J. Indian Inst. Sci., Mar –Apr. 2000, **80**, 129–141.

A desktop video telephony over IP

P. VICTOR ANANDA RAJ AND ANAMITRA MAKUR
Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore 560 012.
email: {victor, amakur}@ece.iisc.ernet.in, Phones: (+91) 080 360 2167/309 2745, Fax: (+91) 080 360 0683/360 0563.

Abstract

The challenge posed by multimedia processing is how to provide services that seamlessly integrate text, audio, speech, image and video information, ensuring the ease of use and interactivity of conventional telephone service. This paper discusses an implementation of video telephony over wired and/or wireless network running Internet Protocol (IP) by taking into account seamless integration, compression of video and associated audio, and network error recovery.

Keywords: Video telephony, real-time compression, wireless, network error recovery.

1. Introduction

While the dictionary definition of multimedia is including or involving the use of several media of communication, entertainment or expression, a technological definition as it applies to communication might be the following: integration of two or more of the following media, viz. audio, speech, image, video, animation, text, data files, etc.

There are two key communication modes in which multimedia systems are generally used, viz. person-to-person and person-to-machine (not discussed, as this falls out of scope). As depicted in Fig. 1, in person-to-person mode,[1] there is a user interface which provides the mechanisms for all users to interact with each other and a transport layer that moves the multimedia signal from one user location to another. The user interface is responsible for creating the multimedia signal, and allowing users to interact with the multimedia signal in an easy-to-use manner. The transport layer has the job of preserving the quality of the multimedia signals so that all users receive what they perceive to be high-quality signals at each user location.

Multimedia processing is a lot more than signal compression and coding, viz. capture/record, display/playback, compress/decompress and transmit/receive audio, speech, video and image. Handling of multimedia signal is a very important part of the process in any multimedia application. By handling we mean how it is delivered to the end-user, displayed and played back. User interface is perhaps the most important component to the success or failure of a multimedia application.

The motivation for this work is as follows. Instead of hearing only the voice, video of the speakers is also displayed and this adds to human touch. In large organizations, almost every individual has a PC (with sound blaster card and webcam) and a connection to Internet, which can be used to support remote communication. Voice communication is possible wherever PC is already in use and more importantly we are not limited any more by 64 kbps bandwidth. As this application can run on wireless environments, a limited mobility within an office premise

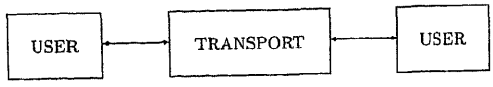

FIG 1 Person-to-person communication.

is available. Desktop video telephony is cost effective. The cost involved is only that of telephony software as opposed to the conventional telephony. There are economical advantages to end users in utilizing an integrated network, not only in terms of direct transmission costs but also in reducing network management costs of running separate and technologically different networks.

In this paper, we present the implementation, working and performance of video telephony which is similar to conventional telephony except that in addition to audio, video too is transmitted and received by both the sides. A PC calls another PC over the network and a full duplex real-time video and audio link between two systems is started. Any party can hang up. Both audio and video are compressed and decompressed in real time in each system. The network is primarily accessed by client programs in PCs and hence it is inherently PC oriented and client/server driven.

2. Specifications

2.1. *Audio specification*

Audio is full duplex, i.e. simultaneous playback and recording in the same system. The lowest available sampling frequency (in PC sound card) of 8 kHz is not supported for full duplex mode and hence the next higher sampling frequency of 11.025 kHz is chosen.

2.2. *Video specification*

The video sequence consists of grey-level images represented in 8 bits per pixel. Each frame is of size 360×288 pixels. The rate of video sequence is 25 frames per second.

2.3. *System specification*

This work was implemented on a 300 MHz Pentium PC running Windows NT workstation. Both the server and the client have a sound blaster card, a microphone to record and speakers to playback. Also, they have Ethernet/WaveLAN card to connect to the LAN. The programming environment is VC++ and the programming language used is C++.

3. Implementation details

3.1. *Audio buffers*

Two audio buffers, one for recording the audio and another for transmitting the recorded audio over the network, should be enough ideally. When the first buffer is full the second one is given to the audio input device and the data in the first buffer are transmitted over the network. By the time the second buffer is full, the first buffer should be ready to take the input. But the rate at which data can be transmitted over the network is unpredictable. If the first buffer has not finished transmitting over the network no buffer is available to take the input and hence

loss of information. This necessitates the presence of a third buffer at the transmitter side. The buffers are circulated in cyclic fashion.

At the receiver end, when the first buffer is fully played out, the second is to be given to the audio output device for playback. It may so happen that the second buffer is still reading from the network and not ready to be given to the audio output device, resulting in silence in the playback. To avoid this situation a third buffer is used. Playback of the first buffer begins when the second buffer is full.

Audible delay increases with the buffer length and hence each audio buffer is of half-a-second length (equivalent to 5512 samples of uncompressed data). If an audio packet is lost, only half a second is lost.

3.2. *Video buffers*

The video buffer size is dictated by video frame size and hence its size is equivalent to the size of the video frame. Unlike audio, continuity is not vital for video. Therefore, video frames are acquired, compressed and transmitted only when the system is ready for the next frame. Consequently, a single video buffer would suffice. However, video compression requires two more buffers to store the previous frame and the compressed frame. Thus, three video buffers are used at the transmitter. For similar reasons, three video buffers are used at the receiver as well.

3.2. *Video compression*

To achieve a higher compression ratio, a compression algorithm should exploit the redundancies in the best possible way. But this method requires quite a lot of computation. A study on various compression schemes and their applicability to real-time compression has been carried out before[2] and based on that conditional replenishment is chosen for the present work.

DCT assumes 8×8 pixels per block and each multiplication and addition takes two instructions. This gives rise to *frame size* $\times 4 \times 8$ total number of instructions per frame. In the case of motion estimation if one uses suboptimal motion estimation techniques such as the three-step search algorithm (the search position reduced to 25) with a search window of ± 6, the total number of instructions per frame is given by *frame size* $\times 50$. In conditional replenishment, one absolute difference and one comparison is needed for each pixel and hence *frame size* $\times 2$ is the total number of instructions per frame. Table I gives the MIPS calculations for the encoding complexity of three different compression schemes for the present video frame size.

A compression scheme which is suitable for real-time encoding will be suitable for real-time decoding as well since decoding is less complex than encoding.

Table I
Encoding complexities of various compression algorithms

Scheme	MIPS
DCT	82.944
Motion estimation	129.6
Conditional replenishment	5.184

3.4. *Protocols*

For delivering real-time data such as audio and video for playback, TCP and other reliable transport protocols are inappropriate.

Reliable transmission is inappropriate for delay-sensitive data. By the time the transmitter has detected the missing packet and retransmitted it, at least one round trip time, likely more, would have elapsed. The receiver either has to wait for the retransmission increasing the delay and incurring an audible gap in playback, or discard the retransmitted packet defeating the very purpose of TCP mechanism. A packet arriving late is more serious than an erroneous one. A single packet lost repeatedly could drastically increase the delay, which would persist at least until the end of the session.

Another important point is that TCP cannot support multicasting. TCP congestion control mechanism decreases the congestion window when packet loss is detected. On the other hand, audio and video have natural rates that cannot be suddenly decreased without starving the receiver. For the above said reasons, user datagram protocol (UDP) has been chosen for the transport.

Since the application has both audio and video sources, it is important to distinguish between the two media. It is also necessary to resequence the packets delivered as they may be delivered out of order. For these two reasons, payload type and sequence number features of real-time transport protocol (RTP) have been made use of.

3.5. *Packetization*

Experiments have been done with various packet sizes ranging from 4 to 32 kbytes over UDP. We observe that if the size of the packets is larger then the throughput is more. A packet size of 32 kbytes or less is chosen in this implementation. As we send compressed data, the frame size depends on the video information in case of video frame.

If a frame is larger in size for transmission, then it is segmented into smaller packets. For example, if a frame is of size 110 kbytes, then it is split into four packets of size 32, 32, 32, and 14 k (Fig. 2).

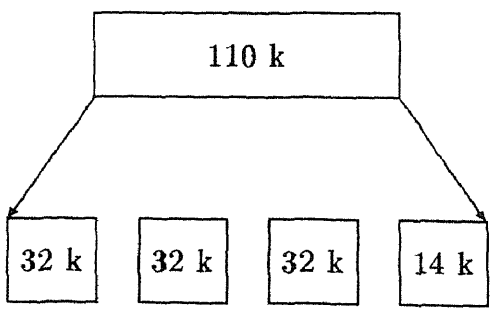

FIG. 2. Segmented frame.

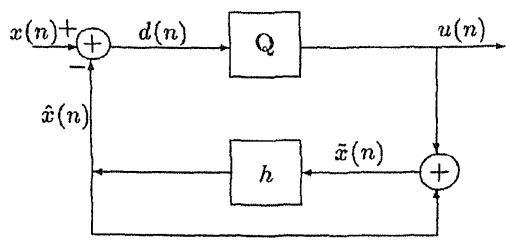

FIG. 3. Linear predictive coder.

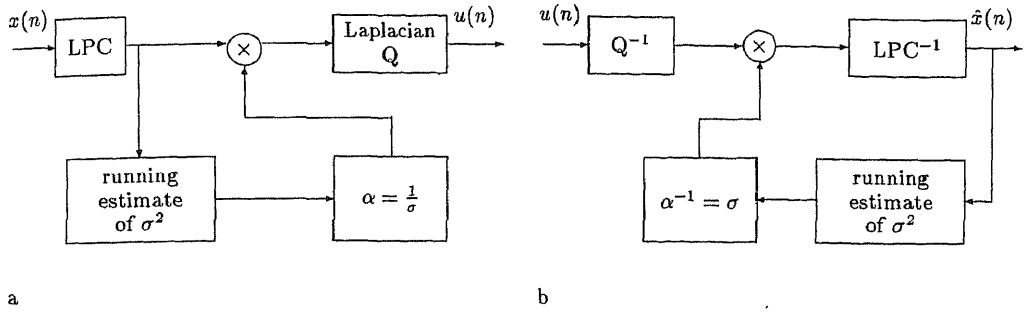

FIG 4a. Forward adaptive LPC with AQB, and b. Decoder for a.

If a frame happens to be smaller in size than the maximum packet size decided upon for transmission, i.e. 32 k, then it is transmitted as it is.

4. Compression

A key problem with packet networks is that they are not designed to handle voice traffic. With the emergence of multimedia PC and its concomitant signal-processing capability, a simple solution was proposed, viz., the compression (decompression) and coding (decoding) of voice into IP packet using the PC processor and the sound card inherent in the multimedia PC. Enormous data rate of video necessitates video compression.

4.1. *Audio compression*

A tenth-order forward adaptive linear predictive coding is employed. The entire audio frame is buffered to calculate predictor coefficients using Levinson–Durbin recursive algorithm.[3] After finding the predictor coefficients, output of the filter $(\hat{x}(n))$ is subtracted from the original signal $(x(n))$. This gives the error signal $(d(n))$.

The error signal is then quantized using a backward adaptive quantizer (AQB).[4, 5] The quantizer has an adaptation window of much smaller size compared to the audio frame size. Assuming Laplacian probability density function (pdf) for the difference signal, pdf-optimized 3-bit uniform quantizer is used. Running variance of the audio signal is used to compute the scale factor α which scales the quantizer input as shown in Fig. 4a. Backward estimates are derived from quantizer output. The predictor filter is reset at the beginning of each audio frame and they are encoded afresh, independent of the previous frame. The decoder for this scheme is shown in Fig. 4b.

Audio data at a sampling frequency of 11.025 kHz, mono, 8 bits per sample give rise to a data rate of 88.2 kbps. By encoding the error signal with 3 bits, instead of the conventional 8 bits, a saving of 5 bits is achieved. This gives a data rate of 33.075 kbps, a compression of 62.5%.

4.2. *Video compression*

The compression scheme adapted is interframe coding using conditional replenishment. This technique exploits the temporal redundancies in the video sequence, which are inherent in mo-

tion video. The previous frame serves as prediction for the present frame. In the present frame, only pixels which differ from the previous frame by an amount greater than a threshold are transmitted with their addresses.

In conditional replenishment,[2] it has been observed that the pixels, whose absolute difference is less than a threshold, occur in runs. Hence, if we use runlength coding to represent the number of such pixels, we do not need to transmit the address information for the pixels which have to be replenished. If $P(x, y, n)$ and $P(x, y, n-1)$ represent the pixel values at position (x, y) in the frames n and $n-1$ respectively, then the absolute value of the difference is given by

$$\text{ADIFF} = |P(x, y, n) - P(x, y, n-1)|$$

The runlength counter is initialized with zero. When ADIFF is less than the threshold, runlength counter is incremented until it touches 127. Then it is transmitted and the counter is reset. When ADIFF exceeds the threshold, the counter is tested. If it is zero, then the pixel value is transmitted. If it is one, the previous pixel value is transmitted first and then the present pixel value. The counter is reset after transmission. If it is greater than one, the runlength value is transmitted first and then the pixel value. The counter is then reset.

All those pixel values which differ from the previous frame by an amount greater than the threshold (value of 10 was chosen) are quantized by a 7-bit (128 level) uniform quantizer. The transmitted values are represented by 8 bits. In the case of wireless access if there is any change in the available bandwidth, the threshold value can be changed appropriately so that the compressed stream always remains within the bandwidth of the link.

5. Video telephony protocol

Packet networks are general-purpose data networks that are not tied to fixed bandwidth circuits. Instead, they are designed to transmit bits, in the form of packets of fixed or variable length, only when there are bits to transmit. Packet networks evolved independent of telephone networks for the purpose of moving bursty, non real-time data among computers. This is distinguished by the property that packet communications are routed by address information contained in the data stream itself. The packet network is especially well suited for sending data of various types, including messages, facsimile, and still images. The network is not well suited for sending real-time communication signals such as speech, audio and video.[6]

The video telephony protocol has four phases, viz., initialization and waiting for a call, call set-up, data transfer, and call termination.[7] Figure 5 depicts the scenario.

When the application is activated, it waits in an indefinite loop for a call from another party. It also keeps checking if the user wishes to start a conversation. If the user does so, the caller sends a special sequence to the callee, signifying connection request. When the callee receives the connection request, it confirms that it is accepting the request by echoing the connection request packet to the caller. The caller is identified by the IP address and port number of the connection request packet. The callee starts the next phase. When the caller receives a confirmation of the request, it starts the next phase.

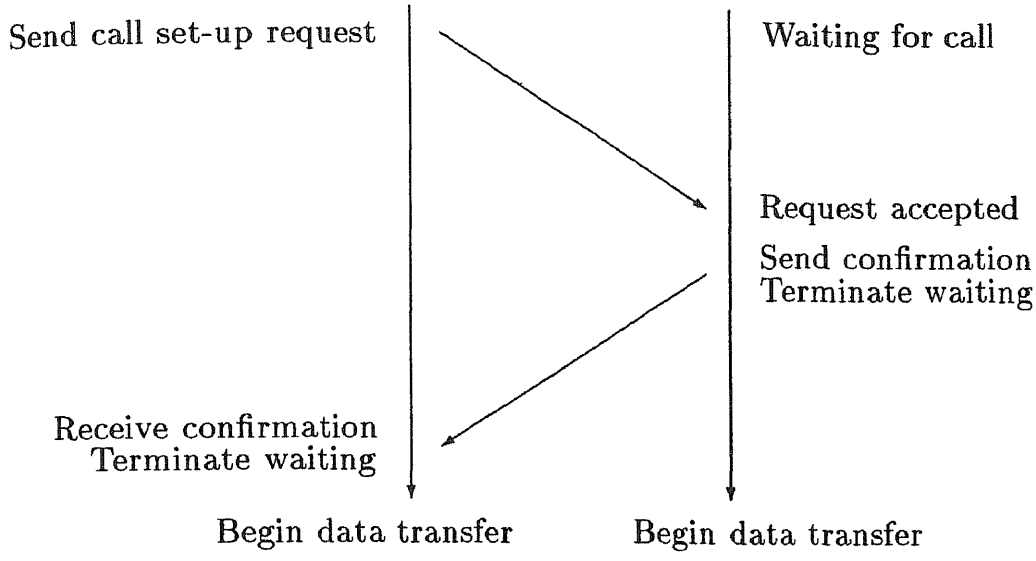

FIG. 5 Video telephony protocol

Before sending each frame, the system checks for termination request by the user. If there is a request for termination, the system stops the transfer and sends a stop transfer request to the other party. This request is another special sequence of bytes. Once the other party receives this request, it stops sending and receiving. Thus a call is terminated. To maintain a fixed frame rate of transmission, system timers are made use of. A frame is transmitted after the timer generates a time out.

6. Display

Windows device-independent bitmap (DIB) is used for the display of video. A DIB carries its own colour information, and hence colour palette management is easier. Windows NT is not optimized for direct display of DIBs. Windows NT operating system has a client server boundary built in. The frame buffer is on the server side and the DIB is on the client side. Hence all communications are through shared memory. The new WIN32 DIB section bridges the gap and this work uses Bitblt and StrtchBlt to directly transfer DIB bits to the display.

To display a single DIB in a window a *logical palette*, a GDI object that contains the colours in the DIB is created first. Then this logical palette is realized into the system hardware palette. This makes use of WIN32 Selectpalette and RealizePalette functions.

7. Functioning

The IP address of the machine (similar to the telephone number of the callee, in case of telephone connection) to which the call is destined is specified in the address field of User Interface (UI). *start* button is pressed to execute the call setup. Call setup executes two threads.

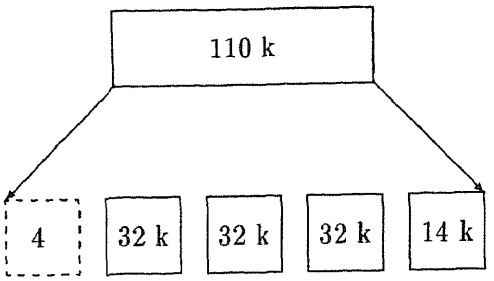

FIG. 6 Segmentation of frame with *forced update*.

One is the sender thread and the other the receiver thread. These two threads carry out tasks independently. The telephony which is duplex calls for multithreads in this application.

The sender thread initiates audio recording by submitting an audio buffer to the audio input device. When the first buffer is submitted, the other two buffers are queued up at audio input device. Then it reads a video frame and compresses it. After compression, it packetizes video (if need be) and sends it over the network. After all the video packets of the present frame are sent, the sender thread checks to see if the audio is ready. If so, it is compressed and transmitted. In case it is not ready, the thread goes to read the next video frame. When both audio and video frames are ready to be transmitted, audio is given preference over video, i.e. *audio-prioritized* multiplexing. The above procedure is repeated all over again until the call is terminated either by the caller or by the callee.

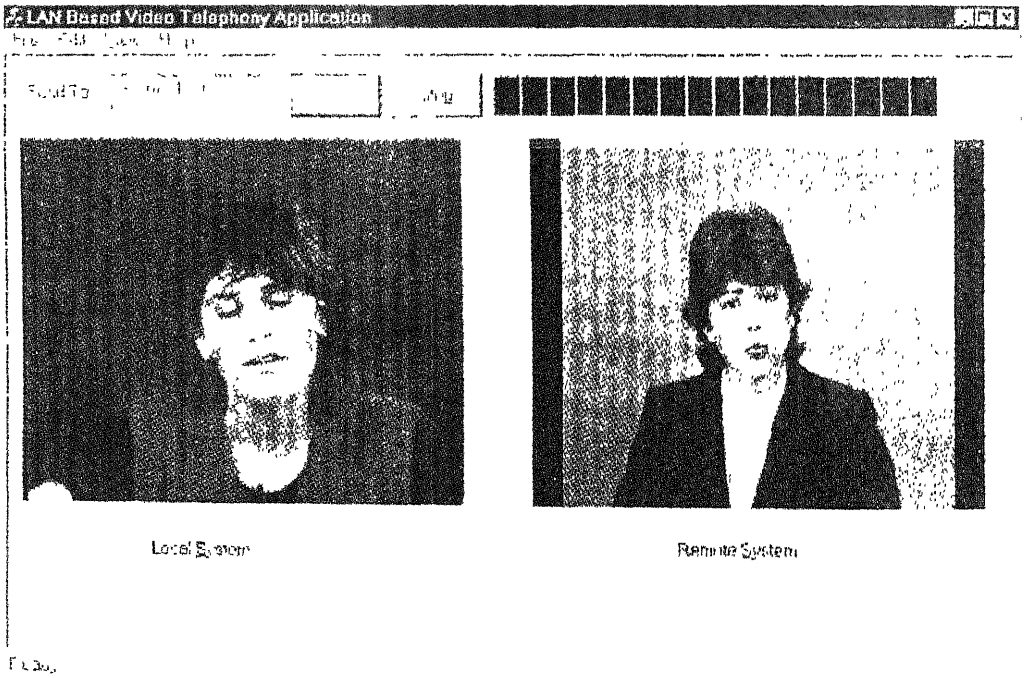

FIG. 7. Video telephony in action.

The receiver thread is activated only if there is a frame to be received. It reads a frame from the network and the content of the frame is determined. If the frame is audio, it is uncompressed and submitted to audio output device for playback. If it is video, it is decompressed first and then the packets are resequenced before the frame is displayed.

8. Network error recovery

One inherent problem with any communication system is channel noise. It is more likely that the channel noise is prominent in the case of wireless link. The information may be altered or lost during transmission due to channel noise.[8] The effect of such information loss can be devastating for the transport of compressed video and audio because any damage to the compressed bit stream may lead to objectionable visual and listening distortion at the decoder.

To circumvent this problem, *forced update*[9] approach is adapted. In *forced update* each frame is divided into five slices. In every frame one slice of video data is transmitted without compression. The compression is implemented in the remaining slices only. A round-robin algorithm decides which slice goes uncompressed. In this approach, the entire video is guaranteed to be reconstructed within five frames, and thus helps in recovering from network errors.

When the *forced update* is in place, a 4-byte *pilot* indicating the size of the next frame is sent before each audio or video frame is sent. Thus the packets that are sent by the server are either a video frame, audio frame or a 4-byte *pilot*. For instance, if the frame size is 110 kbytes, then this is split into four packets of size 32, 32, 32 and 14 k. These four packets will be preceded by a 4-byte *pilot* (Fig. 6). The *pilot* packet is shown in dotted lines.

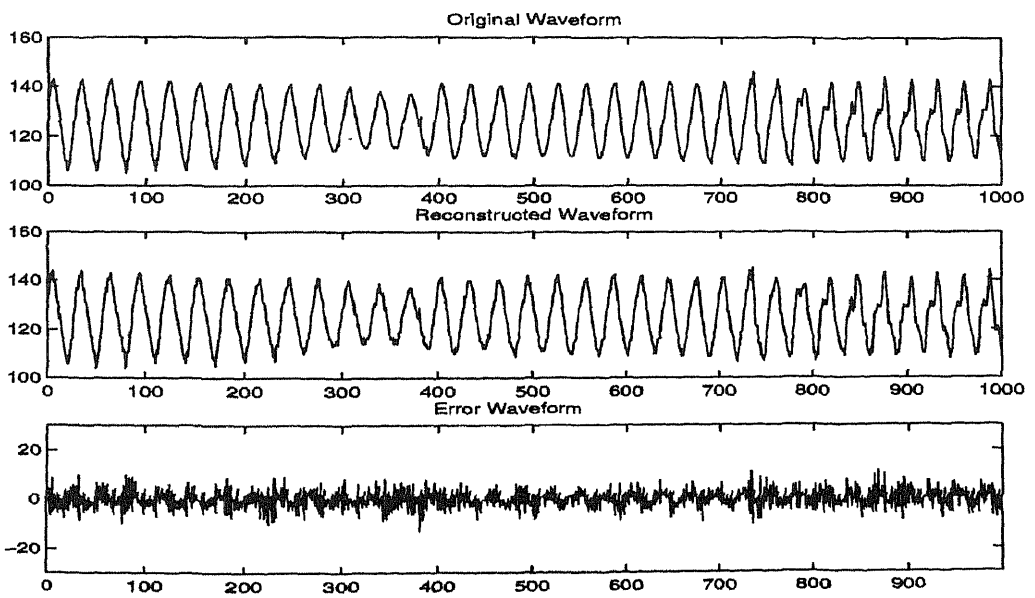

FIG 8. Audio waveforms.

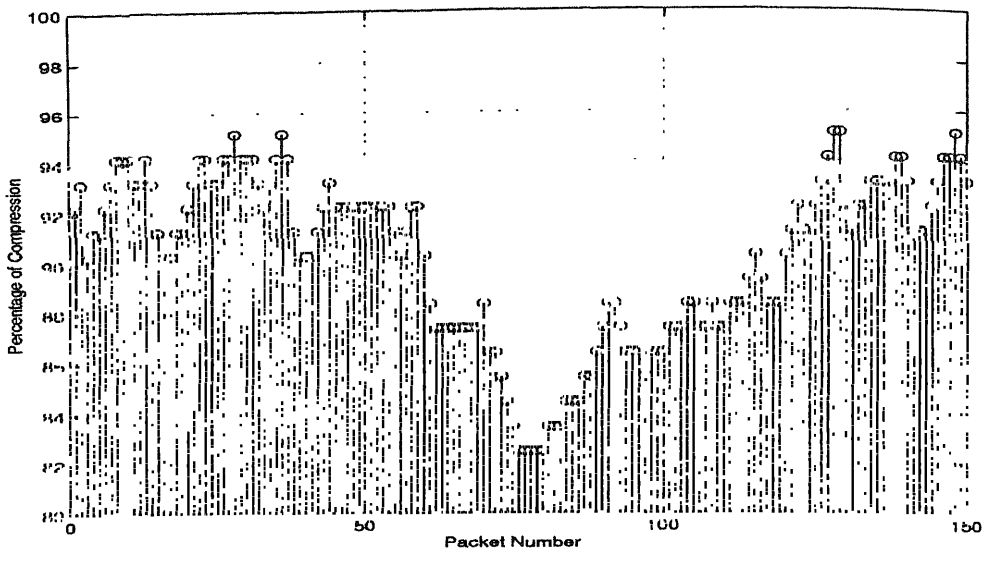

FIG. 9 Percentage video compression vs frame number

To lock onto the stream at any arbitrary moment, as we lost the previous frame owing to network error, the client looks for this *pilot*. This process of lock on returns only after receiving a *pilot* successfully. Thus it makes sure that the client receives the data and decodes it correctly, even in the case of error-prone wireless channel.

The audio coding is modified to take care of the erroneous reception. The predictor filter is reset at the beginning of each audio frame and they are encoded afresh, ensuring that each audio frame is coded independent of the previous frame. No correlation is assumed between audio frames. This guarantees the decoding of each audio frame even if the previous one is lost.

9. User interface

As mentioned earlier, user interface is perhaps the most important component to the success or failure of a multimedia application.

The graphic user interface (GUI) has *send to* edit box where the caller specifies the callee's IP address. The *start* and *stop* buttons are used to start and terminate the call. The progress bar tells the user about the number of frames sent so far. The bar wraps up itself over every 150 frames. Both local and remote-system videos are displayed. The GUI shown in Fig. 7 is a screen shot of the application.

10. Performance

The average time taken for compression and decompression of a 5.5125 k audio frame is 11 and 6 ms respectively. The signal-to-noise ratio (SNR) obtained was 18.71 dB. The original, reconstructed and the error waveforms for a segment of audio are depicted in Fig. 8.

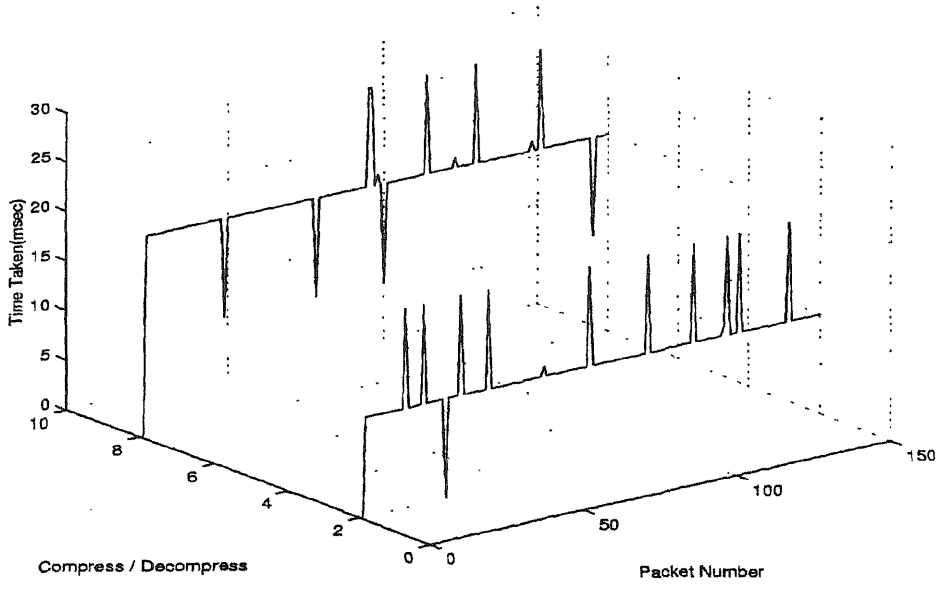

FIG. 10 Video frame number vs compression/decompression time.

Figure 9 limns the percentage video compression for the test sequence of *Miss America*. The average compression achieved for this sequence is 90.246 %. The compression and de-

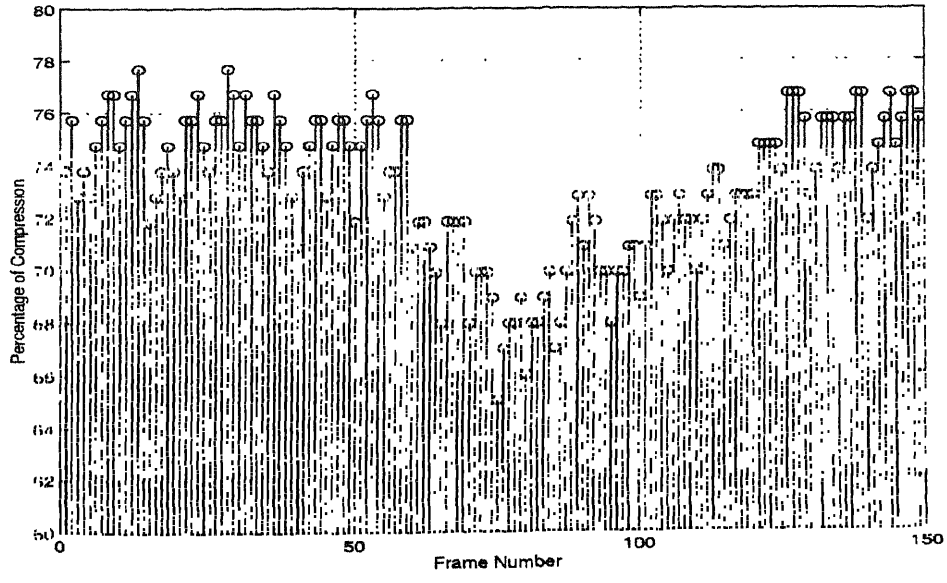

FIG. 11. Percentage video compression vs frame number with *forced update*.

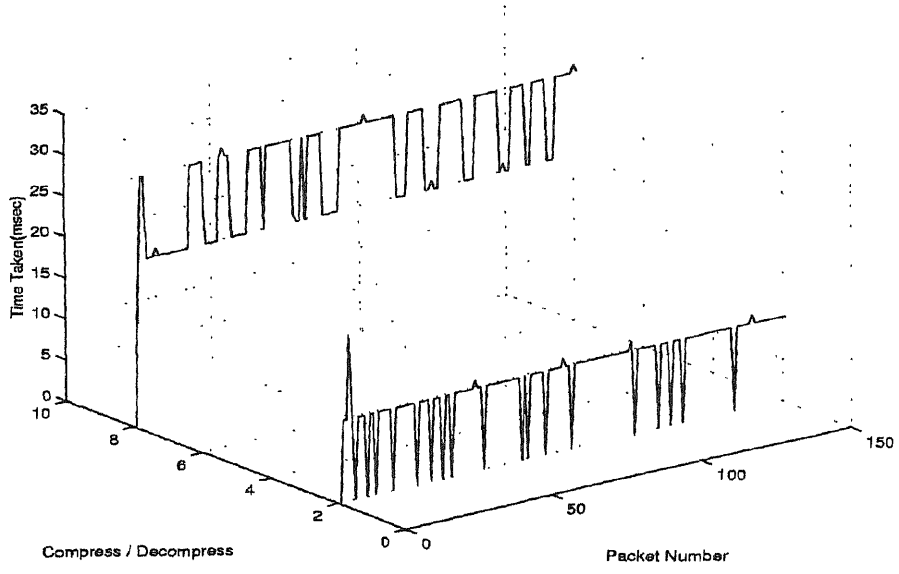

FIG. 12 Video frame number vs. compression/decompression time with forced update.

compression times are shown in Fig. 10. The average time to compress a video frame is 19.9533 ms, and to decompress 10.5467 ms.

The average compression achieved for the compression scheme with *forced update* is 73.5 % (Fig. 11). It can be observed that the average compression achieved in this scheme is lower than that of the technique without *forced update*, owing to slicing. The average time to compress is 21.7067 ms and to decompress 8.6267 ms (Fig. 12). The average time taken to display a frame is 11.9 ms. The video frame rate achieved is typically 10 to 20 frames/s.

11. Conclusion

This paper discusses the implementation of video telephony over both wired and wireless LAN. The service provided has seamless integration of video and associated audio in compressed form. Compressed video data transfer has brought down the data rate significantly. The quality of audio and video delivered to the end-user has not deteriorated by transmission over wireless LAN. Real-time multimedia communications over a packet switching network is subject to packet loss. If a frame is segmented for transmission, the loss of a single packet can render the entire frame useless. This implementation has error-recovery schemes, which take care of erroneous receptions, as part of the compression techniques. The user interface is user-friendly.

The contribution in this work includes audio compression which compresses each audio packet independent of the previous one, video compression with forced update, buffer management, video telephony protocol, packetization, wired and wireless implementations, and graphical user interface (GUI).

Acknowledgements

The authors acknowledge Messers K. V. S. Jagannadha Rao, Shailendra Sinha and Avinash Agarwal for their help and involvement in developing parts of the necessary software. They also acknowledge partial financial and material support for this work by the Ministry of Human Resource Development, Government of India, Samsung, and Intel.

References

1 COX, R. V., HASKELL, B. G., LECUN, Y., SHAHRARAY, B. AND RABINER, L. Scanning the technology on the applications of multimedia processing to communications, *Proc. IEEE*, 1998, **86**, 755–824

2 YOGANAND, R *Implementation of PC based online audio/video service through ethernet*, M. E. Thesis, Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India, 1997

3 MAKHOUL, J Linear prediction: A tutorial review, *Proc. IEEE*, 1975, **63**, 561–580.

4. SPANIAS, A. S. Speech coding: A tutorial review, *Proc. IEEE*, 1994, **82**, 1541–1582.

5 JAYANT, N. S. AND NOLL, P. *Digital coding of waveforms. Principles and applications to speech and video*, Prentice Hall, 1984.

6 ARAS, C. M., KUROSE, J. F., REEVES, D. S. AND SCHULZRINNE, H Real-time communications in packet-switched networks, *Proc IEEE*, 1994, **82**, 122–139

7 SINHA, S *Implementation of multimedia video services through network*, M. E Thesis, Department of Electrical Communication Engineering, Indian Institute of Science, Bangalore, India, 1998

8 WANG, Y. AND ZHU, Q. F. Error control and concealment for video communication: A review, *Proc. IEEE*, 1998, **86**, 974–997.

9. ANANDA RAI, P. V., JAGANNADHA RAO, K. V. S. AND MAKUR, A A LAN-based software-only remote classroom application, *Proc NCC-99*, 1999, pp. 432–437.