properties and select the media items that fit in the network profile. For example, when the client is a small hand–held device and the network layer creates a lot of jitter in the delivery of media items, then if the client has insufficient buffer capacity to compensate for the jitter then a streaming video would not be an appropriate media type.

Given the distributed nature of the multimedia system shown in figure 2, there are two specific places where the adaptation to the network properties can be performed: the client and/or the server(s). In addition to the location, the presentation can be adapted at different points in the process flow: when the presentation is generated, and during the rendering of the presentation.

During presentation rendering, the multimedia system can dynamically adapt to the current state of the network (which is not known at generation time and is hard to predict, since it probably fluctuates to a large extent).

Another issue is where to adapt the presentation to the available network resources. During presentation generation, which happens on the server (apart from the request formulation which happens on the client), the obvious choice is the server. However, during presentation rendering, both the server (which sends the data) and the client (which renders the data) play an active role. Both can and should adapt their behavior to the state of the network (as far as they can deduce this information).

4.1 During Presentation Generation

The adaptation during presentation generation can give hints to the rendering application about which media objects are likely to be played next in an interactive presentation (using the SMIL 2.0 preload statement) and which media–object in a bandwidth-based switch should be played based on network information. It should therefore be able to predict the network bandwidth between the client and server(s). This could be done either by probing (using test streams) or based upon historical information.

SMIL 2.0 has a number of QoS/network related constructs, as described in section 3.1. These allow the player of the multimedia–system to make better guesses about the current required bandwidth, and give alternatives when that bandwidth is not available.

The SMIL 2.0 document could contain all the alternatives for the presentation embedded in the presentation document itself, using switch statements. This approach makes the adaptation process much harder, since for each presentation (which might be a response to a dynamic question) the presentation generation system should generate all possible alternatives of a presentation when only one of these is needed. This would generate a big load on the server.

Instead of generating all alternatives the server could probe the renderer for the features and bandwidth it thinks it supports. This could be done by generating tiny SMIL 2.0 documents which link to specific other SMIL 2.0 document URI's (the documents might still be generated upon request) inside switch statements to determine the profile for the client. The bandwidth could be estimated using a test stream, however this would increase the load on the network.

The ideal way would be for the presentation itself to be streamed, in which case the presentation generation server could adapt the presentation itself on the fly to the new media items which would be required for the current available bandwidth. However, in SMIL 2.0 this is not possible.

4.2 During Presentation Rendering

Since a wide area network can behave very erratically [10], a multimedia system should be able to adapt dynamically to network behavior changes during presentation rendering. This adaptation can be direct, applying to the media–item(s) currently playing or being processed, and indirect, applying to the media–items to be played but not yet started. In the latter case, the multimedia application can decide to show the upcoming media–items that satisfy the bandwidth restrictions observed.

Adaptation to the network for currently playing media items is harder, but is important when looking at multimedia presentations which contain long and bandwidth intensive media items that are streamed. At the start of such a media item (such as video or high quality audio), the bandwidth is available. However, halfway through playing the media item, the network might become more congested.

Depending on the severity of the problem (in other words, the bandwidth that is available in the network between the client and server) the system as a whole can adapt in several ways. One way is to degrade the quality of the media–item (graceful degradation), the other is to show other, content–wise comparable, information to the user (media replacement). Both of these types of adaptation will be described in the following sections. For either of these adaptations to work, the multimedia system needs to be able to perform bandwidth monitoring and make estimates of future bandwidth availability.

Graceful degradation For certain types of multimedia objects (referred to here as continuous media), such as video or audio, the perception of the quality of the media object by the user can degrade, but not diminish completely when a (small) subset of the information is not delivered. For a video, missing a packet might mean missing a small block of a single screen (updated 25 times per second) which will not influence the continuity of the stream as experienced by the user very much (for a majority of the packets, see [12]). In contrast, missing several letters of a text is very disturbing. In a control language such as VRML it might have disastrous effects. Using this property of continuous media objects (which are often quite large in size), one can handle congestion in best–effort networks. If there is congestion, a small number of dropped packets and/or adapting (i.e. lowering) the transfer rate of the media–stream will stop congesting the

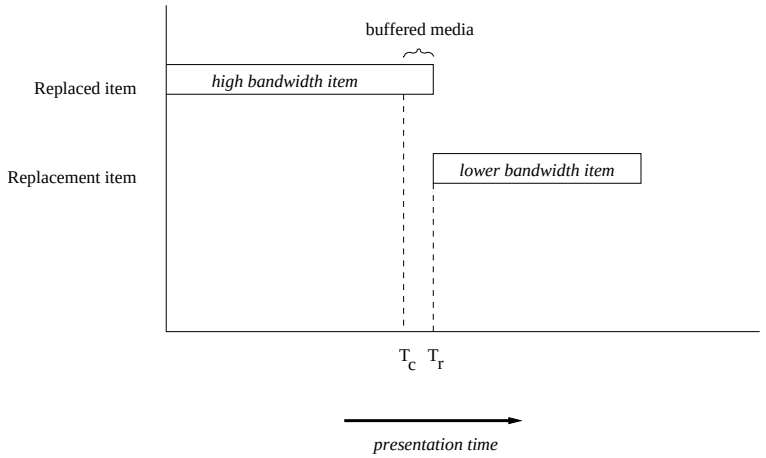

Figure 3: Media replacement during rendering. At time T_c congestion is reported to the application, which then decides to transport the lower bandwidth replacement media item, which starts at T_r. To do this smoothly, the application should have buffered enough of the original media–item to hide the request and transport time of the replacement item.

network, without having a large effect on the user's perception of the presentation.

However, in continuous media–objects some packets of information are more important than other packets (for example i–frames versus p–frames in MPEG video streams), so the packets should not be dropped randomly. Furthermore, dropping too many packets for a certain stream can make the information the media object should convey to the user incomprehensible. In this situation, the system should take more drastic measures, as described in the next section.

Since this type of adaptation is handled mostly by the network components in an architecture such as DiffServ (the Assured Forwarding PHB, see [5] for this application), the adaptation is instantaneous and involves no decision by either the renderer or server and will always apply to media objects that are currently being processed.

When there are multiple streams being sent to a client, these can interfere with each other in a congested network. Given a multimedia presentation, a given stream could be more important than another. Consider a discussion between an anchor-man and a reporter on location: the person talking should be given priority over the other in the typical case. The (streaming) server should thus know that one stream is more important than another, and more aggressively lower the sending rate of the lower priority stream. In the case of a distributed multimedia stream with multiple streaming servers this is more difficult but can be done by informing the servers of the priority of the stream and setting the progression of the transmission rate back down, which is usually a single threshold value in the transmission rate–control algorithm.

However, the effect of this is based on the topology of the network between the servers and the client. The solution sketched above only has the desired effect if all the servers share the same bottleneck between them and the client. This is of course not always the case, and determining this is probably too costly. In many cases, however, these servers will be close to each other on the network, and one can assume that the shared bottleneck exists.

Media replacement For a given high bandwidth media–item, alternatives may be available to the multimedia system that convey essentially the same information but are less bandwidth intensive. When the network between the server and the client becomes too congested to stream a certain media–item, a less intensive media–item could be played instead. This will lessen the overload on the network, while improving the presentation experience to the user.

To perform this replacement, several issues should be investigated (see [3] for a subset of these issues): the time when the media is replaced (either before a media item is rendered, or during

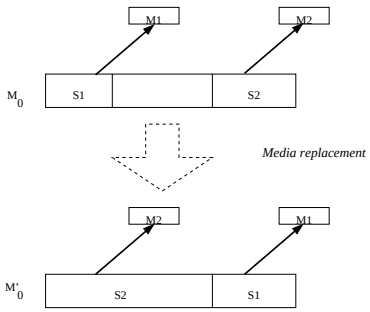

Figure 4: Synchronization issues for replacing media items. The element M_0 contains two synchronization points with other media items (M_1 and M_2) triggered by the subjects S_1 and S_2. The media–object selected to replace this media item (M_0') contains the same subjects, but in a different length and order, so the synchronization arch's need to be adjusted to keep the presentation semantics.

rendering), selecting the right replacement media items and determining when and which media item to replace. The sub–system that does the media replacement needs to be able to decide which media–elements might function as a replacement, and what bandwidth they consume.

In figure 3 an example scenario is given for media replacement during rendering. At the start a high bandwidth media item is streamed and shown to the user. Then, at T_c, the system receives a notification of a congested network. The system then decides that the media item should be replaced with the lower bandwidth item, which is then played at time T_r. In order to give the user a continuous presentation, the original item should have some buffered data that will be played during the transition period (T_c to T_r). This time period will be at least $2 \times \text{rtt} + \Delta$, where rtt is the round trip time between the client and the server and Δ is the amount of time required for processing on both the client and the server, plus any time that is needed to perform some minimal buffering (to avoid stalls due to jitter).

The media item replacing the original should fit into the presentation as it is currently shown. That is, it should have the same information content and the same synchronization points relevant for the rest of the media–items in the presentation (see [3]), or the (synchronization of the) presentation should be adapted to the new media item.

Adapting the presentation–flow is a task for the presentation generator (either human or automatic) since this is the only subsystem that has enough semantic knowledge about the presentation and media–items available. Depending on the multimedia format, the server system could either adapt to the changing network bandwidth as it occurs (during rendering) by changing the multimedia layout or during generation it could already consider several alternatives based on media thresholds. However, this would again depend on the presentation generator considering a set of alternatives in advance.

Another problem is the synchronization between the replaced and replacement media–item. When the original media item, which cannot be played any more due to a congested network, is replaced, the replacement media item should start playing at a corresponding time in order not to disturb the user's experience of the presentation. If the content of the media–items is conceptually the same, but the subjects are out of order, this might even involve dividing the replacement media–item into pieces in a way so that the current subject is played and no other subjects are repeated. Another option would be to revise the presentation structure as a whole, especially if the replacement media item has some other timing for events in the item which should be synchronized with the rest of the presentation. Changing the (temporal) presentation layout on the fly is not supported in SMIL 2.0 without planning the alternatives in advance by the use of switch statements.

These alternative media–items could be automatically generated from the original bandwidth intensive media–item (i.e. [14]) or be human authored, and the selection based upon certain criteria

of the meta–data of the media–items and presentation semantics ([3]).

4.3 Client–Server issues

Since the architecture of a generic multimedia system will be a distributed one, the question arises where the adaptation will take place. For the adaptation during presentation generation, only the presentation generation server applies since this is the only system involved at this stage, apart from the request by the client.

When the presentation is delivered and rendered, both the client and the server can adapt the presentation to the network. Using the standard UDP protocol for streaming the media where the server tags the packets sent with a sequence number, the client can deduce the packet loss in the network (and from that make an educated guess about congestion state and available bandwidth, see [4]). However, without a report back from the client (as can given by the RTCP protocol described in section 2.3) a server does not know about the actual packets received, and cannot make any such guesses. A streaming server can control the number of packets sent to the client, which it should do based on the (estimate of) the available bandwidth between the client and the server. This suggests that the client should give feedback to the server. This can be done using the RTP/RTCP protocol suite.

When the system realizes that not enough bandwidth is available for a certain media–item and it should be replaced, both the client or the server could initiate this change.

For the client to be able to make the decision, the presentation generation server has to annotate the presentation document with alternative media–items (the replacements) together with the bandwidth required for these. Given the actual throughput, the client can then request another media–item in the case of network congestion and shut down the stream for the item to be replaced. This requires the presentation generation server to generate several presentation scenario's if an alternative media item requires a different presentation structure (it might be that the subjects in the alternative media–item are handled in a different order, the duration might impact on other parts of the presentation, etc). This will put a bigger load on this server since it will have to evaluate a number of alternatives of which only a few will be applicable to the presentation that will finally be shown to the user.

When the server makes this decision, the client needs to be informed about which media item to play next, and its influence on the presentation structure (if any). However, the server would only have to evaluate the alternatives for the presentation structure as it arises, so it will have a lighter load, but will need a faster response time (any delays will be much more noticeable to the user, as they would potentially stall the presentation).

There are thus several issues in determining the division of adaptation tasks between the client and the server. These include server load, protocols and standards used. These are trade-offs that need to be made.

4.4 Network API requirements

The applications of network QoS described in this section require more API support than the current network support. The API should be able to report the latency and bandwidth between two hosts, allowing one to anticipate the timings required within the multimedia document with respect to the network load. To be able to perform media–replacement of continuous media streams, the network layer should provide a means of notifying the application of bandwidth changes. This can be done through a call-back function which is called when the bandwidth falls below a certain level.

5. Conclusions and future work

This report touches on a number of issues relating to network QoS and multimedia applications. What is needed is a prototype environment to implement the solutions presented here and experiment with their usefulness on a broader scale network.

For graceful degradation of media items, a rule of thumb needs to be established regarding the minimal bandwidth that needs to be allocated for a given media type for a certain purpose. Audio for entertainment, for example music, should have a better quality than an audio instruction.

call	*arguments*	*description*
getRtt	connection	returns the round trip time for the given connection. This can be used to deduce the latency for this connection
getBandwidth	connection, buffer time	get the current bandwidth. This should be a smoothed function over the given time
onBelowBandwidth	connection, buffer time, limit, callback	indicates the function to be called when the bandwidth (as determined by the previous function) goes below the given limit.

Table 1: Proposed network API extensions

Determining this lower level is more easy in constant bit rate media items than those encoded using a variable bit rate, where the bandwidth required for full quality playback is variable.

As has been remarked in this report, one can adapt the multimedia presentation to the current network bandwidth by replacing media items in the presentation with lower bandwidth ones. However, this requires that these alternative media items are available, and that their structure is the same as the playing one. This relation could be restricted by adapting the presentation structure to the newly selected media item. This requires that the presentation structure be changed during playback. For SMIL 2.0 , this does not seem to be an option. Future players could allow a scripting engine to change the document object model (DOM) of the presentation instead of using links for completely newly generated presentations. Also one could improve the SMIL 2.0 language to allow for more dynamic change.

Most of this text is about adapting the presentation to lower quality of service than expected. However, one should also be able to change to better quality when a sudden spike in network usage has passed. This is more difficult, since it is hard to provide feedback to applications about remaining bandwidth.

References

1. ACM. *ACM Multimedia '99 Proceedings*, Orlando, Florida, October 30 - November 5, 1999. Addison Wesley Longman.
2. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated services, December 1998. RFC 2475.
3. Susanne Boll, Wolfgang Klas, and Jochen Wandel. A Cross-Media Adaptation Strategy for Multimedia Presentations. In *ACM Multimedia '99 Proceedings* [1], pages 37–46.
4. I. Busse, B. Deffner, and H. Schulzrinne. Dynamic QoS Control of Multimedia Applications based on RTP. In *First International Workshop on High Speed Networks and Open Distributed Platforms*, June 1995.
5. Marc Chaumont, Amit Jain, Octavio Medina, and David Ros. Transport of Multimedia Flows on a Service Differentiation Architecture. Technical report, INRIA rennes, 2000.
6. R. Braden Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification, September 1997. RFC 2205.
7. J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski. Assured Forwarding PHB, June 1999. RFC 2597.
8. S. Herzog. RSVP Extensions for Policy Control, January 2000. RFC 2750.
9. International Organization for Standardization/International Electrotechnical Commission. Information technology – Coding of moving pictures and audio, 1999. International Standard ISO/IEC 14496:1999 (MPEG-4).
10. Van Jacobson. Congestion Avoidance and Control. In *Symposium proceedings on Communications architectures and protocols*, pages 314–329, 1988.
11. Van Jacobson, Kathleen Nichols, and Kedarnath Poduri. An expidited forwarding PHB, June 1999. RFC 2598.
12. Luis Rojas-Cárdenas, Emmanuel Chaput, Laurant Dairaine, Patrick Sénac, and Michel Diaz. Transport of Video over Partial Order Connections. *Computer Networks*, 31(7):709–725, April 1999.
13. H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, January 1996. RFC 1889.
14. John R. Smith, Rakesh Mohan, and Chung-Sen Li. Scalable Multimedia Delivery for Pervasive Computing. In *ACM Multimedia '99 Proceedings* [1], pages 131–140.

15. J.R. van Ossenbruggen, F.J. Cornelissen, J.P.T.M Geurts, L.W. Rutledge, and H.L. Hardman. Cuypers: a semi–automatic hypermedia generation system. Technical Report INS-R0025, CWI, December 2000.
16. W3C. Synchronized Multimedia Integration Language (SMIL) 2.0 Specification. Work in progress. W3C Working Drafts are available at http://www.w3.org/TR, 1 March 2001. Edited by Aaron Cohen.