Modular Structure for Software Implementation of H.323 Multimedia Conferencing Terminal

King-Man HO and Kwok-Tung LO
Center for Multimedia Signal Processing
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
HONG KONG

Abstract: - The ITU-T H.323 standard defines the components, protocols and procedures for making multimedia conferencing over packet-switched networks such as the Internet. Although the standard gives many guidelines to the developers, it does not have enough information to implement the whole system. Some of the important parts including capture and playback of audiovisual signal, user control, low-level synchronization and networking are outside the scope of the H.323 standard and need to be defined by the developers. In this paper, a modular design is developed for software implementation of an H.323 terminal in a desktop environment. The design is flexible enough to include further enhancement of the system.

Key-Words: - Video conferencing, multimedia communications, H.323

1 Introduction
Video conferencing [1] has been developing from an emerging technology into a useful and widespread tool for improved communications. A typical video conferencing system is equipped with an audiovisual capture and compression subsystem as well as a display/playback and decompression subsystem. As the compression process involves a large amount of computational intensive operations, most of the video conferencing systems are implemented by specific hardware equipment in order to maintain a certain level of quality of service (QoS). Obviously, this approach is not cost effective and lacks of flexibility. As the power of the personal computer (PC) increases, it is desirable to perform video conferencing in a desktop environment by using software only.

Since the early 90’s, many standards have been developed for video conferencing. The first standard, and the standard that most other standards are an extension of, is called H.320 [2]. This standard was originally developed for video conferencing in a narrow-band ISDN. H.320 supports two different formats of video resolution, and also three different audio coding algorithms, each providing different levels of quality and bandwidth consumption. As video conferencing technology grows, other standards have been created that are really extensions of H.320 but applying to different communication media. Many of these extensions include the same algorithms used in H.320 except they are modified to support the type of network being used. Among them, H.323 [3] is a standard developed for video conferencing over a LAN. As the rapid deployment of the Internet, H.323 has been used for various desktop multimedia communications including the emerging IP-phone applications.

Although H.323 gives many guidelines to the developers, it does not have enough information to implement the whole system. Some of the important parts for implementation are outside the scope of the standard and open for developers. For this purpose, we develop in this paper a modular structure for software implementation of an H.323 terminal in a desktop environment. According to the standard, H.323 provides different service levels for multimedia communication over a data network, that is, voice only; voice and video; voice and data; or voice, video and data communication. In this work, we will focus on voice and video communications.

In the rest of this paper, the proposed modular structure for software implementation of an H.323 terminal will be described in details in section 2. System implementation using the proposed modular design will be discussed in section 3 and some concluding remarks are given in section 4.

2 Modular Structure for Software Implementation of H.323 Terminal
The H.323 standard only specifies the call signaling procedures, video codec and audio codec as well as the RTP packetization. Some of the important parts including audio/video capture/playback, user control,
low-level synchronization and networking are outside the scope of the standard and need to be defined by the developers.

To make the system more flexible and easy to follow, a modular approach is proposed in this work for software implementation of an H.323 terminal. In our design, each module has its own unique purpose and will communicate with other modules by sending messages. A number of modules combine together to form a unit. Fig.1 shows our modular design for implementation of an H.323 terminal. The proposed implementation includes five units, where each unit contains a number of modules that providing a unique operation. The details of each unit are described in the following.

2.1 Application Control Unit (ACU)
The Application Control unit (ACU) is responsible for controlling and coordinating the overall performance of the entire system. This unit acts as a middleman between the user and other units within the system and provides a user interface. The function of this unit is to send a message to other units based on the user requirements. Fig.2 shows the internal structure of the ACU.

2.1.1 Graphic User Interface/Application Control
This unit provides a graphical interface for the user to interact with the entire system, in which the user can select different parameters such as video format, frame rate, audio and video codecs to achieve certain level of system performance.

2.2 Call Signaling Unit (CSU)
The Call Signaling Unit (CSU) is used to establish a logical channel among endpoints so that the media data can be transmitted over the network through this channel during the conferencing stage. This unit follows the call signaling procedure specified in H.225.0 [4] and H.245 [5] in order to achieve communications among different endpoints. The internal structure of the CSU is shown in Fig.3, in which there is only one module called Call Signaling Module.

2.2.1 Call Signaling Module
This module is based on the ITU-T recommendations H.225 and H.245 to provide the call signaling among the endpoints.

2.3 Media Data Transmission Processing Unit
The Media Data Transmission Processing Unit (MDTPU) is an important part of the system. This unit mainly performs three tasks, a) video and audio capture, b) video and audio compression; and c) packetization. According to the H.323 standard, the video and audio capturing methods are outside the scope of this standard. Although the codec and the packetization method are defined in the standard, the interoperability among these components is also beyond the scope of the H.323 standard. As a result, it is necessary to use other methods to solve this problem. This section will introduce a solution for the interoperability of different components in this unit as well as the implementation method. Fig.4 shows the internal structure of the MDTPU.
2.3.1 Video Capture Module
This module provides the low-level mechanism to obtain the video data from the video capture equipment. The captured video data is stored in the video buffer preparing for encoding. Other optional functions such as video format selection, frame rate selection, preview and hue control are also provided.

2.3.2 Audio Capture Module
This module provides the low-level mechanism to obtain the audio data from the sound input device of PC. The captured audio data is stored in the audio buffer preparing for encoding.

2.3.3 Video Encode Module
This module performs video compression to reduce the size of each captured video frame before sending to other endpoints.

2.3.4 Audio Encode Module
This module provides audio encoding to reduce the size of the captured audio data.

2.3.5 Audio/Video Coding Process Control Unit
Encoding of audio and video involves complex computations and is very time consuming. The encoding process may not be completed within a pre-defined time. To avoid the buffer overflow or memory conflict, this unit is used to control the captured audio and video data storing in the buffer.

2.3.6 RTP Packetize/ Multiplex Unit
This module provides the mechanism to multiplex both audio and video streams into a single stream for transferring to other endpoints through a single channel. This unit performs two tasks. Firstly, it appends the real-time transport protocol (RTP) [8] with the compressed media data produced in the Audio and Video Encode Module to provide end-to-end real-time delivery services including payload type identification, sequencing numbering, time-stamping as well as delivery monitoring. Making use of such extra information, the receiver can not only reconstruct the sender’s packet sequence but also provide timing and synchronization for setting the timing at the receiver during the content playback. Furthermore, it can allow the receiver to identify the sender for each received piece of data during multi-point conferencing. The second task of this unit is multiplexing. Actually, each medium is carried in a separated RTP session with its own destination transport address. This unit will make use of the service provided by UDP to multiplex the audio and video stream into a single stream. So during the implementation stage, it is only necessary to create two Window Sockets, which are assigned different port numbers (port number is assigned dynamically during the H.245 procedure) as illustrated in Fig.5.

2.4 Media Data Reception Processing Unit
The Media Data Reception Processing Unit (MDRPU) is another important part of the system. This unit mainly performs four tasks: a) de-packetization, b) audio and video de-compression, c) synchronization and audio mixing, and d) audio playback and video display. Similar to MDTPU, the H.323 standard only defines the codec used in the receiving path. The operation of the media data after decompression is outside the scope of the standard. Fig.6 depicts the internal structure of the MDRPU.

After the media data is decoded in the Audio (or Video) Decode Module, the decoded data will not be played back or displayed immediately. Instead, this data will firstly stay in a buffer within a defined time. The needs of this process are due to the following reasons:

- The packet may be delivered out of order.
- The packet may not arrive at a pre-defined time.

All computers attached to a LAN environment need to participate in a shared environment to share the resources of the network. The computer needs to check for a carrier before it can transmit...
the data over the network. This will result in no delay guarantee and the problem will become worse when network congestion occurs. On the other hand, the time used for performing media data compression cannot be predefined which depends on the complexity of the media data source. As a result, the packet may arrive in a burst or spurt and direct playback or display may cause variation in quality.

- The need for synchronization: The audio and video data with the same timestamp may not arrive at the same time.
- The need for audio mixing.

2.4.1 Buffering Scheme

Fig.7 shows the block diagram of the buffering scheme proposed in our system. In this figure, we use the Audio Decode Unit to demonstrate the operation of this scheme (the operation of Video Decode Unit is similar to that of Audio Decode Unit).

As shown in Fig.7, one buffer stack is allocated to each participant in the terminal at the beginning of the conferencing stage. This buffer stack is composed of two buffer streams. Each buffer stream contains a number of cells and stores the decoded audio data as well as some important RTP information. Each buffer stream will connect to different components within a different time controlled by the Timing Control Unit. Say for example, buffer stream 1 connects to Audio Decoder and buffer stream 2 connects to Synchronization Control Unit in t1. And they will alternate their connections in t2. This results in two buffer streams, which are interlaced by using the Synchronization Control Unit as well as the Audio Decoder during the conferencing stage. So, the buffer confliction does not occur under this scheme. One buffer stream is used for data storing while the other is used for data playing back.

The number of packets in the buffer stream defines the maximum data that can be held in the buffer stream within a defined-time before it can be played back. The frame size of both G.723.1 and G.711 codecs is 30ms. If each buffer stream contains 10 packets, the maximum data that can be stored is 10 within 300ms. The Timer Control Unit will set a timer equal to 300ms in which the timer switches between two buffer streams, alternatively to the Audio Decoder and Synchronization Control Unit when it expires. Under this scheme, the data loss detection can be achieved. Fig.8(a) and 8(b) show the structure and operation of the buffer stack respectively.

![Buffering Scheme Diagram](image)

Firstly, the Timing Control Unit resets the timer that directs buffer stream 1 and buffer stream 2 connecting it to the Audio Decoder and Synchronization Control Unit respectively. The decoded audio data stores into the buffer stream 1.
packet by packet until the timer expires. Once the timer expires, the system examines the content in buffer stream 1 and records the packet that contains the largest sequence number. At this moment, the Timing Control Unit switches buffer stream 1 to the Synchronization Control Unit for playing back and switches buffer stream 2 to Audio Decoder for storing decoded data. Now, the decoded data stores into buffer stream 2 and the system checks all sequence number in the decoded data. If it is found that the sequence number is smaller than the one just recorded in buffer stream 1, the system will discard it. This approach can prevent the system from waiting all necessary packets too long that will result in the loss of real-time property.

2.4.2 Ordering Scheme
Fig.9 depicts the condition that the data arrives out of order. In order to make sure the sequence of data is in correct order before playing back, the procedure of sorting is necessary. The sorting procedure is achieved by the Sequence Process in the Synchronization Control Unit. This process performs sorting according to the sequence number in each packet. In this work, we utilize one of the common sorting techniques known as Insert Sort. The advantages of this sorting technique are that the process time is low if the data size is not large and it will do nothing if the data sequence is in order. So, this sorting technique does not increase the loading of the CPU.

2.4.3 Synchronization Scheme
After the completion of the Sequencing Process, data in the buffer stream can be output to the desired device for playing back or displaying. Before that, the last necessary procedure is the Synchronization Process. This process makes sure that video and sound are synchronized. Fig.10 shows the components necessary to achieve the aims of synchronization. The diagram shows two buffer streams, one for video and one for audio. The value shown in each packet is the timestamp. As mentioned before, audio and video data may arrive at a different time. This may result in the timestamp between two buffer streams not being exactly the same. The timer in the Timing Control Unit controls the connection between the audio buffer stream and the Synchronization Control Unit while the connection between the video buffer stream and the Synchronization Control Unit is controlled by the audio buffer stream. When the timer in the Timing Control Unit expires, it directs one packet in the audio buffer stream to the Synchronization Control Unit. At this moment, the audio buffer stream examines the timestamp value in each packet of the video buffer stream. If it is found that one timestamp value matches itself, it will direct this packet to the Synchronization Control Unit. Then the Synchronization Control Unit passes both audio and video packets to the Audio Playback Module and Video Display Module respectively. If no timestamp value is matched, it will not display the video content at this moment. It is noted that the audio stream is defined as the master stream and the video is defined as the slave. So, all audio data must be played back. The video data can be output depending on whether its timestamp value is matched to the audio one. As a result, the packet with the timestamp value from 1 to 3 in the video buffer stream will not be displayed, as shown in Fig.10. This process will continue until all data in the current buffer stream is processed.

2.5 Network Processing Unit
The Network Processing Unit (NPU) is used to provide a channel for the system to interact with the network interface, to transmit/receive data as well as to do call signaling. The implementation of this unit is outside the scope of H.323 recommendations. This unit is mainly implemented by using Windows Socket API, which allows the application to run on any vendor’s Winsock-compatible TCP/IP protocol stacks such as Microsoft Windows. Fig.11 shows the internal structure of the NPU. It is noted from the figure that two main processes, UDP Process and TCP Process, are used to support two different kinds of transport protocols to adapt to the requirement of data transmission in the Media Data Transmission (and Reception) Processing Unit and the Call Signaling Unit respectively. The UDP Process provides an inherently connectionless service that is capable of supporting broadcast and multicast services. Since video and audio are both delay-sensitive, they cannot tolerant data retransmission
and congestion-control mechanism. As a result, the UDP Process is more suitable for transmitting media data. In contrast, the TCP Process provides a connection-oriented service that is capable of guaranteeing that the data sent across a connection will be delivered with no data missing or out of order. So, the TCP Process is more suitable for transmitting the data requiring a high-level reliability transmission, such as the call signaling over the network.

<table>
<thead>
<tr>
<th>Network Processing Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP Process</td>
</tr>
<tr>
<td>UDP Process</td>
</tr>
<tr>
<td>Terminate Process</td>
</tr>
</tbody>
</table>

Fig. 11 Internal structure of the NPU

3 System Implementation

According to the proposed modular structure, we have implemented an H.323 terminal in a desktop environment using software only. In our system, the hardware setup is very simple and only requires a standard video capture device, like a USB camera as well as a sound card, to be installed at the user side. The scenario of the video conferencing system is described as follows. The audiovisual signal of the user is first captured by the capturing devices of the PC, i.e. the USB camera and the sound card. The digitized audio and video signal are then compressed by a software H.263 [6] video encoder and G.723.1 [7] voice encoder. The compressed audiovisual signal is then packetized and transmitted in multicast format so that one stream of data is sent over the network and shared by multiple users. At the receiver side, the audiovisual data is decompressed and displayed.

For the current achievement, the system can support up to 6 people making a conference within the same local area network concurrently. The frame rate achieved for video is about 15-20 frames/s for CIF sequences. All figures are measured by using Pentium IV computers with 2.0 GHz. It is noted that the number of concurrent users and the achieved frame rate can be further improved if the processing power of the PC is increased since all the subsystems of our system are implemented by software. In our system, it is also possible for the users to create a number of conference groups within a single LAN segment.

As the whole system is implemented by software, the system performance will greatly depend on the computing power of the PC as well as the optimality of the software codes. Therefore, more of the time was spent in optimizing the coding to reduce the loading of CPU so that the system could provide a better performance. Also, as multicast transmission is at present not supported by most of the Internet Service Providers (ISPs), a Multipoint Control Unit (MCU) as well as a gateway are required to perform conferencing across the Internet.

4 Conclusion

In this work, we have proposed a modular design for software implementation of an H.323 multimedia conferencing terminal. According to the modular design, a video conferencing system is developed. The system can support up to 6 people making a conference within a local area network concurrently. It is also possible for the users to create a number of conference groups within the single LAN segment.

Acknowledgements

This work was supported by Center for Multimedia Signal Processing, Department of Electronic and Information Engineering, The Hong Kong Polytechnic University, Hong Kong.

References:


