Videoconferencing in the Internet

Thierry Turletti and Christian Huitema

Abstract— **This paper describes the INRIA Videoconferencing System (ivs), a low bandwidth tool for real-time video between workstations on the Internet using UDP datagrams and the IP multicast extension. The video coder-decoder (codec) is a software implementation of the UIT-T recommendation H.261 originally developed for the Integrated Services Digital Network (ISDN). Our focus in this paper is on adapting this codec for the Internet environment. We propose a packetization scheme, an error control scheme and an output rate control scheme that adapts the image coding process based on network conditions. This work shows that it is possible to maintain videoconferences with reasonable quality across packet-switched networks without requiring special support from the network such as resource reservation or admission control.**

1 Introduction

As the bandwidth available on networks and the speed of computers increases, real-time transmission of video between general purpose work stations becomes a more and more realistic application. However, even with a high speed network, video has to be compressed before transmission. For example, sending uncompressed NTSC video requires about 60 Mb/s. Fortunately, there is so much redundancy in most video sequences that even a relatively simple compression scheme can significantly decrease the rate of video flows. Video compression is generally performed by some form of differential coding, i.e. by sending only the differences between two consecutive images. This leads to highly variable transmission rates because the amount of information to code between two images greatly varies, ranging from very low for still scenes to very high for sequences with many scene changes. Packet switched networks such as the Internet are very well suited for transmitting such variable bit rate traffic [8]. However, videoconferencing requires a minimum level of quality and the Internet does not provide such Quality of Service (QoS) guarantees yet. Nevertheless, we show that it is possible to obtain good quality using control congestion mechanisms to prevent clobbering of the shared resources.

One can find many video compression algorithms in the literature. Some of them have been standardized such as JPEG [2] for still images, or MPEG [18] and H.261 [25], [19] for moving images. MPEG-1 coding is suited for high definition video storage and retrieval [20]. MPEG-2 extends MPEG-1 to High Definition Television Coding (HDTV) applications [21].

The H.261 standard describes a complex video compression algorithm which allows to achieve a very high compression rate[1]. This standard was designed for use over the Integrated Services Digital Network (ISDN), i.e. for a network with fixed rate channels ($p \times 64\,kb/s$, $p \in [1, 32]$). We have implemented a software version of an H.261 codec for use over the Internet. This implementation is the core of the INRIA Videoconferencing System (*ivs*) [31]. By adopting a standardized algorithm, *ivs* can easily interoperate[2] with a large number of H.261-based commercial video codecs [14].

However, this standard is not designed for a packet switched networks such as the Internet. Since the Internet does not provide the same Quality-of-Service (QoS) as ISDN, we propose a set of schemes to adapt the H.261 video compression algorithm to this environment. In this paper, we describe a packetization scheme, an error control scheme and an output rate control scheme which adapts the image coding process according to the network conditions.

T. Turletti and C. Huitema are with INRIA, Sophia Antipolis, France.

[1] H.261 video compression rate can be easily adjusted, see section 5.3.

[2] See also the on-line html document <http://www-.cs.ucl.ac.uk/mice/codec_manual/doc.html>.

These three schemes are respectively developed in sections 3, 4 and 5. Section 6 evaluates the performances of *ivs*. Section 7 concludes the paper.

2 Relative Work

Without the IP multicast technology [10], the set of videoconferencing tools recently developed in the network research community could never be widely adopted. IP multicast technology extends the traditional IP routing model by providing an efficient multi-party packet delivery. The incremental deployment of IP multicast has been realized through the Multicast Backbone (MBone), a virtual multicast network built on top the current Internet [6], [26].

ivs is not the only videoconferencing application used by the MBone community. At the same time when we developed IVS, Ron Frederick from Xerox Parc was developing the Network Video tool (*nv*). More recently, Steve McCanne at UCB-/LBL developed the *vic* videoconferencing tool.

nv uses a custom coding scheme tailored for the Internet and targeted for efficient software implementation [11]. Its compression algorithm is based on a Haar Transform, a low computational complexity transform compared to the Discrete Cosine Transform used in H.261. In spite of a lower compression rate performance, *nv* coding is prefered by the MBone community mainly because of its better run-time performances.

vic has been built upon the lessons learned from both *ivs* and *nv* [27]. It is a flexible application which supports multiple network abstractions and several video compression algorithms. *vic* can interoperate with both *ivs* and *nv*. *vic*'s H.261 encoder uses only INTRA[3] encoding mode which greatly simplifies the algorithm and improves the run-time performances (in spite of a lower compression rate achieved as shown in Figure 17).

All these videoconferencing tools are regularly improved and upgraded versions are available in the public domain. Currently, *ivs* is the only videoconferencing tool which implements a control congestion algorithm by adapting its output rate to the network conditions.

[3] See definition in section 3.1.

3 The H.261 packetization scheme

We first give a brief overview of the H.261 video compression standard in order to better understand the following sections.

3.1 Overview of the ITU-T recommendation H.261

The H.261 recommendation describes a codec scheme to use for audiovisual services at $p \times 64$ kb/s ($p = 1, 2, ..., 30$). An H.261 coder analyses the successive images of the video stream as sets of blocks of 8×8 pixels. The algorithm can be decomposed in several steps: movement detection, Discrete Cosine Transform (DCT), Quantization and Huffman encoding.

After performing movement detection, the coder can either decide to encode the difference between the block and its previous encoded/decoded occurrence, or, if there is not enough correlation, to simply encode the new value. This is respectively referred as INTER-frames and INTRA-frames coding. INTRA-coded codes rely only on the redundancy within a single video frame when inter-coded code also uses the temporal redundancy of video to perform compression. In fact, H.261 does not transmit directly the pixel values or the differences, but rather the coefficients of their discrete cosine transform (DCT) [22]. Once transformed, these coefficients are then quantized and Huffman encoded before actual transmission. The coder scheme is shown in Figure 1.

The H.261 layers

The H.261 coding is organized as a hierarchy of groupings. The video stream is composed of a sequence of images (or pictures) which are themselves organized as a set of Groups of Blocks (GOB) (see Figure 2). Note that H.261 "pictures" are referred as "frames" in this document. Each GOB holds a set of 3 lines of 11 macro blocks (MB). Each MB carries information on a group of 16x16 pixels: luminance information is specified for 4 blocks of 8x8 pixels, while chrominance information is given by two "red" and "blue" color difference components at a resolution of only 8x8 pixels. These components and the codes representing their sampled values are as

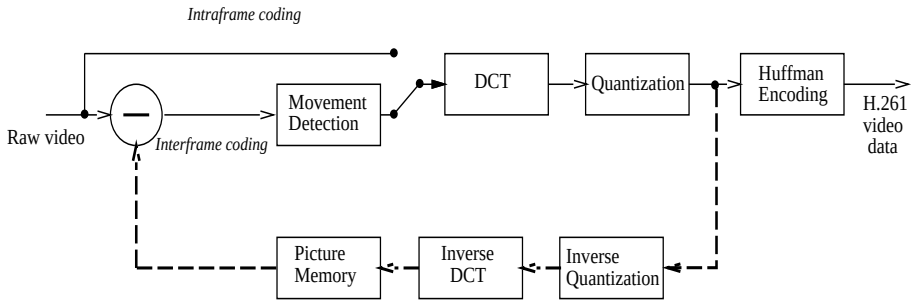


Figure 1: Basic H.261 coding loop

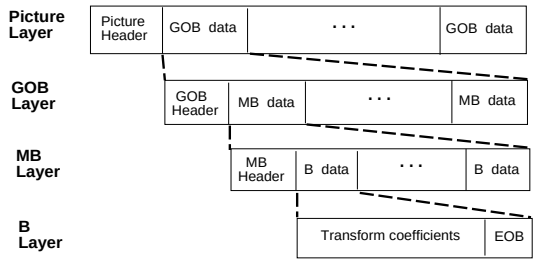


Figure 2: H.261 layers

defined in the ITU-R Recommendation 601 [7].

Two main sizes of images are defined: CIF[4] and Quarter CIF (QCIF). There are 12 GOBs for a CIF picture and 3 for a QCIF picture as shown in Figure 3.

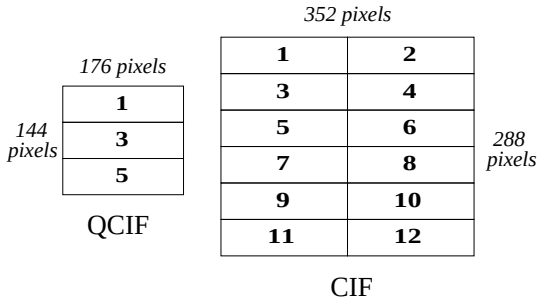


Figure 3: GOBs arrangement in CIF and QCIF

This grouping is used to specify information at each level of the hierarchy:

- At the frame level, one specifies information such as the delay versus the previous frame, the image format, and various indicators.

[4]CIF is the Common Interchange Format defined by the UIT; it is an interchange format for video images with 288 lines with 352 pixels per line of luminance and 144 lines with 176 pixels per line of chrominance information.

- At the GOB level, one specifies the GOB number and the default quantifier that will be used for the MBs.
- At the MB level, one specifies which blocks are present and which have not changed, and optionally a quantizer and motion vectors.

3.2 Considerations for packetization over the Internet

H.261 codecs designed for operation over ISDN circuits produce a bit stream composed of several levels of encoding specified by H.261 and companion recommendations. The bits resulting from the Huffman encoding are arranged in 512-bit frames, containing 2 bits of synchronization, 492 bits of data and 18 bits of error correcting code. The 512-bit frames are then interlaced with an audio stream and transmitted over $p \times 64$ kb/s circuits according to specification H.221[24].

Transmitting a video flow across the Internet requires a different approach. For instance, an application-level error control scheme is more efficient than the 512-bit framing (see section 4). Similarly, instead of using the H.221 framing, most of multimedia application requirements can be provided by the Real Time Protocol (RTP) (see section 3.3). A comparison between H.261 over ISDN and H.261 over IP is depicted in Figure 4.

Directly transmitting the result of the Huffman encoding over an unreliable stream of UDP datagrams would have very poor error resistance characteristics. The result of the hierarchical structure of H.261 bit stream is that one needs to receive the information present in the frame header to decode the GOBs, as well as the information present in the GOB header to decode the MBs.

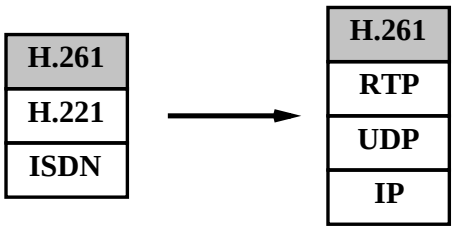


Figure 4: H.261 over ISDN vs H.261 over IP

However, a video image (or even a GOB) can sometimes be bigger than the Maximal Transmission Unit (MTU)[5]. The H.261 recommendation specifies that the maximal size of a CIF image is 32 kbytes (which means 3 kbytes for a GOB, 90 bytes for a MB and 15 bytes for a block). In practice, we observe that the H.261 image size is highly variable according to the quantity of movements and details in the encoded scene: it varies from a few bytes to about twenty kbytes. First versions of the H.261 packetization scheme used a GOB unit of fragmentation. To achieve better performances on lossy environment, the latest version of the packetization scheme takes the MB as unit of fragmentation. In the scheme, packets must start and end on an MB boundary, i.e. an MB cannot be split across multiple packets. Multiple MBs can be carried in a single packet when they fit within the maximal packet size allowed. This practice is recommended to reduce the packet send rate and packet overhead.

In the spirit of the Application Level Framing (ALF) model [9], the H.261 packetization scheme allows each packet received at the decoder to be processed independently. To provide an efficient resynchronization in the presence of packet loss, all the information required to decode an MB independently is sent in a specific RTP-H.261 header conjoined to the H.261 data. This header includes the GOB number in effect at the start of the packet, a reference to the previous MB encoded in this GOB, the quantizer value in effect prior to the start of this packet and the reference Motion Vector Data (MVD) for computing the true MVDs contained within this packet.

Moreover, since the compressed MB may not fill an integer number of octets, the H.261 header contains two three-bit integers, SBIT and EBIT, to indicate the number of unused bits in the first and last octets of the H.261 data, respectively.

[5]The MTU size depends on the network: e.g. it is 1536 bytes for an Ethernet network and it can be as low as 576 bytes for the Internet.

3.3 Overview of RTP

The Real Time Protocol (RTP) aims to satisfy the needs of multi-party multimedia applications: source-identifier, content-identifier, timestamp, demultiplexing, etc [29]. Moreover, RTP allows interoperability between the existing MBONE tools.

The RTP specification describes a very thin transport protocol which is the most often integrated into the application processing rather than being implemented as a separate layer. This is in accordance with the Application Layer Framing (ALF) spirit [9]. In the IP protocol stack, RTP is situated on top of UDP (see Figure 4). As a matter of fact, the RTP specification describes two protocols: the data transfer protocol (RTP) and the control protocol (RTCP).

Each RTP data packet is composed of an RTP header followed by the RTP payload (i.e. the data). The RTP header contains a sequence number, a media-specific timestamp and a synchronization source packet identifier (SSRC). Receivers demultiplex packets using the SSRC which is globally unique within an RTP session.

The RTP control protocol (RTCP) manages control information providing mechanisms for data distribution monitoring, cross-media synchronization and sender identification. RTCP packets are transmitted periodically to all participants in the session and the period is adjusted according to the size of the session. In this way, the RTCP bandwidth is limited in order to avoid the NACK explosion problem.

3.4 Specification of the packetization scheme

The H.261 information is carried as payload data within the RTP protocol. The following fields of the RTP header are specified:

- The payload type should specify H.261 payload format (see the companion RTP profile document RFC 1890).
- The RTP timestamp encodes the sampling instant of the first video image contained in the RTP data packet. If a video image

occupies more than one packet, the timestamp will be the same on all of those packets. Packets from different video images must have different timestamps so that frames may be distinguished by the timestamp. For H.261 video streams, the RTP timestamp is based on a 90kHz clock: this clock rate is a multiple of the natural H.261 frame rate (i.e. 30000/1001, or approximatively 29.97 Hz). That way, the clock is simply incremented by the multiple for each frame time.

- The marker bit of the RTP header is set to one in the last packet of a video frame, and otherwise, must be zero. Thus, it is not necessary to wait for a following packet (which contains the start code that terminates the current frame) to detect that a new frame should be displayed.

The RTP-H.261 header will follow the RTP header and precedes the H.261 data as shown in figure 5:

The fields in the RTP-H.261 header have the following meanings:

- *Start bit position (SBIT)*
  Number of bits that should be ignored in the first data octet.
- *End bit position (EBIT)*
  Number of bits that should be ignored in the last data octet.
- *INTRA-frame encoded data (I)*
  Set to 1 if this stream contains only INTRA-frame coded blocks. Set to 0 if this stream may or may not contain INTRA-frame coded blocks.
- *Motion vector flag (V)*
  Set to 0 if motion vectors are not used in this stream. Set to 1 if motion vectors may or may not be used in this stream.
- *GOB number (GOBN)*
  Encodes the GOB number in effect at the start of the packet. Set to 0 if the packet begins with a GOB header.
- *Macro block address predictor (MBAP)*
  Encodes the macro-block address predictor (i.e. the last MBA encoded in the previous packet).
- *Quantizer (QUANT)*
  Quantizer value in effect prior to the start of this packet. Set to 0 if the packet begins with a GOB header.
- *Horizontal motion vector data (HMVD)*
  Reference horizontal motion vector data (MVD). Set to 0 if V flag is 0 or if the packet begins with a GOB header, or when the MTYPE of the last MB encoded in the previous packet was not motion compensated. HMVD is encoded as a two's complement number.
- *Vertical motion vector data (VMVD)*
  Reference vertical motion vector data (MVD). Set to 0 if V flag is 0 or if the packet begins with a GOB header, or when the MTYPE of the last MB encoded in the previous packet was not motion compensated. VMVD is encoded as a two's complement number.

Recommendations for hardware codecs

Packetizers for hardware codecs can trivially figure out GOB boundaries using the GOB-start pattern included in the H.261 data. The cheapest packetization implementation consists to split the video flow at the GOB level by sending an entire number of GOBs in a packet. But when a GOB is too large, the packetizer has to parse it in order to perform MB fragmentation. Note that this requires relatively little processing since it is not necessary to fully decompress the H.261 stream to fill in the H.261 specific header. However, we recommend to use MB level fragmentation when feasible in order to reduce the output packet rate and therefore decrease the overhead.

At the receiver, the data stream can be depacketized and directed to a hardware codec's input. If the hardware decoder operates at a fixed bit rate, synchronization may be maintained by inserting the stuffing pattern between MBs (i.e., between packets) when the packet arrival rate is slower than the bit rate.

The packetization scheme described in this section is currently proposed as standard to the Audio-Video Transport Working Group (AVT-WG) at the Internet Engineering Task Force (IETF) [32].

4 The error control scheme

Errors in a video stream require a different form of correction than errors in a normal data stream. Tests transmitting video stream over a standard TCP connection allowed us to transmit data over

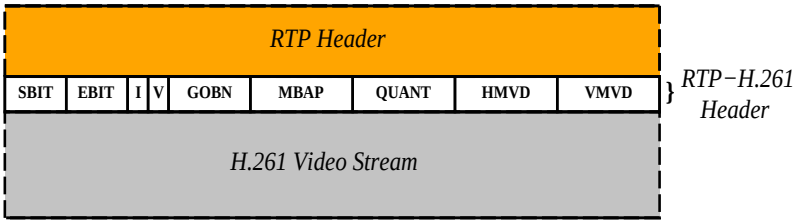


Figure 5: The H.261 header format

the Internet without concern of lost or out-of-sequence packets because of TCP reliability [15]. However, retransmission introduces delays which are not acceptable for real-time applications such as videoconferencing. It is more convenient to use UDP and construct application specific reliability services.

On the Internet, most packet losses are due to congestion rather than transmission errors [3], [28]. Alternatively, packets can be delayed or received out of order. This could happen as a result of the routing and flow control in the network. Due to real-time requirements, delayed video packets are considered as lost packets if delay exceeds a maximum delay value[6]. Using UDP, no mechanism is available at the sender to know if a packet has been successfully received. It is up to the application (i.e. coder and decoder) to handle packet loss and re-sequencing of out of order packets.

Each RTP packet includes a header in which a sequence number field and timestamp are stored. The sequence number is incremented by one for each packet sent whereas the timestamp reflects the time when the frame was grabbed. Packet losses can be detected using the RTP sequence number.

The H.261 algorithm uses the temporal redundancy of video to perform compression. The point is that differential coding (or INTER coding) is sensitive to packet loss. Figure 7 shows a classical "Head and shoulders" video sequence usually called *Miss America*. The image on the left shows the effect of packet loss one image after the loss occured. In this experiment, the *Miss America*'s head was moving to the right. We note that a lot of blocks around the face are corrupted. Actually, the part of image affected by the loss will remain blurred as long as all corresponding MBs are not INTRA encoded. There are several ways to mitigate packet loss:

- The safest way consists to use only INTRA-frame encoding and MB level conditional replenishment[7]. INTRA coding is much less efficient than INTER coding because a large amount of temporal redundancy exists in image sequences. Removing the INTER coding will result in significantly decrease of the compression ratio [8].

- A more efficient way consists to replenish, in INTRA mode, only the MBs concerned by the loss. As all GOB belonging to a given video image carry the same timestamp, the receiver can determine a list of GOBs which were really received for that timestamp and thus identifying the missing blocks. Requesting a specific reinitialization of these missing blocks through a "Negative Acknowledgement" (NACK) packet is more efficient than requesting a complete refreshment of the image through a "Full INTRA Request" (FIR) packet. Figure 6 shows the NACK emission after a packet loss. In this example, the coder uses QCIF and all GOBs are sent in different packets. When the decoder notices that it didn't receive packet B, it sends a NACK packet to the coder. The NACK information means that "GOB 3, image n is lost." The encoder will put all the MB encoded in the lost packet B into packet E, using INTRA mode. Actually, this is a forced replenishment and not a retransmission procedure because encoding occurs for a new frame and not for the previous lost frame. The left image on Figure 7 shows the image after the replenishment procedure [9]. However, the NACK-based method can lead to the feedback explosion

[6] The maximum delay value is empirically set to 100 ms in *ivs*.

[7] This method is currently used by the *vic* videoconferencing tool.

[8] We estimate that the INTER mode increases the compression ration by about 30 %, see Figure 17.

[9] In this case, the replenishment happened 3 images after the blurred image (i.e. 300ms after for this 10 f/s experiment).

Figure 7: INTRA refreshment after NACK receipt

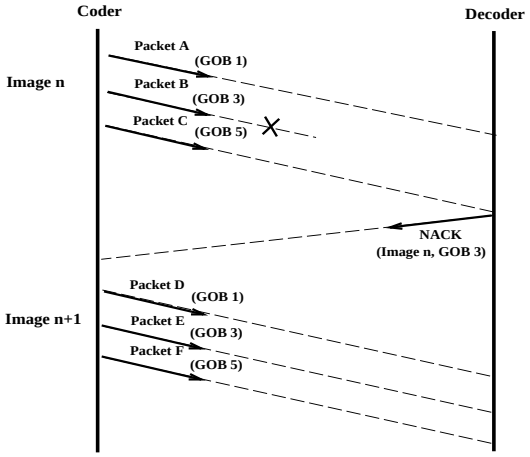


Figure 6: Data and NACK packets

problem when receivers are too numerous. If all participants generate NACKs packets each time a packet is lost, network congestion problems will appear very soon. Also, regular hardware H.261 codecs are not designed to accept NACK packets.

• A third way consists to periodically refresh the image in INTRA encoding mode. For control of accumulation of inverse transform mismatch error, the H.261 recommendation requires to INTRA encoding of each MB at least once every 132 times it is transmitted. In order to speed the recovery procedure, the coder can adapt the INTRA refreshment rate according to the packet loss rate information[10].

ivs implements the second and the third methods, and the method is selected according to the size of the session: we empirically set the threshold to 10 participants. When there are less than 10 receivers, NACKs packets are used. Else, the INTRA refreshment rate is adapted to the network congestion state. Let us examine what it means in term of bandwidth. The two extreme cases for multicasting distribution are the star and the "chain" network topologies, see Figure 8. In the following analysis, p is the average packet

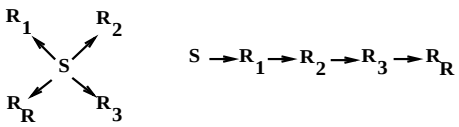


Figure 8: Star and Chain network topologies

loss observed by receivers, R is the number of receivers in the session and N is the number of data packets sent during the T interval of time. In the star network, during the T interval of time, $(R\,N)$ packets are sent by the sender and $(p\,R\,N)$ NACK packets are sent by the receivers. The numbers are (N) and $(p\,R\,N)$ for a chain network, respectively. Then, the proportion of NACK packets to

[10] Receivers can periodically send back to the video coder the packet loss rate they observe [5].

data packets is within the interval $[p,\ p\,R]$. The corresponding bandwidth proportion must take into account the length of the data and feedback packets. With an average of 500-bytes per data packet and 12 bytes per NACK packet sent, the proportion of bandwidth used by the feedback channel is within the interval $[\frac{12}{500}\,p,\ \frac{12}{500}\,p\,R]$. In *ivs*, the maximal R value is 10. For example, if the session has ten participants and if the average packet loss rate is 20%, the corresponding feedback traffic remains below 5 % of the data traffic (between 0.48 and 4.8 % depending on the network topology).

5 The congestion control scheme

Videoconference on the Internet could well be a "killer application"; the network administrators nightmares are probably already populated by thousands of hosts all sending megabits of videos over the net and swamping the T3 based backbones. The text book solution for controlling video over the network is called "resource reservation," generally combined with "admission control." To put it shortly, whoever wants to start a video transmission is supposed to ask for permission first, requesting a certain amount of resources. It is assumed that the characteristics of the data flow is known before starting the session and that the network has enough knowledge to decide if such a flow should be admitted or not. There is one advantage to this model: once accepted, one has a guaranteed QoS. But there are also many drawbacks, like the need to couple admission with accounting, the need to enforce complex reservation schemes, and, last but not least, the need to introduce a virtual circuit philosophy in an otherwise purely datagram network. Intensive work is currently in progress in the IETF to provide Internet applications the QoS required for their data flows (see the *ReSerVation setup Protocol* (RSVP) [34]). However, since such resource reservation are not yet currently deployed, we investigated another solution, trying to validate, for video, the "end to end" control model that had been so successful for classic data exchange.

End to end control needs two components: a network sensor and a throughput controller. Feedback mechanisms for video sources have been proposed for networks with variable capacity channels such as the Internet. There, the goal is to adjust the parameters (and hence the output rate) of video coders based on feedback information about changing network conditions, i.e. changing capacity in the network. Gilge and al. propose to use feedback control, but they do not describe specific control mechanisms [12]. Wakeman at UCL describes a specific scheme [33]. However, this scheme requires that the source of a connection knows the service rate of the bottleneck on this connection. This service rate can be estimated in networks where the switches use a so-called Fair Queueing or equivalent discipline, for example using the packet pair mechanism described in [17]. However, it is not available in networks with FCFS switches such as the Internet. Other work [16] describes a feedback control scheme which requires that switches send their buffer occupancies and service rates back to the source. Next we describe the network sensor and the throughput controller implemented in *ivs*.[11]

5.1 The network sensor

The Internet infrastructure does not provide sources of traffic with explicit feedback information about the state of the network (e.g. queue occupancies at the switches). The only easily available information is implicit information such as measures of losses and/or round-trip delays. In *ivs*, we use a feedback information which is based on measured packet losses.

In order to monitor how many video packets arrive at their destinations via multicasting, a source should obtain information from each receiver indicating packet loss on the path from the source to that receiver. One possible approach is to let each receiver send a NACK packet whenever it detects a loss. However, this can lead to the well-known NACK explosion when the network is congested. Another approach consists to periodically send a quality of service (QoS) measure which is the packet loss rate observed by receivers during a time interval of length T. We refer to T as the averaging interval. In *ivs*, we take T to be the time required at a receiver[12] to get 100 packets. As suggested in the RTP draft

[11] A more detailed description can be found in [4].

[12] Observe that intervals lengths might be slightly different for different receivers.

document, we also make sure that each receiver sends feedback information at least once every 2 minutes.

It is clear that the QoS approach is more efficient than the NACK approach as soon as the packet loss rate is higher than 1%, which is almost always the case on the Internet. To further decrease the impact of feedback traffic on the network, each receiver delays its feedback message by a random amount of time drawn from the range $[0 \dots T]$. This technique is employed to prevent receivers from sending back their feedback at the same time which could create periodic congestion on the network.

Each receiver sends its measured loss rate back to the source using RTP[13]. Then the source converts the different measures of QoS into a "global" measure characterizing how well packets arrive at the receivers. Our approach is to use the median loss rate.

5.2 The throughput controller

Control actions are taken by the coder at discrete points in time, specifically whenever a sequence of 100 packets has been encoded and sent. Of course, the number 100 is chosen so that the interval between successive controls is approximately equal to the interval over which the feedback information is computed at the receivers.

During a control action, the control algorithm adjusts the maximum output rate of the coder max_rate so that the median loss rate stays below a tolerable loss rate. The median loss rate is denoted by med_loss, and the tolerable loss rate by tol_loss. Specifically, max_rate is decreased by a factor of two if the median loss rate is larger than tol_loss. Otherwise, it is increased by a fixed fraction of its current value. We also make sure that the output rate is always larger than some minimum rate to guarantee a minimum quality of the videoconference at the receivers. Receivers whose connections have an insufficient quality are expected to either obtain more resource through some reservation mechanism, or leave the conference.

Thus, the control algorithm is as follows:

if ($med\_loss > tol\_loss$)
  $max\_rate = \max(max\_rate/2, min\_rate)$;
else
  $max\_rate = gain * max\_rate$;

In *ivs*, we set min_rate = 10 kb/s, $gain$ = 1.5, and tol_loss = 10%. We also set the maximum value of max_rate to 100 kb/s. In these experiments, the video source is an *ivs* source at INRIA. The number of receivers is such that the environment is a "large multicast" environment, i.e. receivers send QoS packets periodically back to the source. We analyze the connection between the *ivs* source at INRIA Sophia Antipolis and a receiver at University College London (UCL). Note however that the connection between UCL and INRIA is a multicast connection, i.e. the packets sent over the connection are carried over the MBone. At this time[14], the multicast path from France to Great Britain goes through CERN in Geneva and Amsterdam in the Netherlands.

Figure 9 shows the evolution of the maximum output rate max_rate at the source (plain line) and the average packet loss computed at the receiver (dashed line) during 20 minutes. The unit on the x-axis is the frame number, the average frame rate was about 4 images per second.

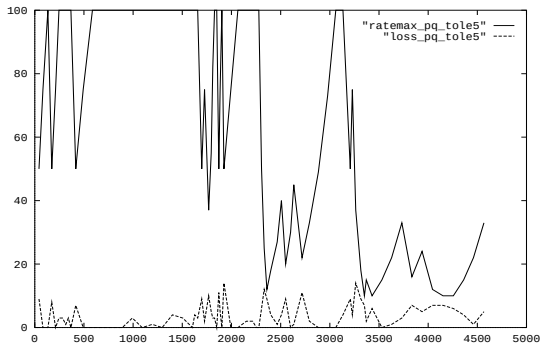


Figure 9: Evolutions of max_rate (in kb/s) and the loss rate at the receiver (in %).

As expected, we observe that the value of max_rate at the source decreases as losses are detected at the receiver. When the packet loss rate is higher than 10%, the max_rate value is decreased by half. Then, the *ivs* video source is

[13] The RTP specification provides a framework to send QoS information [29].

[14] This experiment has been made in October 1993.

able to adapt its output rate to the network conditions observed.

We have described above the control algorithm used in *ivs*, i.e. how the maximum output rate of the coder is adjusted based on the feedback information. Next, we describe how the output rate can be adjusted in H.261 codecs and the way it is implemented in *ivs*.

5.3 Output rate control for H.261 codec

Several parameters of a H.261 coder can be adjusted to change the output rate of the coder. The easiest method consists of modifying the frame rate of the application. However, this solution is only appropriate when the refreshment rate is not a key parameter of the application: experience shows that the rendition of movement is often a very important requirement.

A second way to control the output of the coder is to adjust the value of the quantizer. By using a looser quantization factor for the coefficients, one reduces the precision of the image – this is approximately equivalent to reducing the number of bits per pixel. The resulting coefficients are less variable than the original values, and result in fewer encoding bits after image compression. However, when the quantizer value is set too high, the image becomes blurred during these changes.

Figure 12 shows that the output rate of the coder as a function of the quantizer. These results have been obtained for the well known "Miss America" test-sequence. When the value of the quantizer decreases, the output rate of the coder increases and so does the image quality (see Figure 5.3).

A third way to reduce the data rate is to simply reduce the number of blocks which are encoded for each image. This can be done by raising the movement detection threshold. This threshold controls the number of the blocks which are "sufficiently different" from the previous frame. If the threshold value increases, then the number of blocks to process decreases and hence the encoding time and the number of bytes required to encode each image decreases. Increasing the threshold decreases the sensitivity of the coder to movement and hence yields a lower quality image.

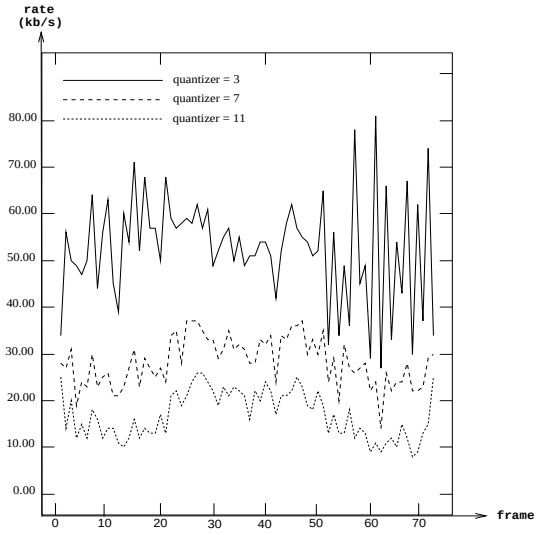


Figure 12: Output rate (kb/s) vs frame for a same video sequence using different quantizers

Within *ivs*, the user can select two different modes: *Privilege Quality* and *Privilege Frame Rate* modes. The mode characterizes what parameters are adjusted in the coder so that the output rate stays below max_rate.[15]

Privilege Quality (PQ) mode is convenient for applications which require high precision in the rendition of individual images (e.g. to transmit slides or still images). In this mode, the values of the quantizer and the movement detection threshold are constant and match the maximal visual quality. Then the coder waits for a sufficient amount of time before encoding the following image so that the output rate stays below the max_rate value.

Privilege Frame Rate (PFR) mode is convenient when the perception of movement is a important factor of quality. The output rate is controlled using different quantizer and movement detection threshold values. We have decided to couple the quantizer with the movement detection threshold using empirical set-ups.

Actually, it is legitimate to couple these two variables since when the quantizer increases, the precision of the rendition decreases, and the likeliness of a large difference between two frames increases. The state of the codec, in PFR mode, is characterized by the target data rate max_rate

[15] The max_rate value is adjustable on the fly by the control congestion algorithm.

Figure 10: CIF image encoded with quantizer values 3 (left) and 7 (right)

Figure 11: CIF image encoded with a quantizer value of 11

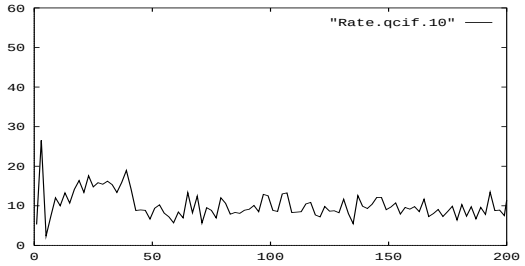
and a couple of quantizer and detector values: several pairs (quantizer, threshold) have been preselected in order to have a linear variation of the output rate. This selection has been obtained by experimentation restricting the quantizer between 3 and 13, and the threshold between 10 and 35.

Since the output rate is rapidly varying, hysteresis is required to prevent undesirable oscillations in the control loop. If the instantaneous rate measured is included in this band, the coupling is preserved. Experiments show that hysteresis of 30% of the maximum bandwidth damps down the output rate. If the instantaneous rate is outside this band, then a new couple is chosen according to the difference between the instantaneous rate and the maximum bandwidth allowed.

The following diagrams have been obtained using a pre-digitized sequence of 200 frames, with QCIF encoding format and three different values of the bandwidth. The first quarter of the sequence is more animated than the rest of the sequence.

Figures 13, 14 and 15 show the instantaneous

Instantaneous rate (kb/s)



Quantizer value

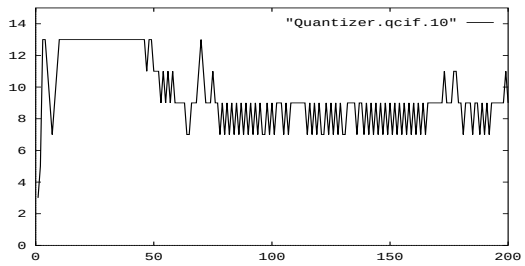
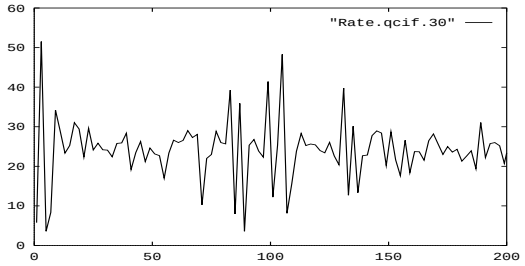


Figure 13: Output rate and quantizer value vs frame number for $max_rate = 10$kb/s



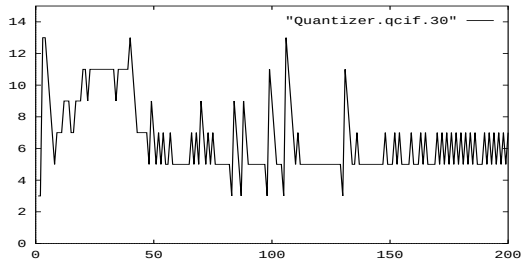
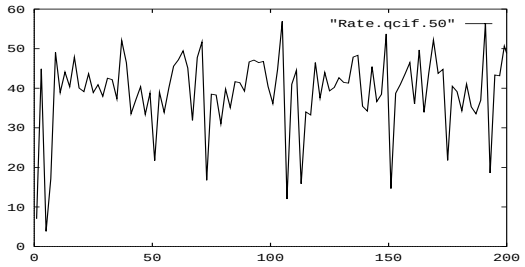


Figure 14: Output rate and quantizer value vs frame number for $max_rate = 30$kb/s



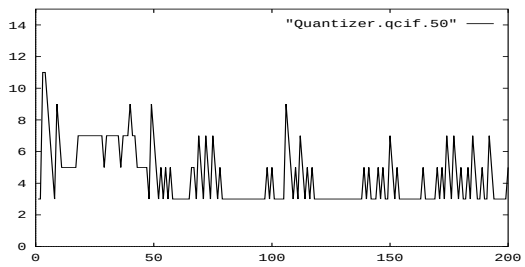


Figure 15: Output rate and quantizer value vs frame number for $max_rate = 50$kb/s

rate and the quantizer with $max\_rate =$ 10kb/s, 30kb/s and 50kb/s, respectively. Figure 16 shows the Signal-to-Noise Ratio (SNR) obtained for these three experiments.

The SNR is a mean-square error measure expressed in dB as shown in equations (1).

$$SNR = -10\log(MSE), \quad \text{with:} \qquad (1)$$

$$MSE = \frac{\frac{1}{JK}\sum_{j=1}^{J}\sum_{k=1}^{K}[G(j,k) - \hat{G}(j,k)]^2}{A^2}$$

where $G(j,k)$ denotes the original luminance value of the pixel (j,k), $\hat{G}(j,k)$ denotes the luminance value of this pixel after encoding/decoding and A denotes the maximum value of $G(j,k)$.

Note that the quantizer selected is inversely proportional to the output rate and that the quality of the image (SNR) is inversely proportional to the quantizer selected. Note also that the quantizer selected is larger during the first quarter of the experiment because the video sequence is more animated, requiring more information to encode.

6 Performance

The output data flow generated by the H.261 coder is non-stationary and rapidly varying. It

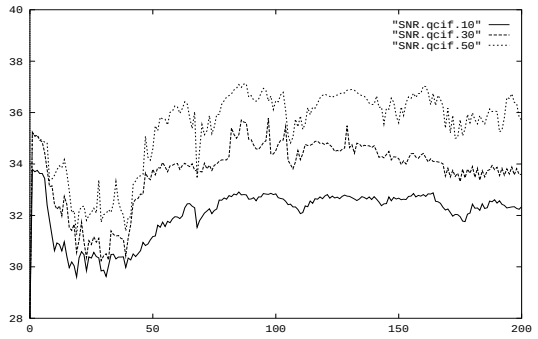


Figure 16: SNR(dB) vs frame number for the three experiments 10, 30 and 50kb/s

depends on the quality of the video camera and the type of the images being encoded, as characterized by the number of scene changes, the scene structure, the scene lighting, etc. It also depends on the definition of the images: CIF or QCIF. We also have to take another element in consideration: the computation power of our codecs. The following experiments have been made on a SPARC IPX work station, with coding and decoding processes running on the same physical machine.

Format	**QCIF**		**CIF**	
Unit	(f/s)	(kb/s)	(f/s)	(kb/s)
Expt. 1	6.0	1.5	3.6	1.5
Expt. 2	4.0	13	2.0	16
Expt. 3	1.9	24	0.5	25

The first experiment is with a still image. When there is no movement, only grabbing and movement detection have to be processed. So, the speed limitation is mainly due to the underlying hardware, in our case the VideoPix board attached to the SparcStation. When there is absolutely no movement, we only encode for each frame the Picture and GOB headers.

The second experiment is characteristic of a classic videoconferencing image, i.e. head and shoulders moving. We can observe that the frame rate is highly dependent on the image size, while the output rate remains almost constant: this is characteristic of a CPU bound application. In fact, the most demanding part of the codec is the computation of the DCT coefficients; the power of the CPU directly limits the number of blocks that can be computed per second, hence limiting the number of coefficients that have to be transmitted and the number of bits sent on the lines.

The third experiment is for a rapidly changing video scene. In such a case, full INTRA mode encoding is chosen in order to suppress accumulation of errors from INTER mode encoding. INTRA mode encoding usage increases the output rate because more coefficients are encoded by blocks. Image rate decreases because all the blocks have to be encoded and more CPU processing is necessary; the data rate increases because the coefficients are much more dispersed than with differential coding, which makes the Huffman compression less efficient.

The frame rate shown in Table 6 is low and might not be suitable for high quality video or remote teaching applications. We found that the frame rate depends on both the video grabbing board and the cpu speed. Therefore, we would expect much better performance with higher performance machines. To illustrate this point, we measured the performance of *ivs* on a SPARCstation 20/501 (bi-processor 2 × 75 MHz) with the VigraPix[16] board. On this platform, version 3.5 of *ivs* is able to encode/decode between 25 and 30 fps in QCIF and between 12 and 30 fps in CIF.

Figure 17 shows the performance obtained both for *ivs*, *nv* and *vic* on a SS 10/20 (41 Mips) platform with the SunVideo[17] board. The video input sequence is very animated (i.e. all the MBs are encoded in each frame) and QCIF color video encoding is selected without video decoding neither local display functions. We used version 2.6 of *vic* to encode both *nv* and *vic-H.261* modes and version 3.5 of *ivs*[18].

The Sunvideo board allows grabbing up to 20 QCIF frames per second on this platform. Note that neither *ivs* nor *vic* can reach this frame rate when the video sequence is very animated. This is due to the high cpu power required for the H.261 compression algorithm. On the other hand,

[16] See URL <http://www.vigra.com/products/vigrapix.announce.html>.

[17] See URL <http://www.Sun.COM/cgi-bin/show?products-n-solutions/hw/wstns/SunVideo.html>.

[18] In this experiment, *ivs* is used with the automatic output rate control disabled.

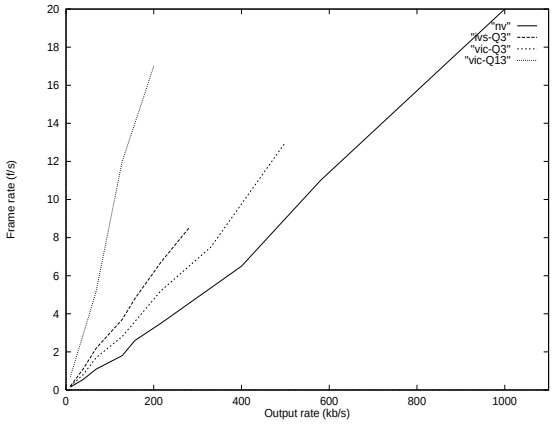


Figure 17: Frame rate (f/s) vs output rate (kb/s)

nv allows obtaining a higher frame rate (in spite of a lower compression rate) because it uses a low computational complexity algorithm. Note that the *ivs-H.261* compression rate is about 30 % higher than *vic-H.261* compression rate with the same (Q=3) quantizer. This is because *ivs* uses the INTER encoding mode on top of the INTRA encoding mode. However, the *ivs-H.261* coder is more greedy of cpu than the *vic-H.261* [only 8.5 frames can be encoded per second instead of 13 for the same quantizer (Q=3)]. Finally, we note that a (Q=13) quantizer gives a compression rate about 4 times higher than a (Q=3) quantizer.

7 Conclusion

In this paper, we described a videoconferencing software application for the Internet. This application, available in the public domain[19] is used over the Internet to hold videoconferences, multicast conferences (e.g. the 4th Joint European Networking conference held in Trondheim, Norway, in May 1993), and weekly MICE[20] seminars [14].

Its main assets are the low bandwidth required, the compatibility with hardware codecs, and the new dimension it brings to classic workstations without high cost, since minimal hardware is necessary. The low bandwidth requirement is very attractive: for instance, low-quality video can be sent on a 9600 b/s link. Moreover, the feedback mechanism we described in this paper allows *ivs* to behave as a "good network citizen."

Further work is ongoing to improve the congestion control algorithm for a heterogeneous multicast environment. Within the Internet, the bandwidth available between several sender-receiver paths can be slightly different. Video gateways can be used to provide different levels of video quality: an application-level gateway for transcoding hardware Motion-JPEG[21] to H.261 video flow has been recently implemented [1]. However, a smarter solution to the problem of multicast video transmission over heterogeneous networks consists of using a hierarchical video coding scheme. In such a scheme, the video is split in several flows: the base flow includes the low resolution information, whereas the enhancement information is sent in additional flows. The idea is to transmit the base flow to all the receivers in the session but to transmit the additional flows only to uncongested receivers. This method will be efficient on the Internet when we are able[22] to associate a higher priority to the essential base flow [30]. We are currently working on a wavelet-based video coding scheme which we expect to include in *ivs*.

Acknowledgments

We are grateful to Steve McCanne, Steve Casner and Mark Handley for providing helpful comments on the H.261 packetization scheme. Jean Bolot contributed to improve the congestion control scheme described in this paper. We thank the MICE research team and the MBone community who have kindly tested the ivs implementation, reported bugs and provided support for new platforms. Finally, we would like to thank Henry Houh and the anonymous reviewers for their constructive feedback.

[19] *ivs* sources and binaries are available by <ftp://zenon.inria.fr/rodeo/ivs/last_version>. See also URL <http://www.inria.fr/rodeo/ivs.html>.

[20] MICE stands for Multimedia International Conferencing for European Researchers. MICE is an European project, which aims at providing appropriate multimedia, multiparty conferencing to researchers in Europe. See also URL <http://www.cs.ucl.ac.uk/mice/mice.html>.

[21] Motion-JPEG stands for JPEG coding applied to moving images. It is based on INTRA coding without movement detection.

[22] Next IP generation (IPng) specifies a priority mechanism associated to the packets sent from a same source using the TCLASS field [23].

References

[1] Amir, E., McCanne, S., and Zhang, H. "An Application-level Video Gateway", *ACM Multimedia '95*, San Francisco, Nov. 1995.

[2] Aravind R. et al. "Image and video coding standards", *AT&T Tech. Journal*, pp. 67-89, Jan/Feb. 1993.

[3] Bolot, J.C. "End-to-end packet delay and loss behavior in the Internet", *Proc. ACM Sigcomm '93*, pp. 289-298, San Fransisco, CA, Sept. 1993.

[4] Bolot, J.C. and Turletti, T. "A rate control mechanism for packet video in the Internet", *Proc. IEEE Infocom '94*, pp. 1216-1223, Toronto, Canada.

[5] Bolot, J.C. and Turletti, T. "Scalable feedback control for multicast video distribution in the Internet", . *Proc. ACM Sigcomm '94*, Vol. 24, No 4, pp. 58-67, Oct. 1994.

[6] Casner, S. and Deering, S. "First IETF Internet audiocast", *ACM CCR*, Vol. 22, No 3, July 1992.

[7] "CCIR recommendation 601-2: Encoding parameters of digital television for studios", *International Telecommunication Union*, 1990.

[8] Clark, D. "The Design Philosophy of the Darpa Internet Protocols", *Sigcomm '88 Symposium*, pp. 106-114, Stanford, CA, Aug. 16-19, 1988.

[9] Clark, D. and Tennenhouse, D.L. "Architectural considerations for a new generation of protocols", *Proc. ACM Sigcomm '90*, pp. 200-208, Sept. 1990, Philadelphia.

[10] Deering, S. "Multicast routing in a datagram internetwork", *Phd dissertation*, Stanford University, California, Dec. 1991.

[11] Frederick, R. "Experiences with real-time software video compression", *Sixth International Workshop of Packet Video*, pp. F1.1-1.4, Portland, OR, Sept. 1994.

[12] Gilge, M. and Gusella, R. "Motion video coding for packet-switching networks - An integrated approach", *Proc. SPIE Conference on Visual Communications and Image Processing*, Boston, MA, Nov. 1991.

[13] Guichard J., Eude, G. and Texier, N. "State of the art in picture coding for low bitrate applications", *Proc. ICC '90*, pp. 120-125.

[14] Handley, M., Kirstein, P.T. and Sasse, M.A. "Multimedia integrated conferencing for European researchers (MICE): piloting activities and the conference management and multiplexing centre", *Computer Networks and ISDN Systems*, pp. 275-290, vol. 26, No 3, Nov. 93.

[15] Huitema, C. and Turletti, T. "Software codecs and work station video conferences", *Proceedings of INET '92*, pp.501-508, Kobe, Japan.

[16] Kanakia, H., Mishra, P. and Reibman, A. "An adaptive congestion control scheme for real-time packet video transport", *Proc. ACM Sigcomm '93*, pp. 20-31, San Fransisco, CA, Sept. 1993.

[17] Keshav, S. "A control-theoretic approach to flow control", *Proc. ACM Sigcomm '91*, pp. 1-11, Zurich, Switzerland, Aug. 1991.

[18] LeGall, D. J. "MPEG: A video compression standard for multimedia applications", *Communication of the ACM*, No 4, Apr. 1991.

[19] Liou, M. "Overview of the $p \times 64$ kb/s video coding standard", *Communication of the ACM*, No 4, Apr. 1991.

[20] "Coding of moving pictures and associated audio (MPEG)", *ISO/IEC JTC1 SC29*.

[21] "MPEG 2 Video standard", *ISO/IEC 13818-2*.

[22] Rao, K. R. and Yip, P. "Discrete Cosine Transform :Algorithms, Advantages, Applications", *Academic Press Inc*, 1990.

[23] Bradner, S. and Mankin, A. "The recommendation for the IP Next Generation protocol", *RFC 1752*, Jan. 1995.

[24] "$p \times 64$ Channel frame structure for audio/video conferencing", *ITU-T Recommendation H.221*, 1993.

[25] "Video codec for audiovisual services at $p \times 64$ kb/s", *ITU-T Recommendation H.261*, 1993.

[26] Macedonia, M. R. and Brutzman, D. P. "MBone provides audio and video across the Internet", *IEEE Computer Magazine*, pp. 30-36, Apr. 1994.

[27] McCanne, S. and Jacobson, V. "vic: a flexible framework for packet video", *ACM Multimedia*, pp. 511-522, San Francisco, CA, Nov. 1995.

[28] Sanghi, D., Agrawala, A.K. and Jain, B. "Experimental assessment of end-to-end behavior on Internet", *Proc. IEEE Infocom '93*, pp. 867-874, San Fransisco, CA, Mar.

1993.

[29] Schulzrinne, H., Casner, S. Frederick, R. and Jacobson, V. "RTP: A Transport Protocol for Real-Time Applications", *RFC 1889*, Nov. 1995.

[30] Turletti, T. and Bolot, J-C. "Issues with multicast video distribution in heterogeneous packet networks", *Proc. 6th International Workshop on PACKET VIDEO*, pp. F3.1-3.4, Portland, Oregon, 26-27 Sept. 94.

[31] Turletti, T. "The INRIA Videoconferencing System (IVS)", *Connexions Magazine*, pp. 20-24, Oct. 1994.

[32] Turletti, T. and Huitema, C. "RTP Payload format for H.261 video streams", *RFC TBD*, 1996.

[33] Wakeman, I. "Packetized video - Options for interaction between the user, the network and the codec", *The Computer Journal*, vol. 36, No 1, 1993.

[34] Zhang, L., Deering S., Estrin, D., Shenker, S. and Zappala, D. "RSVP: a new resource reservation protocol", *Proc. IEEE Network*, Sept. 1993.