

# ***High Quality Video Delivery over Local Area Networks***

## ***With Application to Teaching at a Distance***

**Chris Jesshope and Yong Qiu Liu**

E-mail: [C.R.Jesshope@massey.ac.nz](mailto:C.R.Jesshope@massey.ac.nz) [Y.Liu@massey.ac.nz](mailto:Y.Liu@massey.ac.nz)

Institute of Information Sciences and technology  
Massey University, Private Bag 11222  
Palmerston North, New Zealand

### ***Abstract***

Multimedia communication over networks is an important topic in distance learning, computer-based multimedia and telecommunications. The quality of the video stream, the throughput of the communication and the time delay of the delivery are the important issues that must be considered. This paper describes some experimental work undertaken to optimise the issues that influence data loss and hence image quality, and transfer delay in MPEG video streams transferred over Local Area Networks (LAN). It describes an application in some detail and analyses it in terms of its periodical loop and the frequencies of its threads' loops, which are sending and receiving data to or from network.

Keywords: Video, Multimedia, MPEG, IP Networks, Education

## ***1. Introduction***

### ***1.1 Video communication for distance education***

For some time now Massey University has had a large cohort of extramural students and traditionally the extramural papers have been taught with paper-based study guides and little contact with the lecturers. This means that our distance students are really second class citizens, unless they can attend on-campus block courses. An alternative is to bring the lectures to the students using modern technology. Massey is also a multi-campus University and like any university, is trying to ensure that all papers are delivered in an economically viable manner. Often for specialist papers the numbers of students on one campus are not economic but if papers could be delivered simultaneously to all campuses economies of scale can be achieved, providing the delivery is cheap and does not involve significant additional overhead. It can be seen therefore that the use of technology in teaching is of great interest to staff at Massey University. Indeed a number of experiments have been conducted recently in offering live lectures at a distance using compressed video over both local and wide area networks. Some results of this work have been published in (Pearson and Jesshope 98).

In this experiment, lectures were delivered at a distance, between two campuses, using compressed video over an experimental ATM network. A bi-directional link was used, which meant that the lecturer had visual/audio feedback from the remote class. The video

was compressed by hardware, which used the motion JPEG standard. Three major problems were encountered in this work. The first was that the motion JPEG standard has relatively poor compression ratios (typically from 3:1 to 5:1). Although this was acceptable using a large bandwidth on the experimental network, it would have been costly had the network usage been charged for at commercial rates (a bandwidth of 10-20 Mbits per second in each direction was used). Secondly a significant additional overhead in terms of the delivery of the material. For example a technician was required to mix and produce the live feed that was transmitted across the network. The final problem was in the delivery of high quality presentation graphics to the remote class. Using the video channel as a carrier for presentation graphics was found to be a poor solution. High resolution static images are reduced in quality significantly and, of course, could be transmitted at significantly reduced bandwidth using static image compression techniques, such as the PNG standard (Roelofs 2000).

It was because of these limitations that we started looking at different solutions for the delivery of lectures at a distance. There were then a number of commercially available solutions that provided low-resolution, low-frame-rate video over IP networks. These however, did not provide the quality of lecturing experience that we required. We were interested in transferring broadcast quality video and audio in both directions. Moreover, the application developed should enable the lecturer to monitor and direct his presentation at a distance without the assistance of additional technical staff. Again there were video conferencing solutions that would have satisfied most of our constraints. These solutions were very expensive however and had one major limitation. In lecturing, one of the major issues is the delivery of clear presentation graphics. The video conferencing solutions have no mechanisms to deliver this, requiring the presentation graphics to be delivered via the video stream, which is wasteful of bandwidth and delivers a very poor quality presentation to the remote students.

We had already developed a method of recording, compressing and playing back presentations that comprised images, vector graphics and sound. This development was for asynchronous teaching and is described in detail by Jesshope, Shafarenko and Slusanschi (1998). It would be relatively easy for us therefore to use a network connection instead of a file as the link between lecturer and class and this would then provide a channel for the delivery of high-quality, synchronous presentation graphics and annotations. Thus all we were left with the problem of the video compression and delivery and a user-friendly application to tie this all together.

## **1.2 Video communications over LAN**

Video communications over IP networks is a rapidly developing field that has many applications, such as: internet telephony, desktop video conferencing, collaborative computing, business conference calling, support and help desk applications, interactive Web shopping as well as our requirement for distance learning, see Adie (1993) for a survey.

To summarise the technical process, video pictures must be captured, digitised, compressed, transmitted, decompressed and finally displayed on a monitor or screen. Because we are dealing with IP networks, the compressed video stream must be divided into small packets before being sent them on to network, usually using the UDP protocol, which unlike the TCP protocol, is not checked for delivery and hence packets may be lost, see Ziyang Shao (1998) for details on IP protocols. At the other end of the network, the packages are received by the UDP endpoint, are reassembled and decompressed to reconstruct the

video stream, which may be degraded due to packet loss. This is a one-way multimedia communication over network. For two-way communication, each end has the ability of both sending and receiving video streams. The quality of the video communication is depended on the video image size, its resolution, frame speed, data lost and in two way communications, the latency of communication.

### **1.3 Why Use Video Compression?**

For the high quality video communication, the size of the image should be big enough to be seen by the communicators and the frame speed should be fast enough to represent the moving pictures. Because of these two aspects, the throughput of the data will be large. For example, if we want to display a movie of 640 \* 480 size with full resolution, the frame rate is 50 frames per second and we use 24 bit colour, then throughput required to transmit such a video stream, without compression, would be 387 Mbits per second. Bidirectional video requires this bandwidth in each direction. This is a very large throughput and would very quickly saturate most networks. To reduce the burden on the network, some form of compression is required. The video stream is encoded into compressed form and then decoded to regenerate the original video stream. The compression ratio obtained varies for different methods of compression. For example, the motion JPEG compression we have already tried gives from 3:1 to 5:1 compression, depending on quality required. Even with the same compressing method, the ratio varies with different contents of the images.

One of the emerging standards for digital video broadcast and transmission is the MPEG standard. This method not only compresses each frame as an image, but also attempts to gain greater levels of compression by encoding only those parts of the image that change between frames, see Filippini (1997) and Wired Inc. (1999) for more information on the Motion Picture Experts Group (MPEG) standard. MPEG will generally achieve compression ratios of up to 100:1. With this compression ratio, the throughput of 351.6 Mbit/s can be reduced to 3.5 Mbit/s, which is feasible on modern local area networks. Thus a codec is an essential way to ensure the large video data can be transferred over network. (A codec is the shortened expression for an encoder/decoder). The codec we use in our experiments was implemented in hardware and available as a pair of cards to encode and decode from Wired Inc (1990).

### **1.4 Factors which influences the video transfer over network**

There are many factors that influence video communication accomplished by this kind of transmission. Data loss is just one of the factors that affects the quality of the video transmission. Many video compression protocols permit a portion of data to be lost, without degrading the picture significantly. MPEG is one such example, the image can be reconstructed from a previous frame is a loss does occur. However, too much loss will cause the picture to break up or to jump discontinuously. Another factor that influences the quality of video transmission is transmission latency. This is particularly important in two-way interactive situations, where say, a lecturer is engaged in a discussion with a student. If the delay is large, it is very difficult to participate in such an exchange. Encoding, decoding, network transmission and buffering delays all contribute to the perceived transmission delay or latency. As we have seen, for high-quality video communication, where large amounts of data is transferred over network, the use of encoding and decoding is mandatory. We use hardware encoding in order to minimise these delays. It is important therefore to optimise the other delays, such as buffering and network transmission.

## **2. The Structure of the LectureLink application**

Before describing our experiments on the optimisation of the transmission of video over IP, it is important to understand the structure of our application and the details of the environment in which it is being used, for example, details of our host computers and physical properties of our network.

### **2.1 Application environment**

We have been using Power Macintosh computers, a 7600 and an 8600 model, both with 200 MHz CPU speed and 64 MB built-in memory. The operating system is Mac OS 9 with QuickTime 4. For the video codec, we use a Butane II MPEG2 encoding card and a Mason X MPEG decoding card, manufactured by Wired Inc. Both cards are PCI cards with external connections for both composite and s-video input and output. For input we use a Sony Hi 8 mm video camera and for output either a TV monitor or the AV input to the Macintosh 8600 computer. In the latter case we can display the video received on the Macintosh screen. The cards support either MPEG1 or MPEG2 encoding and decoding. The choice is software switchable. MPEG1 has maximum resolution of  $352 * 288$  at a frame rate of 25 frames per second with an average throughput of 1.5 Mbps whereas MPEG2 has maximum resolution of  $720 * 576$  at a frame rate of 50 frames per second with the average throughput of 6 Mbps.

### **2.2 The LectureLink Application**

The LectureLink is an application written in C++, using CodeWarrior and its PowerPlant application framework. The software provides a control channel between the lecturer and the remote computer for the remote control of the display and has the ability to transmit video streams in both directions simultaneously. Each video stream can be set in terms of resolution, frame rate and transfer rate. Thus there are two main threads, one that reads the Butane II card and transmits data and one that receives data and writes to the Mason X card for display. The Transfer protocol used is UDP on 10 base 2 Ethernet which has bandwidth of 10 Mbits/s. UDP does not require the acknowledgement of each packet transmitted and thus permits a portion of the data to be lost. This is a good strategy and retransmission does not make sense for a real-time application, especially as the video stream can maintain good quality in the presence of small data losses.

The main application, like all PowerPlant applications, comprises a periodical loop. Within this loop events are tracked, including system events, displaying a video preview, decoding the video data, encoding the video data, threads loops and updating the information window. The threads loops include the loops of waiting for a new client connection, receiving data and sending data, see figure 1. These loops are executed several times within one periodical loop.

The repetition period time of the periodical loop is a very important parameter for the data transmission, because encoding the outgoing data and decoding the incoming data must both be processed in this period. Another important parameter is the thread scheduling rate. For example, if the frequency of the sending thread is too slow, the encoded data will not get enough time to be sent and will be accumulated at the sending end. On the other hand, if the frequency is too fast, there is not enough encoded data available to be sent. An empty loop provides wasteful overhead to the transmission and increases the end-to-end latency. The same is true for the receiving end. The structure of the application is shown in figure 1.

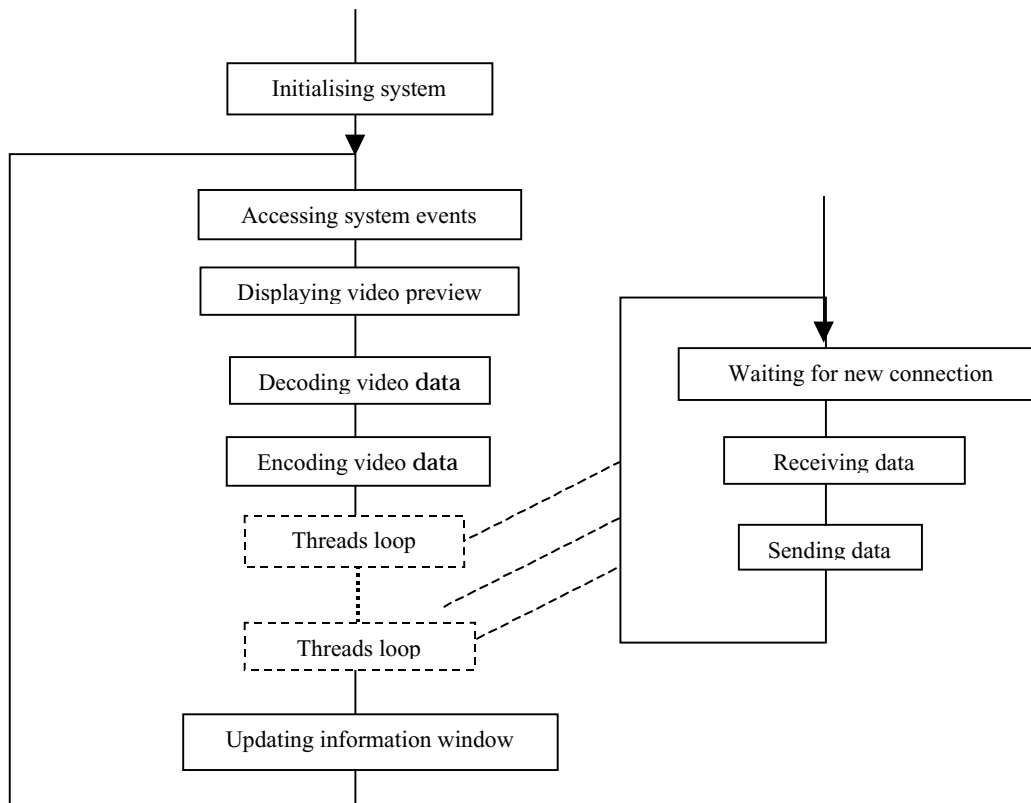


Figure 1 Periodical and threads loop

### 3. Periodical Analysis

In this section, we will analyse the efficiency of the application through monitoring the application. Using these experiments, it is possible to deduce the frequency of the encoding and decoding processes and the frequency of the threads which send and receive the data.

#### 3.1 Periodical frequency test

The aim of this experiment was to record and compare the frequencies of the periodical loop under a range of different conditions. This was required to obtain the right reference data for the subsequent analysis of the transmission problems. To achieve these three types of data transfer were measured:

**Self-transfer:** this is a transfer that sends a video stream from one computer to itself. In this situation, there is no actual data sent through network. That means the transfer channel is an ideal one. There is no data lost, no data conflict, no transfer delay and no practical bandwidth limit.

**Two way transfer:** this is a transfer between two computers connected using a dedicated Ethernet segment, where both computers are sending and receiving video streams to each other in real time. This is the worst case scenario, we have data loss, data transfer conflict, transfer delay and of course a bandwidth limit.

**One way transfer:** this is a transfer between two computers connected together over Ethernet, where just one computer sends a video stream to the other again in real time. In this situation, there is no data transfer conflict but the other limitations remain.

In the periodical frequency test, different encoding data bit rates were used from 0.8 Mbits/s to 8.0 Mbits/s to compare the periodical frequency under different data loads. The test result is shown in Table 1.

**Table 1 Periodical frequency comparison**

Bit Rate (Mbps)	0.8	1.5	3.0	4.5	6.0	8.0
Transfer type	Frequency (Hz)	Frequency (Hz)	Frequency (Hz)	Frequency (Hz)	Frequency (Hz)	Frequency (Hz)
Self transferring	52	55	53	49	49	42
Two way	59	61	56	49	41	34
One way(Broadcaster)	56	56	56	53	48	44
One way(Receiver)	62	65	69	73	72	71

From Table 1 we can see that with the increasing of the encoding bitrate, the periodical frequency of two way transfer, self transfer and one way transfer (at the sending end) is decreased. This is because high bit-rate encoding needs more time to process and the frequency of the application decreases in inverse proportion. In the receiving end of a one way transfer, there is no encoding action and the periodical frequency does not decrease.

When the encoding bit rate is between 0.8 to 8.0 Mbits/s, the periodical frequency of way transfer is in a range from 34 to 59 Hz. That means their periodical intervals (especially the encoding interval and decoding interval) are between 29.4 to 16.9 ms respectively. Whereas the periodical frequency of one way transfer on the receiving end is from 62 to 73 Hz or the interval the interval between two decode actions is between 16.1 to 13.7ms.

### 3.2 Data transfer test

In this experiment, we record the frequency of data sending and receiving in a two-way transmission, to see how often the data is sent and received in the application. The results of this experiment are shown in Table 2, where "Thread frequency" in Table 2 means the frequency of the threads which send or receive data to/from network. For the sending thread if there is data available, it sends the data through the UDP endpoint. Otherwise it will perform an empty loop. Similarly, the receiving thread tries to receive data from network. If the data is not available, it just timeouts and quits.

**Table 2 Frequency comparison  
(Transfer threads in two way transfer)**

MPEG1 / MPEG2	Buffer size: 72KBytes		Test period: 100 s		Self transfer	
Bit Rate (Mbps)	0.8	1.5	3.0	4.5	6.0	8.0
Buffer size (Kbytes)	Frequency(Hz)	Frequency(Hz)	Frequency(Hz)	Frequency(Hz)	Frequency(Hz)	Frequency(Hz)
Thread frequency	66	68	63	54	50	47
Sending frequency	4	9	16	24	32	42
Receiving frequency	4	8	16	22	23	23

Note that the "Sending frequency" is the frequency that the sending thread actually sends available encoded data to the network. The empty loop is not taken into account. Similarly the "Receiving frequency" is the frequency that the receiving thread receives the available encoded data from the network, again the empty loop is not taken into account.

From these results we can see the following:

- Not surprisingly the thread frequency decreases with the increasing of the encoding bit rate. This is because a larger data encoding needs more time. Also the total periodical period increases and hence the thread frequency decreases. With the encode bit rate increasing from 0.8 to 8.0 Mbits/s, the frequency of the threads decrease from 66 to 38 Hz.
- Although the frequency of the threads is high, the frequency of the sending thread, which sends data to the network, is from 4 to 28 Hz when the encode bit rate is from 0.8 to 8.0 Mbits/s. This means that most of the time, the sending thread performs an empty loop. In another words, the frequency of the sending thread is fast enough to send the encoded data to the network.
- Similarly the frequency of the receiving thread which receives data from the network is between 4 to 23 Hz when encode bit rate is from 0.8 to 8.0 Mbits/s. This means that most of the time, the receiving thread did not receive data and performed an empty loop. In another words, the frequency of the receiving thread is also fast enough to receive the encoded data from the network. Unfortunately, some data is being lost during the transmission. This will be discussed in the following section.

## **4. Data Loss**

Data loss is one of the most common problems during video communication. Excessive data lost will cause pictures to jump or the audio stream to be lost. There are many factors that cause data loss. In respect of the application, data loss could be caused by improper frequency of the encoding function, the sending or receiving thread. Another factor is the encoded data's buffer size. In respect of the network, it could be caused by a limit in bandwidth, data conflict and transfer protocol. This experiment is designed to find some optimal perimeters for the encoding data buffer size and transfer throughput to eliminate the data loss.

### **4.1 Transfer bandwidth and encoding bit rate**

Bandwidth is a hard barrier for large-scale data transfer over network, as increasing the bandwidth means increasing your investment in communication infrastructure. Sometimes improving the program to suit the existing bandwidth is much easier than obtaining funding to improve infrastructure. In the case of our application the encoding bitrate can be set by the user, to match the available bandwidth. The other parameter that can be adjusted is the data buffer size, which is critical to the efficient running of the application. This experiment is to determine the buffer size to set in the program, as a function of encoding bit rate.

Table 3 gives the results of this experiment again using three types of transmission, two-way transfer, one-way transfer and self-transfer.

**Table 3 Data lost in different encoding bit rate during transfer**

Encoding bit rate	0.8(Mbps)	1.5(Mbps)	3(Mbps)	4.5(Mbps)	6(Mbps)	8(Mbps)
Self transfer	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
One-way transfer	0.0%	0.0%	0.1%	1.2%	2.3%	5.1%
Two-way transfer	0.3%	1.1%	1.1%	8.1%	26.0%	34.8%

From Table 3 we can see that when the encoding bit rate exceeds 4.5 Mbits/s for the two-way transfer, the data loss increases rapidly. This is because the bandwidth of our Ethernet is 10 Mbits/s. For two-way transmission with the same data throughput in both sides, the theoretical limit for throughput from each side is 5 Mbits/s. In practice, protocol overhead and collision of packets reduce this. Clearly this must limit the encoding bit rate.

We can also see that when the encoding bit rate is less than 3 Mbits/s data loss on transfer for both one-way or two-way transfers is less or equal to 1.1% and in one way transfer is less than 0.1%. Of course a large encoding bit rate will cause a large data throughput but although the UDP protocol permits a portion of data to be lost, as long as the throughput is not close the limit of the network bandwidth and providing the implementation of the application is efficient, it has been shown that data loss can be constrained within tight bounds.

#### 4.2 Influence of the encoded data buffer size

When Butane video encoder card captures the video stream, it is encoded and then placed into memory buffers called video buffers. Similarly, the audio stream is also captured and encoded and then put into another set of memory buffers called audio buffers. If a multiplexed stream (with both video and audio signals) is demanded, the encoding processor checks the two sets of buffers every 0.25 s. If at least one video buffer and one audio buffer are full, it will combine them to a system multiplex stream and put it into a third set of memory buffers called multiplexed buffers.

During the communication, both audio and video streams are captured continuously and encoded by the Butane card, which places the data in the audio and video buffers. This data is then used to generate a system or multiplexed stream, which is put into the multiplex buffers from which the UDP endpoint thread will send it out to the network. This is illustrated in figure 2.

The size of the three buffers is very important for the encoding and data transfer since during transmission, no buffer should overflow or data loss will occur. This is important, as for the synchronisation of sound and video stream, only the same time-scaled audio and video encoded data can be mixed to create a multiplex stream. The multiplex encode can be performed only when at least one audio buffer and one video buffer are available. On the other hand if the buffers are made unnecessarily large, this will lead to a greater encoding latency. Thus the absolute and relative size of the buffers in each list (see Figure 2) is very important.

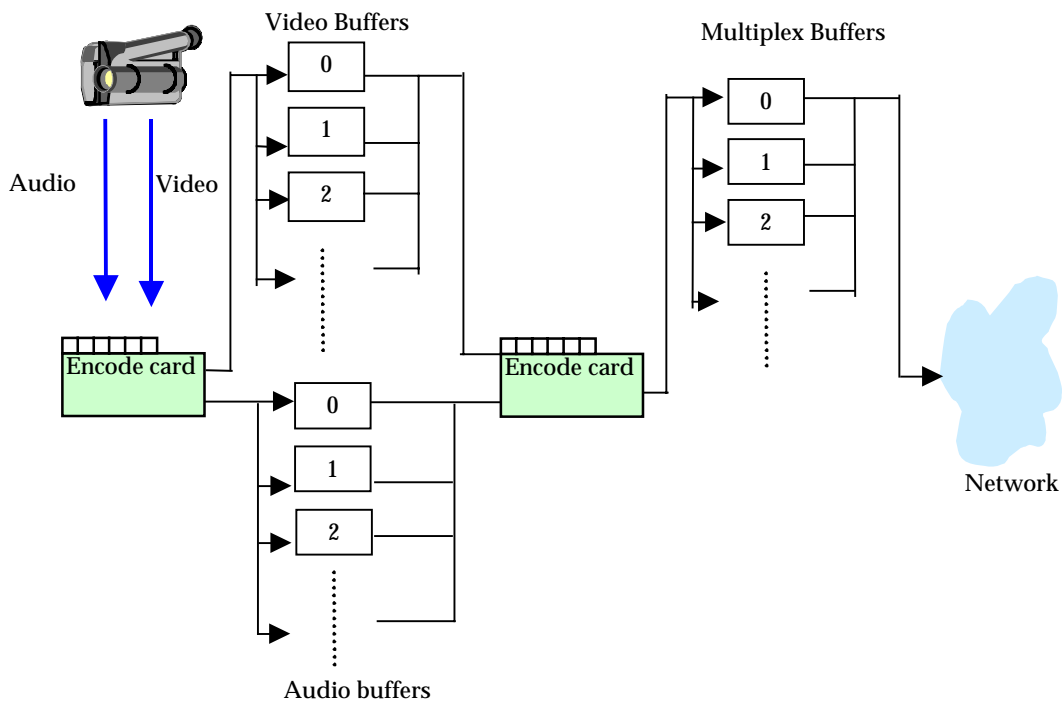


Figure 2 Three lists of encoded data buffers

For example, consider the following: video encoding bit rate of 6 Mbit/s, an audio encoding bit rate of 192 Kbits/s, 10 video buffers each with a size of 800 Kbits and 10 audio buffers each with a buffer size of 260 Kbits. To fill one audio buffer requires  $260/192 = 1.35$  s. But to fill all the video buffers will require just  $10 * 800 / 6 * 1024 = 1.3$  s. In this situation overflow occurs, because the audio buffers are too large and the first audio buffer has not be filled when all the video buffers have overflowed.

Because the encoding bit rate can be changed by the user dynamically from 0.2 to 10 Mbits/s, the size of the three buffers should be adjusted dynamically. If we select total size of the video buffers to be 16 Mbits, the audio buffers to be 1,344 Kbits then in each 0.25 s at least one video buffer and one audio buffer will become full with encoded data. The size of each buffer used is listed in Table 4.

**Table 4 The buffer sizes in different encoding bit rate**

Encode bit rate			Video buffers			Audio buffers			Multiplex buffers		
Total (Mbps)	Video (Kbps)	Audio (Kbps)	Total size (Mbits)	Buffer size (Kbits)	Number of buffers	Total size (Mbits)	Buffer size (Kbits)	Number of buffers	Total size (Kbits)	Buffer size (Kbits)	Number of buffers
0.8	627.2	192	16	157	104	1.3125	48	28	205	192	2
1.5	1344	192	16	336	49	1.3125	48	28	384	192	2
3.0	2880	192	16	720	23	1.3125	48	28	768	192	4
4.5	4416	192	16	1104	15	1.3125	48	28	1152	192	6
6.0	5952	192	16	1488	11	1.3125	48	28	1536	192	8
8.0	8000	192	16	2000	8	1.3125	48	28	2048	192	11

It is the multiplex buffers that are the most critical. Whenever all the multiplex buffers are full, their data will be sent to the network by the UDP endpoint. Otherwise the data will be kept in them. To eliminate the time delay, the total size of the multiplex buffers should be as small as possible. Also, for the best performance in network transmission, it is better to send a small amount of data regularly to the network, at short intervals, rather than send large amount of data at long intervals. This minimises collisions on the network. The size of the multiplex buffers is selected to be the sum of a single audio buffer and a single video buffer.

We can also reduce the total size of all of the audio and the video buffers to save the resources. If we restrict the total size of the audio and the video buffers to hold just 2 seconds of encoded data, we obtain the optimised buffer sizes (Table 5).

**Table 5 The buffer sizes in different encoding bit rate (Optimised)**

Encode bit rate			Video buffers			Audio buffers			Multiplex buffers		
Total (Mbps)	Video (Kbps)	Audio (Kbps)	Total size (Mbits)	Buffer size (Kbits)	Number of buffers	Total size (Kbits)	Buffer size (Kbits)	Number of buffers	Total size (Kbits)	Buffer size (Kbits)	Number of buffers
0.8	627.2	192	1	157	8	384	48	28	205	192	2
1.5	1344	192	3	336	8	384	48	28	384	192	2
3.0	2880	192	6	720	8	384	48	28	768	192	4
4.5	4416	192	9	1104	8	384	48	28	1152	192	6
6.0	5952	192	12	1488	8	384	48	28	1536	192	8
8.0	8000	192	16	2000	8	384	48	28	2048	192	11

Using the CodeWarrior, debugging facility, the frequency of writing encoded data into the multiplex buffer is recorded. Also, the average number of buffers being written a second is also recorded. Both measurements are given, along with different encoding bit rate, in Table 6.

**Table 6 Multiplex data writing in different encoding bit rate**

Encoding bit rate (Mbps)	0.8	1.5	3.0	4.5	6.0	8.0
Writing frequency (Hz)	5	9	10	10	12	17
Number of buffers(Hz)	5	9	17	25	33	33

From Table 6 we can see that the writing frequency and the number of the multiplex buffers being written to increase, as expected, with increasing encoding bit rate. We also notice that the number of buffers does not increase when the encoding bit rate increases from 6 to 8 Mbits/s. This is because in the debug mode, the speed of the application is less than that in the normal condition. The sending thread's sending frequency is not fast enough to send all data in the multiplex buffers to the network. Overflow will occur, but this will not happen during normal transmission.

## 5. Transfer Latency

Transfer latency is a major issue in the video communication, especially for two-way interactive communication. We can not remove the time delay completely because it is an unavoidable problem during encoding, transfer and decoding. This experiment establishes some optimal parameters to minimise the time delay.

The video communication can be divided into three parts, encoding, transfer and decoding. So the time delay comprises the sum of encoding delay, transfer delay and decoding delay. There are two distinct components in the encoding delay. One is the encoding processing delay and the other is the waiting delay before the encoded data is put in the buffer to await transmission. Similarly, there are two components to the decoding delay. They are decode waiting, where data is waiting for being decoded, and decode processing.

Table 7 below show some statistics for the total delay, the send waiting delay, the decode waiting delay and other delays (the sum of the encode processing, transfer and decode processing delays).

**Table 7 Two-way transfer time delay**

Encoding bit rate (Mbits/s)	Total Delay (s)	Sending waiting delay (s)	Decoding waiting delay (s)	Other delay (s)
0.8	2.83	0.37	2.14	0.32
1.5	1.6	0.27	1.09	0.24
3.0	1.1	0.26	0.56	0.26
4.5	0.9	0.29	0.41	0.22
6.0	0.9	0.37	0.38	0.16

8.0	1.0	0.52	0.31	0.19
-----	-----	------	------	------

From Table 7 we can make following observations.

- On average, the other delays (sum of encode processing, transfer and decode processing delays) consume less time than send waiting delay and decode waiting delay, i.e. less than 1/3 s.
- The decode waiting delay is longer when the encoding bit rate is smaller.
- The send waiting delay, like other delays, remains relatively constant for all encoding bit rates.
- When the encoding bit rate exceeds 3.5 Mbits/s, all time delays are reasonably stable and the total time delay is less than (or equal to) 1 s.

## Conclusion

We have described and analysed an application for transferring MPEG encoded data over 10 Mbps Ethernet, using the UDP over IP protocol. Our analysis has shown that the application's periodical frequency (34 to 59 Hz) is fast enough for updating the encoded data buffers. The minimum frequency should not be less than 17 Hz. We also show that the sending and receiving thread's frequencies (47 to 66 Hz) are also faster than the actual frequency of sending (4 to 42 Hz) or receiving (4 to 23 Hz) data. We note that the minimum frequency of the sending thread should not be less than 42 Hz, in order to ensure it can send data from the multiplex buffers to the network on time, specially when other application are running simultaneously.

We have also shown that the application can consume a high percentage of the bandwidth of the Ethernet in a two-way transfer situation. Clearly the total throughput must be less than the limit of the bandwidth of the network. But for example, if 10 base 2 Ethernet is used, two-way transfer data loss is to be only 1.1% when the encoding bit rate is 3 Mbits/s and one-way transfer data loss is to be only 2.3% when the encoding bit rate is 6 Mbits/s.

We have demonstrated that the size of each buffer in the three lists of encoder buffers, the video buffers, the audio buffers and the multiplex buffers, will influence the transfer data lost as well as transfer time delay. The optimised buffer sizes are given in [Table 5](#).

Finally, concerning latency, we have shown the major contribution to be the decode waiting delay, which increases rapidly when the encoding bit rate is less than 3 Mbits/s. We also note that when the bit rate exceeds 3.5 Mbits/s, there is no significant change to the latency.

Thus we have demonstrated that our application is capable of sending MPEG encoded video over Ethernet and when optimised, we can bound the latency of end-to-end transmission to about 1 s, while still respecting the bandwidth constraints on the network. All experiments were carried out on an otherwise quiescent Ethernet.

## References

Chris Adie (1993) A Survey of Distributed Multimedia Research, Standards and Products, retrieved from the World Wide Web, 15/2/00 <http://cui.unige.ch/OSG/info/MultimediaInfo/mmsurvey/>

Guy Cote, and Faouzi Kossentini (1998) H.263+: Video Coding at Low Bit Rates, *IEEE Transactions On Circuits And Systems For Video Technology*, **8** (7) November 1998 p849.

Luigi.Filippini (1997) MPEG Moving Picture Expert Group - Frequently asked questions, retrieved from the word-wide web on 17/2/00, <http://www.crs4.it/~luigi/MPEG/mpegfaq.html>

C. R. Jesshope, A. Shafarenko and H. Slusanschi (1998) Low-bandwidth multimedia tools for web-based lecture publishing, *IEE Engineering Science and Educational Journal*, 7 (4), pp148-154.

M. Pearson and C. R. Jesshope (1988) Multi-campus teaching using computer networks, *Proc. of the Third Australasian Conference on Computer Science Education* pp 106 - 111, July 1998 (Association for Computing machinery Inc.).

Leigh Anne Rettinger (1995), *Desktop Videoconferencing: Technology and Use for Remote Seminar Delivery*, Thesis downloaded from the World Wide Web on 15/2/00, [http://www2.ncsu.edu/eos/service/ece/project/succeed\\_info/larettin/thesis/](http://www2.ncsu.edu/eos/service/ece/project/succeed_info/larettin/thesis/)

Roelofs, G (2000) Portable Network Graphics, Retrieved from the web on 1/2/00, <http://www.cdrom.com/pub/png/>

Wired Inc. (1999) Wired Incorporated MPEG FAQs (Frequently Asked Questions), retrieved from the world-wide web on 15/2/00, <http://www.wiredinc.com/faq.html>

Wired Inc. (1990) Products, retrieved from the world-wide web on 15/2/00, <http://www.wiredinc.com/product.html>

Ziying Shao (1998) Multimedia Communication, accessed from the World Wide Web on 15/2/00, <http://www.cis.udel.edu/~zshao/content.html>