Efficient Multiplexing Protocol for Low Bit Rate Multi-point Video Conferencing*

Haohuan Fu, Xiaowen Li, Ji Shen, and Weijia Jia

Department of Computer Science, City University of Hong Kong,
83 Tat Chee Ave., Kowloon, Hong Kong
wjia@cs.cityu.edu.hk

Abstract. This paper discusses an efficient implementation of the multiplexing protocol H.223, which is an important part of 3G-324M protocol stack required for 3G mobile multimedia communications. Our implementation of the protocol aims to support the multi-point video conferencing with the capability of transmitting/receiving multiple video/audio streams simultaneously. Conference managements such as admission and audio channel scheduling are also discussed. As a result, the implementation improves efficiency and makes the conference more convenient to set up and operate. Our prototype system is stable and its performance is satisfactory.

1 Introduction

Currently, multimedia streams of 3G video calls is transferred using 3G-324M protocol [1] derived from ITU-T H.324 [2] standard by International Telecommunications Union (ITU) to enable multimedia communication over low-bit rate terminals (in the following, "ITU-T" is dropped for simplicity). H.324 is an umbrella protocol, referencing other important standards such as H.223 [4] that specifying data multiplexing/demultiplexing, and H.245 [3] that specifying the control messages and procedures. H.324 and its several mobile specific annexes are usually referred to as H.324M (M stands for *mobile*). The 3rd Generation Partnership Project (3GPP) [12] has adopted the H.324M with some modifications in codec and error handling requirements to create 3G-324M standard for 3G wireless networks [14].

This paper describes our prototype implementation called *anyConference*, which is a prototype multi-point video conferencing system over the mobile terminals, based on our 3G-324M protocol stack implementation. We mainly focus on the enhanced implementations of the multiplexing protocol H.223, which is designed to be capable to coordinate the sending and receiving of multiple media streams from different participating entities so as to support a multi-point conference rather than a simple video call.

The rest of the paper is organized as follows. A brief introduction of the 3G concerned standards and concepts used are given in section 2. In section 3, we describe our prototype video conference system first, and then we illustrate our

* This work is supported by CityU strategic grant nos: 70001587, 70001709 and 70001777.

X. Jia, J. Wu, and Y. He (Eds.): MSN 2005, LNCS 3794, pp. 478 – 487, 2005.

efficient implementation of the H.223 protocol with support for multi-point conversation; conference management such as admission control and audio channel scheduling will also be discussed. Section 4 gives evaluation of our implementation in the prototype system. We conclude the paper and give the future research direction in Section 5.

2 Background

2.1 Overview of 3G-324M/H.324M Standard

The architecture of 3G-324 protocol stacks is shown in Fig. 1 and the components protocols are illustrated as follows:

(1) H.324 — the Base Protocol, which consists of main protocol components.

(2) H.245 — specifies the Call Control Protocol which provides the end-to-end signaling for proper operations of the H.324 terminal. These signaling operations include audio and video capabilities exchange, opening and closing of logical channels, exchange of the multiplex tables of each party, and etc.

(3) H.223 —provides a multiplexing/demultiplexing service for the upper-layer applications. More details will be described in Section 2.1.

(4) Standards for video/audio codecs, such as H.263 [8], H.261 [9], G.723.1 [10], AMR (Adaptive Multi Rate) [13], and etc.

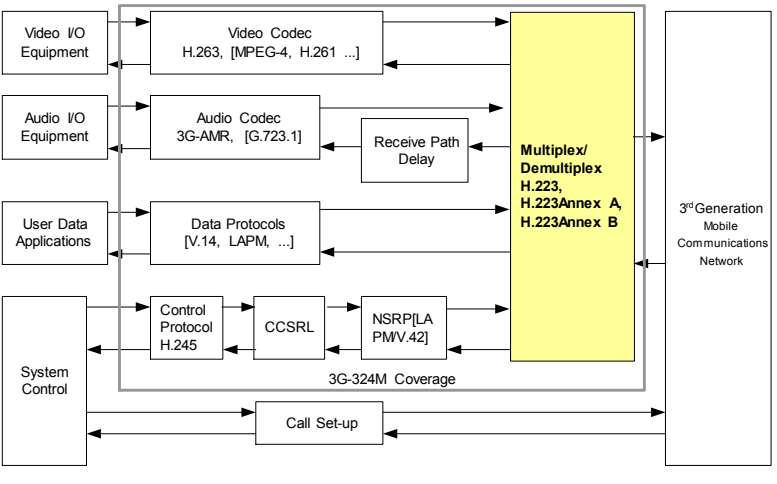

Fig. 1. Architecture of the 3G-324M protocol stack

2.2 Multiplex Protocol H.223

H.223 is usually used between two multimedia terminals, or between a terminal and a gateway adapter. As the interface in the middle of the above application layer (video/audio codecs and system control) and the below physical layer (WCDMA or

other 3G physical link), H.223 provides low delay/overhead by using segmentation, reassembly, and multiplexing information from different logical channels into one single packet. The entire function of this recommendation is divided into two layers:

(1) The first layer is Adaptation Layer (AL), which is mainly responsible for error detection/correction and optional retransmission for lost or corrupt packets. It is actually an interface for upper-layer applications, and it still deals with different source separately. This layer could be further divided into three more specific sub-layers (1) AL1 for control information and data; (2) AL2 for audio stream; (3) AL3 for video stream.

(2) The second layer is Multiplex Layer (MUX), which performs the actual multiplexing. In this layer, data traffics from different sources are regarded as streams from different logical channels, which are identified by a unique Logical Channel Number (LCN), in the range from 0 to 65535. LCN 0 shall be permanently assigned to H.245 control channel. Other channels can be assigned to other streams such as video/audio. Different logical channels would be multiplexed into one packet according to some rules which are negotiated by two peers at the beginning of the communication. These rules are described in the forms of multiplex table. A multiplex table has maximum 16 different table entries. And each of them would define a specific multiplexing pattern. For each packet, the terminal will choose one of these patterns to do the multiplexing.

2.3 Multi-point Conference for 3G

Based on the existing point-to-point 3G video calls, multi-point conference becomes increasingly important as it permits a large number of mobile terminals to concurrently participate in a meeting. Multi-point conferences involve three or more terminals each time, and call control and media handling in multipoint conferences are more complicated than that in point-to-point calls.

Multipoint Control Unit (MCU) [11] supports multi-conferencing between three or more terminals and gateways. There are generally two types of multi-point conferences controlled by MCU as shown in Fig. 2 below:

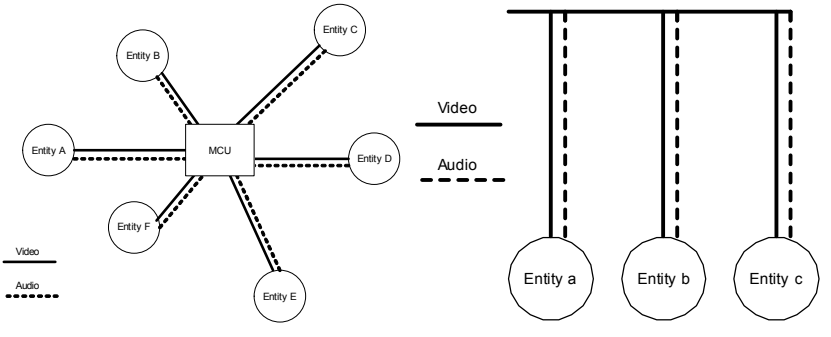

Fig. 2. Centralized and decentralized multi-point conferencing

(1) Centralized multipoint conference: All participating terminals are point-to-point connected to a MCU. MCU is regarded as the centre of the conference. Its functions include audio mixing, data distribution and video switching/mixing and sending back the resulting media streams to the participating parties. In this model, the endpoints that participate in the conference are not required as powerful as in the decentralized model. Each endpoint only has to encode its locally produced media streams and decode the set sent by the MCU. The MCU's capability to provide custom-mixed media streams allow otherwise constrained endpoints to participate in conferences.

(2) Decentralized multipoint conference: The MCU is not involved in this operation and the terminals communicate directly with each other. If necessary, the terminals assume the responsibility for summing the received audio streams and selecting the received video signals for display.

Thus, differing from the participants in point-to-point video calls, the MCU in centralized conferences and terminals in decentralized conferences are required to have the capability of handling multiple media streams from multiple end points and the original H.223 protocol needs to be enhanced for such multi-point situations.

3 Design and Implementation of Enhanced H.223

In this section, we will firstly give an introduction to our prototype video conference system, which includes a full implementation of 3G-324M protocol stack. Then we will focus on the H.223 part of 3G-324M. Enhanced capabilities to this recommendation, such as concurrent handling of multiple media streams and conference management, will be discussed.

3.1 Prototype Video Conference System

A prototype video conference system called *anyConference* has been implemented to test our 3G-324M protocol stack [5]. The system includes four general modules and each module is designed to be supported by other modules through various functional calls:

(1) H.324 Module -- This is a main module of the system which is responsible for cooperating with its umbrella modules such as H.245 Control Module and H.223 Multiplexing Module, to perform necessary functions such as call set-up etc.

(2) Video Capture/Display Module -- This module is designed for encoding and displaying the video captured from the local camera. For video codec, an open implementation of H.263 is used to encode video frames collected from the camera and decode the video frames from remote terminal. For video display, basic Windows Multimedia Software Development Kit (SDK) such as 'DrawDibDraw' is applied to handle the video as the sequence of Bitmap images.

(3) Audio Recorder/Player Module -- The Recorder module is responsible for capturing the audio data from the microphone devices. These audio data is then passed to the corresponding modules by Call-back functions once the buffer is full. The Player module is responsible for playback of the audio data received from the remote terminal.

(4)Socket Interface -- This is a network processing module provides a channel for the system to interact with air interface.

3.2 Handling of Multiplex Table

According to H.223 standard, there should be four different kinds of streams from the upper layer: control information, data information, audio and video streams. Every information stream is identified as a logical channel with a unique LCN. And the multiplex table entry specifying a LCN and the corresponding data length can describe how a data packet is multiplexed.

The data structure of multiplex table entry, which is called multiplex descriptor, takes the form of an element list. Each element in the list represents a slot of data from a specific information source. A typical example of element list with two elements is shown below

{LCN1, RC 24}, {LCN2, RC UCF}

where LCN is the logical channel number; RC is the repeat count and UCF is until closing flag. In the first element, RC 24 means that the first 24 bytes of the packet will be filled with data from logical channel 1. In the second element, RC UCF means that after the 24 bytes from logical channel 1, bytes from channel 2 will be filled in the packet until the closing flag (end of the packet). More specific description about the multiplex table entry could be found from [3] and [4].

For the simple multiplex patterns, this kind of multiplex descriptor is easy to handle. However, when the descriptor becomes complex, it will not be so easy to manipulate. The problem is that the multiplex descriptor uses a nested form for complicated multiplex patterns. Each element in the element list can be extended to a sub-element-list, which also contains other elements. Thus, when the structure is complicated with many sub-lists in the descriptor, the processing overhead may be high as recursive function calls will be needed to handle the nested lists.

In order to tackle the problem stated above, we apply a serialization approach [6] to the multiplex table, which transforms the 'nested' structure of a multiplex descriptor by flattening it into two serialized linear lists. The key point for the approach is that RC UCF will appear only once in the multiplex descriptor, i.e., only one part would be repeated until the end of the packet. Thus the whole descriptor could be divided into two parts: the RC part with finite data length; the UCF part, which is repeated until the closing flag. The whole serialization process could be divided into two steps, as shown in Fig. 3 below. At the first step, we find the point where the UCF part starts and divide the whole descriptor into the RC part and the UCF part. Then at the second step, we serialize the two parts into two separate lists of Descriptor Atoms.

Descriptor Atom is used to distinguish that our concept is different from the element we have described above. The Descriptor Atom is a very simple data structure as shown in the Fig. 3. The logical channel number specifies the information source. The repeat count is a finite number that specifies how many bytes from that source would be filled in. And it also contains a pointer which links up the following atom. As it is called an 'Atom', it is indivisible, and can not be extended to be a sub-list. Using these two serialized lists can save much processing time during the multiplexing process and introduces less overhead for the modification of the multiplex descriptors [6].

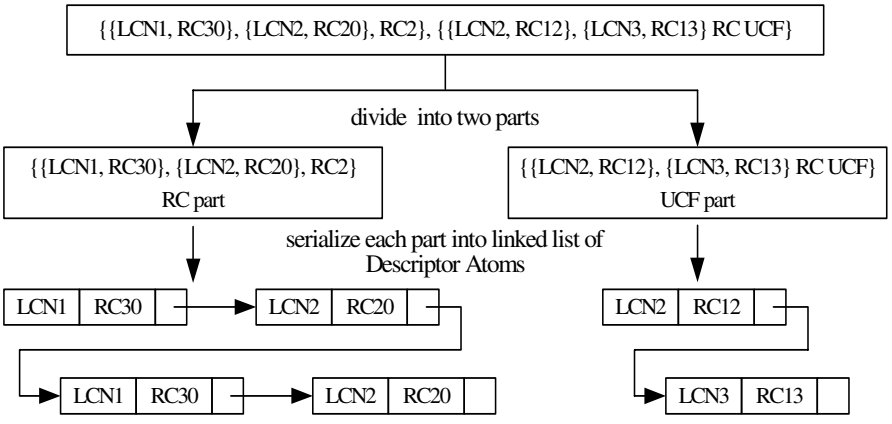

Fig. 3. Example of serialization

3.3 Structure Design of the Enhanced H.223 Module

Original H223 Design. Due to the convenience and efficiency of object-oriented approach, the major components of 3G-324M protocol stack, including H.324, H.245 and H.223, are designed and implemented as classes. H.324 class performs the negotiation about video/audio settings, etc., with the remote terminal through H.245 class, and handles the video/audio data exchange through H.223 class. The H.223 and H.245 classes also keep a pointer to each other so that the H.245 class can send out or receive control messages through H.223. The H.223 can also retrieve the information about various settings from H.245.

Thus, simply speaking, H.223 provides the sending/receiving interface for the above layers. In the point-to-point conversations, a H.223 object with a send() function, a recv() function and a buffer set for multiplexing/demultiplexing and sending/receiving, is already capable to handle all the tasks. However, for multi-point conference, H.223 might need to handle multiple incoming streams and a number of outcoming streams concurrently. Therefore, a more complex structure design might be needed.

Improved MultiH223 Design. Firstly, decomposition is applied to the original H223 design. The original H223 class is divided into two separate parts: H223_Sender assumes the responsibilities of multiplexing and sending out the outcoming streams; meanwhile, H223_Receiver handles the task of receiving the incoming streams and demultiplexing. Both H223_Sender and H223_Receiver provide convenient sending or receiving interfaces and keep their own buffer set for concerned operations.

After that, in order to handle the multiple media streams in a multi-point conference, the new designed MultiH223 module will keep two separate linked lists of H223_Senders and H223_Receivers. Thus in a real-time multi-point conference,

the MultiH223 module will initialize a new sender/receiver at any time to handle a new incoming/outcoming stream. On the other hand, the sender/receiver can be destroyed at any time during the conference to release the resources.

3.4 Simple Management of the Multi-point Conference

Although H.245 has already defined many messages and procedures for the set-up of a point-to-point video call, there is no information defined for a multi-point conference. Thus, in order to make the set-up and management of the multi-point conference more convenient, some simple conference management functions have also been added into our H.223 module.

As noted in section 2, maximum 16 different multiplex descriptors can be defined to describe 16 different multiplex patterns. For each H.223 data packet, there is an MC (Multiplex Code) field which identifies the multiplex descriptor used for this packet. However, in practical experiments, we have found that current 3G mobile terminals only use 3 different multiplex descriptors (MC = 0, 1, 2) at most. Thus, we can use MC = 15 to identify that the content in this packet is information used for conference management. The following are the major control messages defined to accomplish the most common management functions:

Table 1. Major Control Messages Defined for Conference Management

Command	Description
CONNECTDEFAULT	Request to establish a connection to the remote terminal.
CONNECTED	Accept the connection request from the remote terminal.
DISCONNECT	Leave the current video conference.
KICKOFF	Expel the specified terminal from the conference.
TALK_O	Request a token to speak out in the group
TALK_X	Return token to the group
CONNECTION_REJECTED_F	Notify that Max. Number of Clients is Reached
CONNECTION_REJECTED_I	Notify that illegal Connection! Please Connect to Host
CONNECTION_REJECTED_B	Notify that No Permission to Join this Conference

In our management mechanism, one terminal will act as the host of the conference. The host is the creator of the conference and keeps the real-time information of the conference, such as the list of clients and their IDs. It is also responsible to broadcast the conference information to participating clients timely. A terminal can send the CONNECTDEFAULT message to the host to indicate its intention to join in the conference. The host has to make the admission control decisions according to terminal information, such as ID, or the constraints of the conference, such as maximum number of participants. After that, the host sends back CONNECTED or CONNECTION_REJECTED_F/I/B message to admit or decline the requesting terminal.

For the voice channel scheduling, our conference system allows only one terminal to speak at any time, and the voice is generally broadcast to all participants of the conference. Similar to the mechanism used in token-ring network, a terminal is

granted a permission to broadcast its voice data over the network as long as having the "token". However, the way of token passing is different from the token-ring network. The grant of token is governed by the host of the conference. A terminal has to send a request `TALK_O` to the host in order to speak out in the conference. After finished speaking, the terminal has to send an indication `TALK_X` to the host for return of the token.

4 Performance Evaluation

4.1 System Interface

Our prototype system is implemented on PC but not a mobile handset. The PC can access 3G mobile network through 3G modem card. However, as the current 64 kbps video call bandwidth is not large enough to support multi-point video conferences, we test the performance of our system on the physical channel of 802.11 WLAN.

Fig. 4 shows the user interface of our system. Currently, the terminal supports the display of four different video channels; while there can be more than 4 clients in the conference. For audio channels, as noted in section 3.4, only one terminal is permitted to speak at any time.

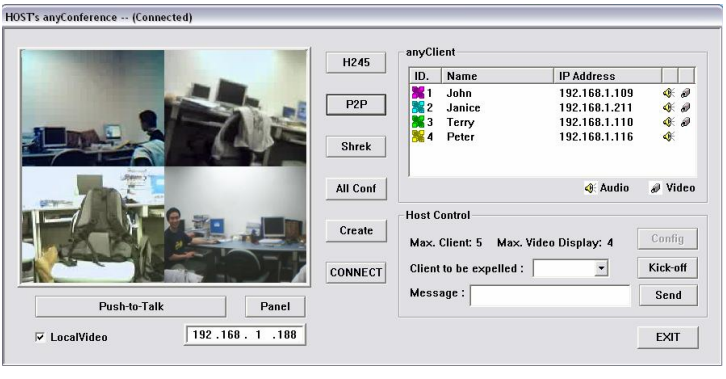

Fig. 4. GUI of the prototype system - anyConference

4.2 Performance of H.223 Implementation

Multiplex Descriptor Serialization. The performances of original and serialized multiplex descriptors are compared. With the original descriptor, the processing cost of multiplexing one packet can be divided into two parts: C_m is the cost of filling bytes into a packet, i.e. the cost of memory operations; C_n is the cost of handling the nested descriptor structure, i.e. the cost of recursive function calls.

For multiplexing of k packets, the total cost is $C(k)=k*(C_m+C_n)$. With the serialized multiplex descriptors, the processing cost of multiplexing one packet is C_m only. However, there is an additional cost to serialize the original descriptors at the initialization stage, which is defined as C_i.

For the initialization stage, maximum 16 nested structures need to be handled. Thus it is reasonable to assume that $C_i \approx 16*C_n$. For multiplexing of k packets, the total cost can be calculated as: $C(k)' = C_i + k*C_m \approx 16*C_n + k*C_m$.

A number of experiments have been done to evaluate the parameters C_m and C_n. The result is as follows: time taken for handling a nested descriptor structure (C_n) for one million times is 0.0472s and the time taken for filling bytes into a packet (C_m) for one million times is 0.353s. From the measurement result, we can estimate that: $\frac{C_n}{C_m} \approx \frac{0.0472}{0.353} = 13.4\%$. For the multiplexing of k packets,

$$\frac{C(k)'}{C(k)} = \frac{16*C_n + k*C_m}{k*C_n + k*C_m} = \frac{16*0.134 + k}{0.134k + k} \xrightarrow{k \to \infty} \frac{k}{0.134k + k} = 88.2\% .$$

Thus, the overall execution cost can be optimized by 11.8% when serialization is used as compared with recursive approach.

MultiH223 Module and Simple Conference Management. With the MultiH223 module and the simple conference management functions added into H.223, a video conference could be set up and managed conveniently and efficiently. Based on our conference testing over 802.11 WLAN, the performance of video and audio communication is satisfactory and stable.

5 Conclusions and Future Work

We have presented an efficient implementation of the data multiplexing protocol H.223, targeted for the case of multi-point video conferences. The design and implementation of the protocol is based on our prototype system '*anyConference*'. With serialization approach to the multiplex descriptors, support for multiple media streams and simple conference management functions, our H.223 implementation makes the set-up and performing of video conferences much more convenient and efficient. With tests on the prototype system, the performance of our protocol implementation is satisfactory and stable.

For the future work, we intend to do some work for the transcoding and compatibility between 3G-324M terminals and H.323 terminals. We'll try to make our protocol stack and conference system compatible with both 3G-324M and H.323. Thus, the video conference can be performed over both 3G mobile and internet networks, with support of MCUs and gateways.

References

1. ITU-R Rec. PDNR WP8F, Vision, Framework and Overall Objectives of the Future Development of IMT-2000 and Systems beyond IMT-2000, 2002.
2. ITU-T Rec. H.324, Terminal for low bit rate multimedia communication, March 2002.
3. ITU-T Rec. H.245, Control protocol for multimedia communication, July 2003.
4. ITU-T Rec. H.223, Multiplexing protocol for low bit rate mobile multimedia communication, July 2001.

5. B. Han, H. Fu, J. Shen, P. O. Au and W. Jia, Design and Implementation of 3G-324M - An Event-Driven Approach, Proc. IEEE VTC'04 Fall.
6. H. Fu, B. Han, P. Au and W. Jia, Efficient Data Transmission Multiplexing in 3G Mobile Systems, Proc. Globe Mobile Congress 2004.
7. ITU-T Rec. T.120, Data protocols for multimedia data conferencing, 1996.
8. ITU-T Rec. H.263, Video coding for low bit-rate communication, 1998.
9. ITU-T Rec. H.261, Video codec for audiovisual services at p × 64 kbit/s, 1993.
10. ITU-T Rec. G.723.1, Speech coders: Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s, 1996.
11. ITU-T Rec. H.243, Procedures for establishing communication between three or more audiovisual terminals using digital channels up to 1920 kbit/s, 2000.
12. The 3rd Generation Partnership Project (3GPP): http://www.3gpp.org.
13. 3GPP TS 26.071 V4.0.0, AMR Speech Codec; General Description, 2001.
14. 3GPP TS 26.111 V5.1.0, Codec for circuit switched multimedia telephony service: Modifications to H.324, June, 2003.