Adaptive Video on Demand Service on RSVP capable Network[1]

Carlos Veciana-Nogués, Jordi Domingo-Pascual

Computer Architecture Department
Advanced Broadband Communications Center
Polytechnic University of Catalonia
Campus Nord. Mòdul D6. Jordi Girona 1-3. 08034 Barcelona.
{carlosv, jordid}@ac.upc.es

Abstract. At present, the provision of quality of service is one of the most relevant research topics. The Integrated Services approach defined by IETF and based mainly on the resource reservation protocol (RSVP [1]) is one of the issues that attracts many research work. On the other hand, research on adaptive applications [2] is another research topic. In this paper, we present a proposal that combines the features of RSVP and adaptive applications altogether. We put into work two protocols, RSVP and SVA[2] to get an optimum resource usage for a video on demand transmission. RSVP helps to maintain a given network bandwidth available, while SVA adapts video flow taking into account not only the network impairments but the client resources variations as well. We think that it is very important to cope with the resource availability at the end systems to provide a QoS to the final user. Once a flow is being transmitted, RSVP may renegotiate the reservation to adjust the bandwidth usage. Flow adaptation is done at the video server by reducing video flow size. We choose MPEG1 format because it has a good compression factor and allows a good quality. Our mechanism may work on other video formats with hierarchical relationships among frames. SVA[3] is a protocol that interchanges information between the client and the video server. The client informs the server about how many frames per second it is able to decode and to display, then the video server adapts the video source rate while it maintains frame synchronization and sequence dependencies. The experiences we present demonstrate that a good collaboration of reservation and adaptation procedures may provide a given QoS.

[1] This work has been partially funded by the Spanish Research Council under grant CICYT TEL97-1054-C03-03.

[2] SVA stands for the Spanish name “Servicio de Video Adaptativo”; in English “Adaptive Video Service”

Introduction

Delivering interactive multimedia information across a packet switched network with a best effort service is not a simple task. There are a lot of factors to take care of such as packet losses, synchronization among flows, error recover, scalability, admission control, bandwidth usage optimization and so on. Many proposals have been presented in order to "introduce" network quality of service. Both, the Integrated Services and Differentiated Services approaches are well known proposals with a lot of research work being carried out. From the user's point of view, QoS is a non-quantifiable characteristic about the "goodness" of the application [4](i.e. the video being displayed in the terminal). Usually, a range of "acceptable" quality is defined or allowed. QoS applies to all layers in a communication process: application, transport, network, link and physical medium. We will focus on the network, transport and application layers.

Usually, real time video communications work on a connectionless transport service (RTP/UDP) [6] over network service (IP), and they appear to the application layer as one service. QoS at network layer must allow video flows to maintain a given bit rate, a maximum delay and a delay jitter between a maximum and minimum value that allow the decoders to keep the synchronization.

At application layer, QoS must keep flow synchronization while decoding and displaying the video on the terminal. At this point resource management within the terminal (which includes the OS, the decoding process, inter process communication, etc.) affects the overall QoS perceived by the user. Resource reservation in the network (or providing different classes of services [5]) is necessary to provide QoS but it is not sufficient. Resource reservation at the end system is necessary as well.

Our scenario considers the following assumptions. There exists a resource reservation in the network (RSVP). A video server may deliver video flows to an heterogeneous set of terminals; each of them has its own properties and resources (i.e. hardware or software decoding). The end system may not be dedicated exclusively to receive and to process the video flow and thus the internal resources available at a given time may vary (i.e. CPU or memory). Most proposed adaptive applications take into account network losses and react according to network performance. We propose that the adaptation process must include the terminal performance as well.

We implemented an adaptive mechanism that monitors the terminal performance as well as the network performance. We tested it in an experimental platform [7] using the RSVP protocol to maintain QoS at network layer and SVA protocol to monitor and to adapt the client performance at application layer.

The paper is organized as follows. The first section introduces the framework for adaptive applications in real time communications. Section 2 gives a brief description of RSVP and explains how it fits in our QoS management architecture. The third section presents our proposal for the SVA protocol. Section four and five include a detailed description of our selective frame discard algorithm, how it applies to MPEG1 flows, and how it maintains frame quality while reducing flow volume. Section 6 presents the details of the application and the scenario for the tests including RSVP, heavy traffic and the video server, the client and SVA protocol.

Finally, some results that show the behavior of the overall system and its adaptation properties.

1 Simple QoS framework

Layered frameworks help developers and designers to simplify their job. QoS management must be present in all layers. However, layered frameworks are not the "best" solution to manage QoS in an effective way. Timing restrictions are the most important QoS characteristics for interactive multimedia applications. Information about QoS parameters must be interchanged and managed. The management plane controls certain aspects of several layers together.

Usually QoS requirements at each layer have been defined [4]. These definitions allow designers to get control of QoS per Layer. Nevertheless, the mapping of QoS parameters between Layers is a complex task.

Simplifying the QoS framework for a certain kind of application makes QoS parameters management simpler. We can see in [8] and [9] how, with a set of a few parameters (one per layer), a Video distribution application may provide some kind of QoS control.

In our proposal, we consider three Layers: Network, System and Application. We manage these parameters together (as a control plane) to get a good frame quality. The video stream is optimized taking into account the client performance, the server and the network load. The parameters considered are the following: packets per second (pps) at Network Layer, complete frames per second (cfps) at System layer, and visualized frames per second (vfps) at Application layer.

We consider Network Layer includes all the layers below IP layer. System Layer includes all the processes between Network layer and Applications layer, such as flow streaming, segmentation and reassembling. Application Layer includes stream decoding, synchronization and presentation. On top of them, the User Layer reflects the set of parameters related to user perception. Figure 1 shows our simplified framework.

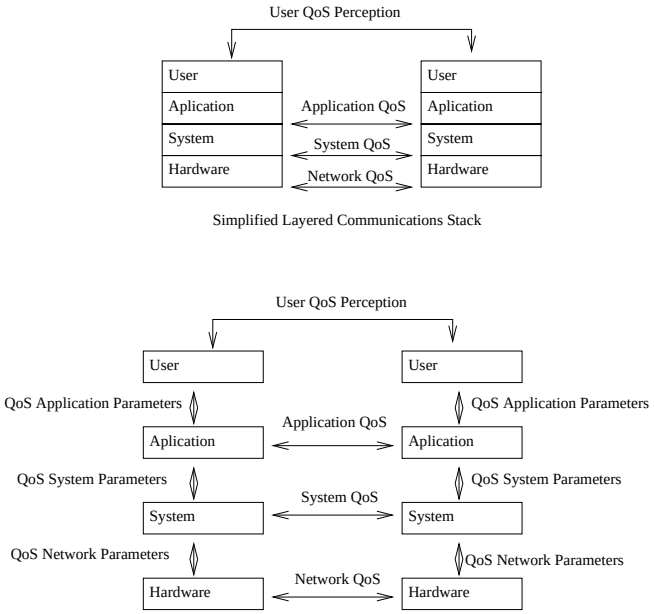

Figure 1. Simple QoS framework

Parameters pps cfps and vfps are directly related. Variations in received packets/second (pps) affect the number of complete frames/second available (cfps), and this affects the number of displayed frames/second (vfps). CPU load variations, concurrent network traffic, and concurrent processes affect vfps too. A feedback mechanism is used to inform the Server about the Client performance. The Client will send periodically the value of vfps to the server. Then, the Server will modify video flow, by discarding frames (or by including more frames) to fit in the free resources of the client. It is a self-regulated mechanism as shown in Figure 2.

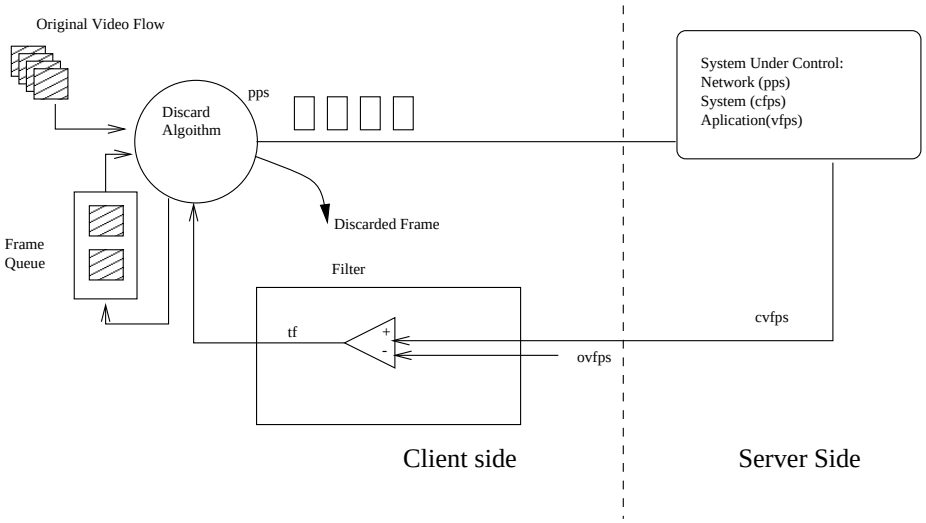

Figure 2. Self-regulated system based on a feedback parameter and discarding frames.

2 Allocating bandwidth with RSVP

The Reservation Protocol, RSVP [1], is an IETF's proposal to standardize a mechanism to reserve resources along the path within the network. We use RSVP over IP network protocol to reserve bandwidth for unidirectional connectionless transmissions (UDP).

The server advertises its session with a PATH message. The PATH message contains a list of parameters that specify the flow characteristics (mean bandwidth, peak bandwidth, MTU size...). This information and some other added by intermediate routers allow the client to ask for a reservation. Reservation message (RESV) is sent back to the server, creating a "soft state" at all intermediate routers that will allow to keep free resources to maintain the QoS for this flow. The soft state at the routers is refreshed via periodically PATH and RESV messages [10].

Mapping multimedia application parameters into Tspec (PATH parameters) and Rspec (RESV parameters) is too complex to leave this task for the user of the application. Some authors [11] propose a new layer above RSVP to simplify this task. They propose the definition and use of a database of well-known sources and their corresponding reservation parameters. This requires a lot of experimentation to find the "optimal" set of parameters for each application and configuration. We use a similar method by using only the bandwidth parameter for the reservation.

Nevertheless, RSVP forces all transmission systems (routers and end systems) to support this protocol. Soft-state per-flow management is too complex to be implemented at core routers. Core routers deal with large amounts of flows and data and require high speed and simple algorithms. Scalability is one of the main issues to

be solved. For backbone high-speed routers, it is envisaged to use differentiated-services oriented mechanisms to guarantee QoS.

Moreover, RSVP helps to maintain QoS but does not guarantee it. While it is maintained between routers, shared Ethernet LANs may produce packet loses. RSVP may help to protect a flow from other aggregated flows so that the adaptation mechanism gives a good performance. As mentioned above, end systems and workstations may behave poorly when the CPU load varies and affects multimedia applications performance too. This is the main reason to use an adaptive application [2] together with RSVP.

The reservation re-negotiation procedure can take a while due to high CPU load at the client. Response time may be critical for high quality flows in multimedia applications. Another problem may arise if RESV messages are lost due to high congestion in the network. In this case, response time for new reservation may be long and soft state information in the routers may be cleared by timeout.

3 Adaptation at application layer

Once a given QoS is guaranteed at network layer, the application layer must keep the quality specified at this layer. In the case of a video-based application QoS at application layer is good frame quality and timing, but not frame rate in most cases. Video degradation usually comes from the loss of frames at network layer or from a poor performance at application layer. Trying to maintain a given frame rate when the resources are not enough is not a good approach. There are several possible alternatives to reduce the bit rate of a video flow to fit in the available resources. Re-coding the video flow in a simpler format is one option. Another solution may be based in a discrete quality reduction, such as removing color. We discard these two alternatives because re-coding is too expensive in time and resources, and the reduction of colors is not scalable; once color has been removed from a flow no other simple treatment is possible.

Our approach is to reduce the number of frames per second because it allows smooth variations. As far as the frames are completely recovered, the quality is acceptable while the sensation of continuity is maintained at least until the frame rate is reduced to a half of the nominal rate. Most high quality video flows are coded at 25 or 30 frames per second. Displaying 12 or 15 frames per second gives a good continuity feeling if frame quality is maintained. Even in the case the bandwidth of the transmission channel is very narrow or the server/client resources are very scarce, less than 10 frames per second may be acceptable in some cases.

Using RSVP packet losses will not be frequent. However, loss of frames at the server or the client is possible because of overload of these systems. We will not discuss the Server system because we assume stored video flows distribution with enough resources. With this assumption, video coding is already done and files accesses and distribution on the network are the main tasks of the server. The Client at the receiver terminal must decode the video flow, and perhaps more than one flow if it is listening to a multiparty session. Lack of specific hardware, for example an MPEG decoder card, could be an important handicap when considering CPU usage in

the client. Other software and Operating System processes may slow down the video decoding process.

3.1 Adaptation mechanism in the Client.

Several factors limit the decoding rate. The first factor is the number of packets per second (pps) that are delivered by Network Layer. As we are using RSVP for allocating sufficient bandwidth, pps will be the same at Server and Client side. However, loss of packets is still possible, for example during reservation renegotiations. The loss of certain kind of information could produce the loss of a part of some frames and even the loss of whole frames, depending on the video format. The sequence of the received packets is the result of segmenting a certain number of frames per second into packets of a size according to the Network MTU. It will be a variable value because most video formats generate variable bit rates.

The client will receive these packets and reassemble them to build the complete frames. This process will give us the number of complete frames per second (cfps). The value of cfps may not be the same at the client and at the server side because of several reasons. The lack of CPU, buffer overflow at the Application Layer, the system layer being unable to deal with that frame rate may be some of the problems. In summary, all these factors directly affect the upper layer parameter perceived by the user: the number of visualized frames per second (vfps).
In an ideal situation where there are neither packet losses nor frame losses, cfps vfps are constant values. The Vfps is the Application Layer QoS parameter. It is the one that gives the end user the feeling of a good visualization. We may deal with pps, cfps and vfps together to manage the control plane, but we have decided to use only the vfps parameter because it comprises the overall performance of the client.

Table 1**.** QoS mapping, related parameters

QoS Layer	**QoS parameter**	**Acronym**
Network	Packets per second	Pps
System	Complete frames per second	Cfps
Application	Visualized frames per second	Vfps

The vfps parameter will be monitored during video presentation. Then, varying values in pps and cpfs parameters will be detected at Application layer. As stated above, the main factors that contribute to a certain vfps value are:

- Packets lost at network.
- Too late arrival packets
- Client buffer overflow
- Client insufficient capacity

3.2 Adaptation mechanism

Periodically, the client sends feedback information to the Server. This information includes the current value of vfps (cvfps). The server compares this value with the original vfps of the stored flow (ovfps). Then, the server computes the frame reduction it must apply to fit the Client resources.

$$\text{Transmission Factor} = \text{cvfps}/\text{ovfps} \quad (\mathbf{1})$$

The Transmission Factor parameter (TF) means how much the server should reduce the flow, in frames per second, to adapt to current client resources.

Next step is to decide how to use the TF. For example, if we have a video coded at 30 fps (ovfps) and our client is displaying 10 fps (cvfps), TF is 0,33. It means that we should drop 20 frames out of 30.

The TF means the frame transmission probability too. If we calculate TF every transmission time and accumulate it, when the accumulated value is greater than 1, we should transmit a frame.

Pause time between consecutive frames transmission varies to according to cvfps variable. Then synchronization is maintained. Table a in Figure 6 shows this example.

The whole system (Client cvfps, Server ovfps, TF, and the related parameters pps and cfps) is self-regulated. It means that it gets the optimal transmission rate taking into account both the channel and the client performance.

4 MPEG-1 principles

MPEG-1[12] format gets a high compression rate thanks to intraframe and interframe redundancy reduction. Intraframe compression does not affect our mechanism because we remove complete frames. If a packet loss occurs, the whole frame will be discarded. Interframe compression affects directly our mechanism because of inter-frame dependency.

MPEG-1 codification generates three kinds of images. Type I images (Intraframe codified) are stand-alone and do not depend on any other. Type P images (Predicted) depend on previous complete decoded P or I image. Type P images only code the parts of the image that have changed from the previous P or I image respectively. Finally, type B images (Bi-directional predicted) code differences between previous and future type I or B images. Compression rate for B images is higher than for P images, and P compression rate is bigger than that of I images. MPEG-1 combines these three kinds of images to get the best ratio between complexity and compression. Figure 3 shows inter-frame relationships.

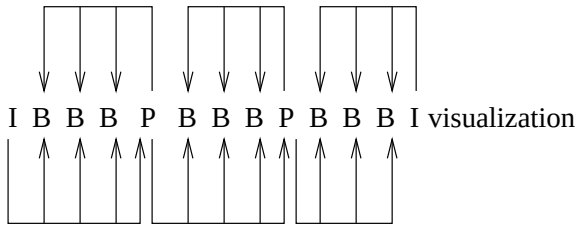

Figure 3. Inter-frame relationships. Arrows means that source frame is used to generate destination frame.

Because of this interframe relationship, frames are stored in a different order to optimize the decoding speed and to reduce buffer requirements. Figure 4 shows store/transmission order and visualization order.

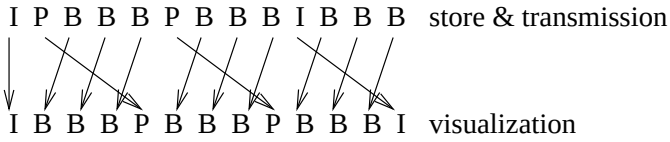

Figure 4. Store and transmission order

Discarding certain kind of images produces different effects on the rest of the video flow. Discarding B images do not affect video sequence decoding, B images errors are not propagated. Discarding P images affects past and future (in the order they are displayed) type B images, and future type P images. Discarding I images affects all images until the next I image arrives. Figure 5 shows and example of how P image discard/lost affects displayed flow. Frame loses produce some block error decoding in related frames and visualized frame may contain non sense square areas.

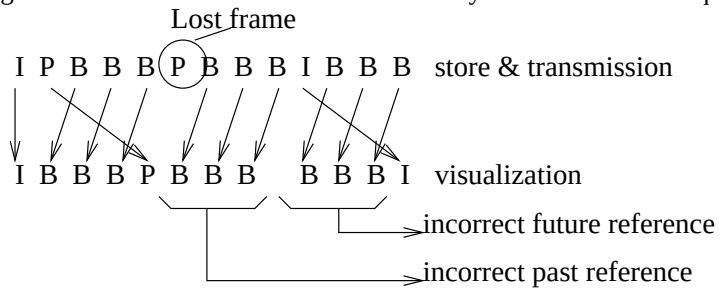

Figure 5. Lost frame effects

5 Selective frame discard for MPEG-1 video flows

In the previous section, we have seen a mechanism to decide how to reduce a video flow by dropping frames. However, frame discard affects the quality of the following frames. In order to maintain frame quality and sequence selective frame discard

algorithm must be applied. This selective discard must maintain inter-frame relationships [3].

In order to maintain interframe relationships we introduce a queue in the server. This queue will contain dropped frames with potential importance. We can send correct sequences combining the Transmission Factor with the current frame type to be sent and the frames stored in the queue,.

A high-level code description, which implements this process, is presented next. Note that cvfps and TF are constants in this piece of code, but they are updated in parallel by a monitoring process.

```
Loop
wait(1/cvfps)
send_prob=send_prob+TF// recalculate send probability

if TF<1   then // it's time to drop a frame

  if current_frame_type=B then
                           drop_frame

  if current_frame_type=P then
                           put_queue(q,current_frame)

  if current_frame_type=I then
                           empty_queue(q)
                           put_queue(q,current_frame)

else // TF>=1, it's time to send a frame

  send_prob=send_prob-1   // remember left probability
  if current_frame_type=I then
                           empty_queue(q)
                           send(current_frame)

  if current_frame_type=B then

          if queue_lon=0     then send(current_frame)
          if queue_lon>0     then send(extract_first(q))

if current_frame_type=P then
                           send(extract_first(q))
                           put_queue(q,current_frame)

  if current_frame_type=I then    empty_queue(q)
                                  send(current_frame)

end if
loop
```

A careful look at this algorithm shows that not all types of frames are dropped with the same probability. B frames first ones to be dropped, followed by P frame, and finally I frames. Then, a reduction in a percentage of frames does not contribute in the

same way to reduce the flow. This is because I frames are larger than P and B, while the discard probability is in reverse order. A queue to store and recover the last important P or I dropped frames is necessary. Figure 3 shows this modification.

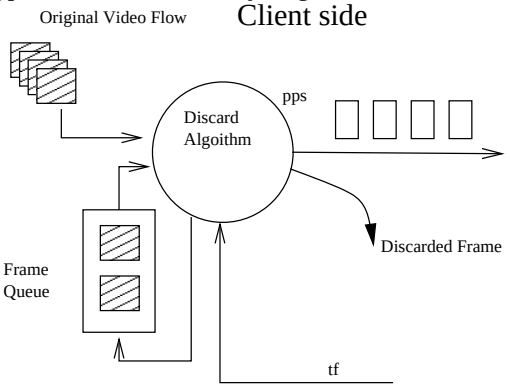

Figure 6. Client side with selective discard algorithm.

Figure 7 shows an example of the application of the selective discard algorithm, compared to raw discard. We can see how frame relationships are maintained, while some images are sent later. Images are never delayed beyond next displayed I time. MPEG-1 must transmit at least one I image per second. In the most greedy discard application this delay will always be not greater than a second and it will be transparent to the users most of the time.

$$\text{Transmision Factor} = \frac{\text{Current Frame Rate}}{\text{Real Frame Rate}}$$

Frame Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Decoding Order	I	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	B	B	P	B
Presentation Order	I	B	B	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	B	B	P	B	B	P	B	B	I	B	B
Transmission Factor	1	.66	.32	.98	.64	.3	.9	.62	.28	.94	.6	.26	.92	.58	.34	.9	.56	.22	.88	.54	.2	.86	.52	.18	.84	.5	.16	.82	.48	.4
Not Transmited		X		X			X			X			X			X			X			X			X			X		

Table a

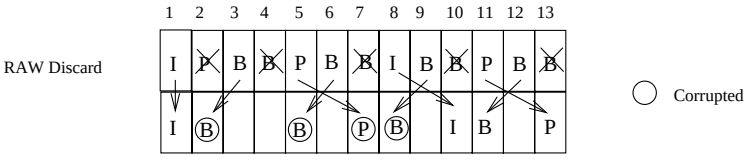

Table b

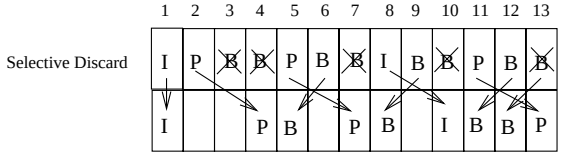

Table c

Figure 7. Selective Discard example

6 Application architecture.

Our test application SVA is MPEG-1 video server. It serves an MPEG-1 video flow over an unreliable UDP connection. We use RTP [6][13] for segmenting and synchronizing video frames. The feedback control channel is a reliable TCP connection that the client uses to send the feedback information (visualized frames per second) to the server. The server implements the selective frame discard algorithm described above to reduce the video flow (number of frames sent). The client uses the Berkeley mpeg_play [14] decoder to visualize video. Mpeg_play code has been modified to share some parameters that the client uses to calculate the number of visualized frames per second.
An RAPI [15] front-end application [7] is activated at the same time to make a reservation along path from server to client.

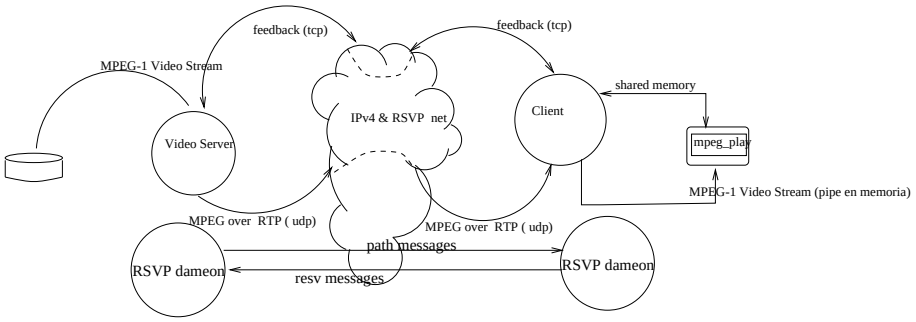

Figure 8. Application components and communication channels

A point to point transmission is established in the experiment. The proposal may be extended using multicast transmission. Scalability may be improved by using a hierarchy of active nodes [16], which apply the selective discard algorithm. In this way, the server can feed video flows for a group of clients with the QoS adaptation for each of them. In this case, the feedback information must be sent to the nearest active node, not to server in order to avoid control information overhead in the server. Active nodes must interchange feedback information too, to know which video size they can receive.

7 Results

Our test environment is based on a real deployment of the Internet architecture including the implementation of the RSVP test application and the SVA mechanism (figure 9). There is a router (CISCO 7026) interconnecting three nets: CCABA (Ethernet 10Mbps), SABA (Ethernet 10Mbps) and SABA-ATM (ATM 155Mbps). This configuration allows us to send interfering traffic straight through the router.

Router performance is affected while there are no collisions on the Ethernet segments.
WS1 (SunUltra-1) is the server and WS2 (SunUltra-1) is the receiver (client). RSVP reserves resources at the router for this UDP flow. Then, WS1 serves a video flow to WS2 on the reserved channel.
WS3 (SunUltra-1) stresses the router interface by sending Mgen [*] traffic to WS4 (PC486). The router must route this traffic from SABA-ATM network to SABA segment.
In this scenario, the frame-rate transmission is adapted to the client free resources only, because the network bandwidth is maintained in the router by using RSVP.
Two additional workstations WS5 (SunUltra-1) and WS6 (SunSparc20) are used to perform traffic measurement and run tcp_dump [*] to collect statistics about sent and received packets on both the source and the destination Ethernet LAN's.

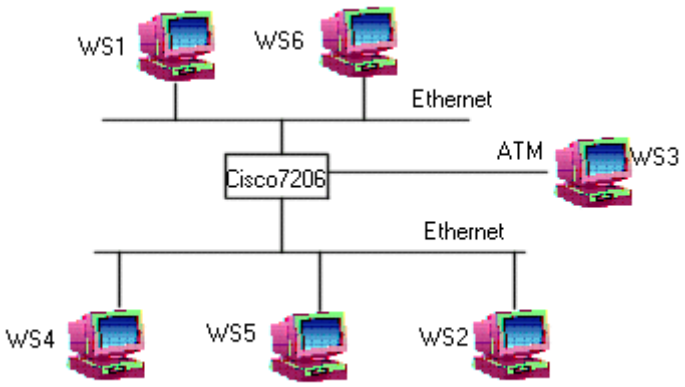

Figure 9. Network interconnection scenario.

The following tables show the effects of the selective discard algorithm on some video transmission. WS1 is Video Server and WS2 is the client and includes the video decoder. Table 2 shows the video characteristics: frame size, sequence type, decoding frame rate and number of each type of frames.

Table 1. Video characteristics

Video	Frame size	Sequence	fps	frames	I	P	B
Rnm	320x240	IBBBBBBBBBPBBBBBB BBBPBBBBBBBBB	25	1211	41	81	1089
ToEdge	160x120	I	30	1730	1730	0	0
Bbima	160x112	IBBPBB	30	13126	3751	1875	7500
Work1	382x288	IBBPBBPBBPBBPBB	25	1390	110	390	920
Team1	160x120	IBBPBBPBBPBBPBB	30	820	69	137	614
Team2	160x120	IBBPBBPBBPBBPBB	30	2694	225	450	2019

Valk	160x120	IBBPBBPBBPBBP	30	1684	141	281	1262

Table 3 shows how the video flow adaptation works in the server. Reduction of video flow depends on the decoding complexity. Decoding complexity varies due to the image size, sequence type and intraframe compression factor. Most videos do not reach original fps value because of the decoding process being made by software. Note that image dropping is done with different percentages depending on the image type. As mentioned before, the image type most affected by the discard algorithm is a B type frame, followed by P frames and finally I frames.

Several MPEG-1 video sequences have been used in the experiments. In Work1 (*2) test, the client CPU has been stressed with another local mpeg_play process with the same MPEG file.

Table 2 shows the results for the experiments. Those experiments labeled by number 2 correspond to a second scenario where additional CPU load is present in the client workstation. Work1(*2) is executed while another local mpeg_player is active. Valk(*2) is decoded with 24bit depth color, while valk(*1) is decoded with 8bit depth color.

Table 2. Selective discard algorithm results.

Video	Fps	Frames	I	P	B
Rnm	22	1089	35	75	979
ToEdge	28	1638	1638	0	0
Bbima	20	12451	3687	1742	7022
Work1(*1)	23	1291	100	339	852
Work1(*2)	15	743	74	237	614
Team1	30	820	69	137	614
Team2	30	2692	225	450	2019
Valk(*1)	30	1684	141	281	1262
Valk(*2)	23	1269	114	230	925

Valk(*1) can reach original fps without client CPU overload, but it decreases fps value if 24 bit depth decoding is applied. Figure 10 shows the client vfps (visualized frames per second) parameter and figure 11 shows frames per second served. A transient effect may be observed during the first seconds due to the empty buffer effect. Service speed changes are modulated by client buffer occupation. This effect will be removed by reducing the buffer and modifying the Mpeg_play to process strict time-stamp information.

Figures 12 and 13 show the same parameters but with a more complex decoding process. Application adaptation is done at a lower service rate.

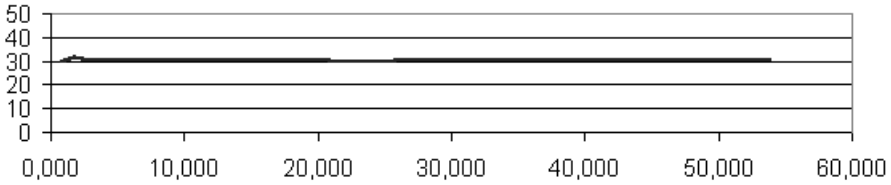

Fig. 10. Client frame rate during simple decoding process.

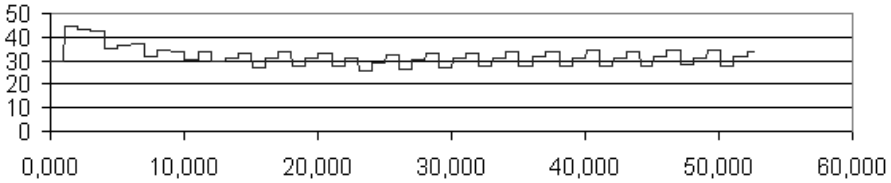

Fig. 11. Server frame rate during simple decoding process.

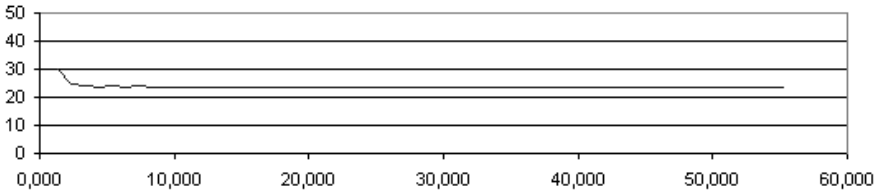

Fig. 12. Client frame rate during complex decoding process.

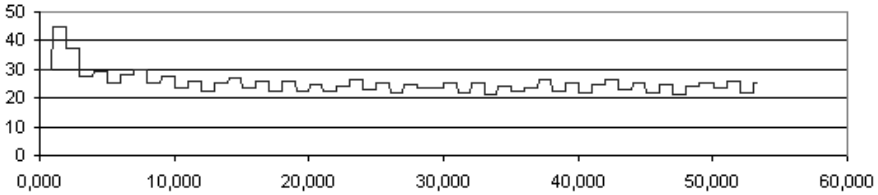

Fig. 13. Server frame rate during complex decoding process.

Conclusions

The RSVP protocol will help in QoS management by simplifying the QoS stack. Mapping stream characteristics on RSVP parameters, QoS at network layer and below is managed. However, parameters interchange between application layer and network layer must be introduced. This information interchange allow network layer to adapt to application performance, and application to adapt to network layer resources.

Selective frame discard algorithm allows us to reduce smoothly and gradually MPEG flows. Some approaches reduce flow volume by reducing frame quality and maintaining frame rate. These solutions display fast poor images. Selective frame discard maintains frame quality but reduces frame rate. Frame rate reduction in high quality videos can be done without affecting QoS perception in most of the cases. High frame rate reduction not decreases user QoS goodness perception as reducing frame quality does, especially in video-presentation and cooperative work applications, where users pay attention to audio and slides.

Response time of the RSVP and the feedback protocols must be improved to correct calculation of the global system configuration (server-network-client). Period of the feedback messages and the reservation refresh must be tuned jointly.

Adaptive applications must take care of QoS parameters in layers above network too. CPU load, as network congestion, can reduce severely application performance. Transmission of video streams larger than the client cant deal with is a waste of network resources. QoS management at application layer must help to optimize overall resource usage. Operating System re-design for networked multimedia applications is need for general-purpose computers.

REFERENCES

[1] R. Braden, L. Zhang, S. Berson, et al. “Resource ReSerVation Protocol (RSVP) – Version 1” Functional Specification. RFC 2205, 1997.

[2] Josep Mangues-Bafalluy, Jordi Domingo-Pascual. “A framework for Adaptive Applications”. Research report nº UPC-DAC-1998-7 (to be published). UPC-DAC Barcelona 1998.

[3] Carlos Veciana-Nogués, Jordi Domingo-Pascual. “Adaptación de flujos MPEG-1 para protocols de QoS best-effort”. Research report nº UPC-DAC-1998-10. UPC-DAC Barcelona 1998.

[4] Daniel G. Waddintong et al. “Specifying QoS for multimedia communications within distributed programming environments”. Lecture Notes in Computer Science, 1(1185):75—103. 3rd Cost 237 Workshop, Barcelona 1996.

[5] Network Working Group, "An Architecture for Differentiated Services". RFC 2475, December 1998.

[6] Audio Video Transport Group. “RTP: A Transport Protocol for Real-Time Applications”. RFC 1889, 1997.

[7] Esteve Majoral, “Multimedia Applications evaluation and implantation over new Internet protocols”, Final Project Degree. Facultat d’Informàtica - Polythecnic University of Catalonia, February 1999.

[8] Shanwei Cent, Calton Pu, et al. “A distributed real-time mpeg video audio player”. Lecture Notes in Computer Science, 1(1018):151—162. Proc. Of NOSSDAV, Durham 1995.

[9] Carlos Veciana-Nogués and Jordi Domingo-Pascual. “Transmisión de flujos multimedia con gestión de la Calidad de Servicio”. Proc. Yuforic’97, pages 13—20, Barcelona, April 1997.

[10] L. Zhang, S. Deering, D. Estrin, et al. “RSVP: A New Resource Reservation Protocol.” IEEE Network, September 1993.

[11] Bob Lindell. “SCRAPI: A Simple ‘Bare Bones’ API for RSVP, Version 2”, draft_lindell_rsvp_scrapi-01.txt (expires May’99)

[12] Joan L. Mitchell, et all. “MPEG Video Compression Standard”. Champman & Hall, 1986. ISBN 0-412-08771-5.

[13] Audio Video Transport Group. “RTP: Payload Format for MPEG1/MPEG2 Video”. RFC 2038, 1997.

[14] MPEG player. University of Berkeley. “MPEG utilities”. Available at: http://bmrc.berrkeley.edu/ftp/pub/multimedia/mpeg/, 1997.

[15] R. Braden and D. Hoffman. RAPI “An RSVP Application Programming Interface. Version 5”. Internet draft, draft-ietf-rsvp-rapi-00, 1997.

[16] R. Wittman and M. Zitterbart. “Amnet: Active Multicasting Network”. Proc. Of the 4th COST237 Workshop, pp. 154—164. Lisboa, Portugal, December 1997.