Efficient 3G324M protocol Implementation for Low Bit Rate Multipoint Video Conferencing

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

Fung Po TSO and Lizhou Zhang
Department of Computer Science, City University of Hong Kong
83 Tat Chee Ave., Kowloon, Hong Kong, SAR China
{poscotso, lizhuocs}@cityu.edu.hk

Abstract[*]**--This paper discusses an efficient design and implementation of control and multiplexing protocols H.245 and H.223, which is an important part of 3G324M protocol stack, for mobile wireless video conferencing. Our implementations aim to support the multipoint video conferencing with the capability of transmitting/receiving multiple video/audio streams simultaneously. Conference managements such as admission control, video/audio channel management are also discussed. As a result, the implementation improves efficiency and makes the conference more convenient to set up and to operate. Our prototype system is stable and its performance is satisfactory.**

Index Terms – 3G, 3G324M, H.245, H.223, Low bit Rate, Video-Conferencing

I. INTRODUCTION

Currently, multimedia streams of 3G video calls is transferred using 3G324M protocol [1] derived from 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 control protocol H.245 [3] and multiplexing/demultiplexing protocol H.223 [4] that specifying connection setup, negotiation and tear down; data multiplexing/demultiplexing. 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 3G324M standard for 3G wireless networks [14].

This paper describes our recent prototype design and implementation called *anyConference*, which is a multi-point video conferencing system over the mobile terminals, based on our implementation of 3G324M protocol stack. We mainly focus on the discussions of enhanced implementations of control protocol H.245 and multiplexing protocol H.223, which are 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 Sec. 2. Sec. 3 illustrates the prototype video conference system first, and then explains the efficient algorithms and implementations of H.245 and H.223 protocols for enhanced 3G324M. Sec. 4 describes modified multipoint multiplexing algorithm and then conference management such as admission control and audio channel scheduling are also discussed. Sec. 5 gives evaluation of our implementation of the prototype system. Sec. 6 concludes the paper and discusses the future work.

II. BACKGROUND

■ *A. Overview of 3G324M/H.324M Standard and Protocol Stack*

The architecture of 3G-324 protocol stacks is shown in Fig. 1 and the main protocol components are in the middle part of Fig. 1, these are illustrated as follows:

- H.324 — the Base Protocol, which consists of main protocol components.
- H.245 — specifies the Call Control Protocol which provides the end-to-end signaling for proper operations of the H.324 terminals..
- H.223 — provides a multiplexing/demultiplexing service for the upper-layer applications.

[*] This effort is partially sponsored by City University of Hong Kong APR research project no. 9610027 and grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. 9041129 (RGC Ref. No), CityU 113906].

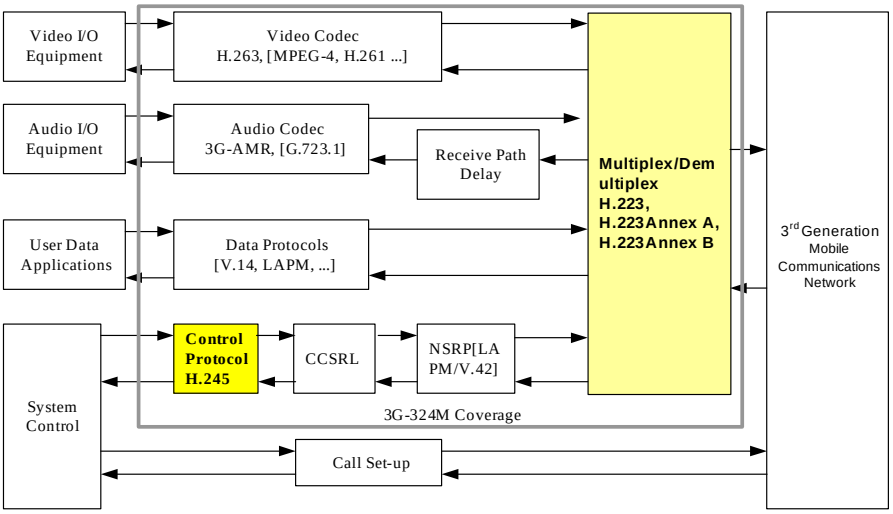

Figure 1. Architecture of 3G324M Protocol Stack

- Standards for video/audio codecs, such as H.263 [8], H.261 [9], G.723.1 [10], AMR (Adaptive Multi Rate) [13] etc. are also used in the H.324 stack.
- Equipments – audio and video I/O hardware equipments.

B. 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 air/links), 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 standard is divided into two layers:

1. Adaptation Layer (AL): The first layer 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 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. Multiplex Layer (MUX): The second layer 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.

C. Control Protocol H.245

H.245 standard has been defined to be independent of the underlying transport mechanism, but is intended to be used with a reliable transport layer, which provides guaranteed delivery of correct data. H.245 carries on connection setup and capability exchange between two connection points and specifies syntax and semantics of terminal control messages as well as the procedures for in-band negotiation at the start or during the communication. The messages cover receiving and transmitting capabilities as well as mode preference, logical channel signaling and control. The message syntax is defined using ASN.1 formatted data [12] and is transformed into bit-stream based on ASN.1 encoding standard of Packed Encoding Rules (PER) [11].

Initially, H.245 Control Channel is routed between the endpoints through gatekeepers. MCU-endpoint signaling is governed by H.245 control channel between endpoints and the multipoint control (MC) within the Multipoint Control Unit (MCU). When the conference switches to multipoint, the MC at the gatekeeper can be activated for subsequent operations. If one or both of endpoints have an MC, normal set up procedures is required. H.324 terminals may be used in multipoint configurations via interconnection through MCUs as illustrated in next subsection.

D. 3G Multi-point Conference

Based on the point-to-point 3G video calls, multipoint conference becomes increasingly important as it permits a large number of mobile terminals to concurrently participate in a meeting. Multipoint 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, Centralized multipoint conference and Decentralized multipoint conference as shown in Fig. 2 below:

Centralized multipoint conference: As illustrated in Fig. 2(a), all participating terminals are point-to-point connected to a MCU. MCU is regarded as the centre of

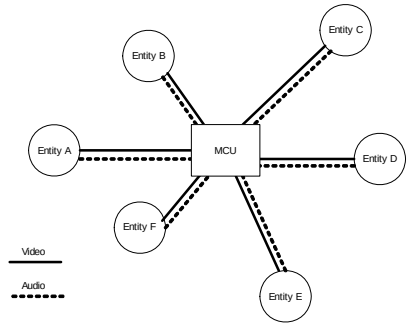

Figure 2 (a). Centralized Multipoint Conferencing

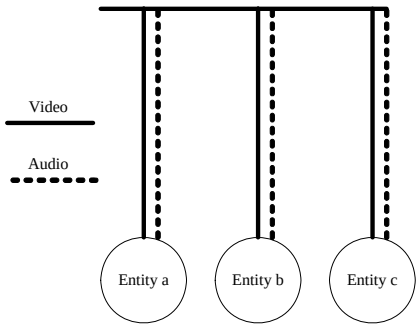

Figure 2 (b). Centralized Multipoint Conferencing

the conference. Its functions include audio mixing, data distribution and video switching/mixing and sending back the result 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 (to be discussed below). 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.

Decentralized multipoint conference: As illustrated in Fig. 2(b), 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 centralized conferences, the terminals in decentralized conferences are required to have the capability of handling multiple media streams from multiple endpoints. Thus the original control protocol H.245 and multiplexing protocol H.223 need to be enhanced for both point-to-point and multi-point situations.

III. ENHANCED DESIGN AND IMPLEMENTATIONS

We first give introduction to our prototype video conference system, which includes the implementation of 3G324M protocol stack. Then we focus on the discussion of enhanced implementations of H.245 and H.223 within 3G324M stack. Enhanced functions such as capability exchange, concurrent handling of multiple media streams and conference management will be discussed.

■ *A. Prototype Video Conference System*

A prototype video conference system called *anyConference* has been implemented based on 3G324M protocol stack [2]. The system includes four general modules and each module is designed to be supported by other modules through various functional calls:

- H.324 Module -- This is a main module of the system which is responsible for the cooperation 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.
- 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.
- 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 terminals.
- Socket Interface -- This is a network processing module provides a channel for the system to interact with air interface/physical layer.

■ *B. Efficient algorithms for the enhanced 3G324M*

1. Efficient System Control – An Event Driven Approach

To coordinate the different parts of the 3G324M protocol stack efficiently, our implementation for 3G324M applies event-driven programming [5] which is characterized by triggering the execution of protocol in an arbitrary order, rather than in a pre-determined order such as in a sequential control procedure. This approach fits to the behaviors of network protocols, as their actions may be unpredictable. Based on event-driven programming, it is easy to control the software architecture and maintain a cooperation of the protocol stack in the overall system.

Although events are not defined explicitly in 3G324M specification, there are generally three types of events in 3G324M: *message*, *primitive* and *timeout* events. Message events occur when the terminal receives control messages from another terminal; primitive events happen when there is an interaction between 3G324M and its upper layer applications; and timeout events occur when timer expires in 3G324M protocol stack. Fig. 3 shows the overview of our Event Driven control system. In this Event-Driven control approach, two core parts are defined as Event Generator and Event Dispatcher. Event Generator generates corresponding events based on the current state and takes care of the

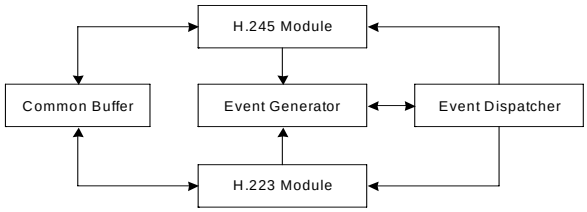

Figure 3. An Overview of Event Driven Control System

maintenance of event queue and timer chain. Event Dispatcher calls related handling procedure based on the finite state machine definitions and it may also call Event Generator to generate other events. Our implementation uses event generator and dispatcher to coordinate H.245 and H.223 protocols to send/receive multimedia data as well as the control messages. Thus, the Event-Driven mechanism is executed across the entire 3G324M protocol stack.

2. Optimized Message Processing in H.245

Before H.245 messages are sent to H.223 module for multiplexing, all messages are expected to be packed into *Multimedia System Control Message* (MSCM) and then encoded by PER. Thus messages received from H.223 module should be decoded and unpacked for further processing. The implementation of packing/unpacking and encoding/decoding involves some similar memory access and bitwise logical operations. To better utilize the limited resources of a terminal, our implementation intends to compress and integrate some of the procedures. The recent implementations of ASN.1 formatted data encoding/ decoding apply *multiple-step tree-based message transformation* [8], which organizes routines and procedures for message processing in a tree-like structure. In order to encode a H.245 message, the top-level encoding routine calls the lower level encoding routines, and the encoding routines at different levels set values to the corresponding items in a message. The processes go ahead until the messages are encoded into a bit-stream eventually. The multiple-step tree-based approach works efficiently in dealing with code redundancy and maintenance. However, the encoding/decoding operations are still complicated because the lengthy and complicated encoding rules have to be executed step by step along the tree-structure as defined in PER. In our implementation, we propose a simple and efficient approach on implementing PER called *single-step direct message trans-formation* in which the tree-structured multi-step message encoding/decoding can be replaced by a single-step of message encoding/ decoding. An element in a message is taken as a node and the nodes in a message are linked together into a list that represents the message. Then all messages are further linked together for quick mapping to the syntax of MSCM. In [11], PER rules are defined according to different data types, thus we have implemented the explicit encoding/decoding functions for each data type accordingly. When encoding/decoding a message, the nodes in the message list are encoded one by one. This approach is still easy to manage, and we have packed the procedures into our final implementation. As a result, based on this approach, the PER codec in our implementation is simplified without increasing the workload of encoding/decoding operations or modifying the syntax and semantics of the messages. Fig. 4(b) shows the processing flow of our single-step implementation where SM stands for Specific Message, EM stands for Encoded Message. In the normal operation, we need to pack a specific message into MSCM in accordance with tree-structure specification, and then the MSCM is passed to PER encoder to be encoded into a bit-stream following the working process as shown in Fig. 4 (a). In our implementation, in order to simplify the process, we have combined the packing and encoding procedures into one procedure by directly transforming the message into a bit stream.

3. Multiplexing Optimization in H.233

According to H.223 standard, there should be four different kinds of streams from the upper layer: (1) the control information between different terminals or between terminals and the Multipoint Control Units (MCUs) as defined in H.245; (2) information from data applications such as T.120 [7] data for real-time audio-graphics conferencing; (3) audio stream encoded by audio codecs defined in 3G324M, such as Adaptive Multi-Rate (AMR) codec; (4) video stream encoded by video codecs defined in 3G324M, such as H.263 and MPEG-2. 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 will 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},

in which LCN is the logical channel number; RC is the repeat count and UCF is the 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. 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 can be found from in [3] and [4].

Multiplex descriptor uses a nested form for complicated multiplex patterns. For the simple multiplex patterns, this kind of multiplex descriptor is easy to handle. However, when the descriptor becomes complex, performance penalty may be introduced. We consider that 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, this may result in low processing performance.

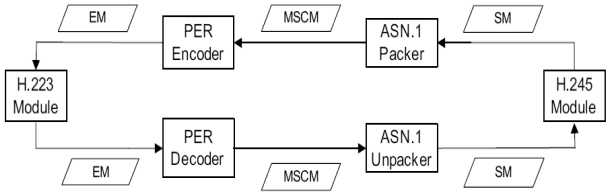

Figure 4 (a). Normal Implementation Approach

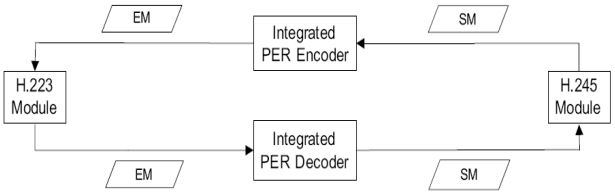

Figure 4 (b). Characteristics of processing flow in our implementation

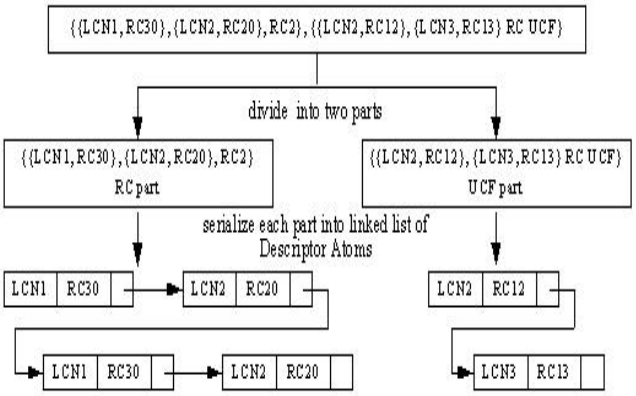

Figure 5. Example of Serialization

In order to tackle with the problem stated above, we apply a serialization approach [6] to the multiplex table, which will transform 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 will repeat until the end of the packet. Thus the whole descriptor can be divided into two parts: the RC part with finite data length; the UCF part, which is repeated until the closing flag. Fig. 4 shows an example: First finds the point where the UCF part starts and then divides the whole descriptor into the RC part and UCF part; and second step serializes the two parts into two separate lists of *Descriptor Atoms Descriptor Atom* is used to distinguish our concept which differs from the element we have described above. The Descriptor Atom is a very simple data structure as shown in Fig. 5. The logical channel number specifies the information source. The repeat count is a finite number that specifies how many bytes for this source will 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].

IV. EFFICIENT ALGORITHMS FOR MULTIMEDIA VIDEO CONFERENCING MANAGEMENT

■ *A. Multipoint Multiplexing Algorithm*

Generally speaking, H.223 provides the sending/receiving interface for 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 out-coming streams concurrently. Therefore, the original approach may not fit such need.

To solve this problem, instead of developing a complex multiplexing algorithm, we have developed a simple approach called MultiH223 Module which is capable of handling multi-point sending/receiving and multiplexing/demultiplexing. In our design, as illustrated in Fig. 6, a MultiH223 Module consists of two parts, a list of H223_Receivers and a list of H223_Senders, each H223_Receiver and H223_Sender pair is responsible for interfacing with one terminal.

For the implementation, we first apply decomposition approach to the original H223 design by dividing H223 class into two separate parts: H223_Sender assumes the responsibilities of multiplexing and sending the outgoing streams; meanwhile, H223_Receiver handles the task of receiving the incoming streams and performs the demultiplexing for the incoming media streams. Both H223_Sender and H223_Receiver provide convenient sending or receiving interfaces and keep their own buffer set for the concerned operations.

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

■ *B. Management of Multipoint Conference*

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

As noted in Sec. 2, maximum 16 different multiplex descriptors can be defined to describe 16 different multiplex patterns. For each H.223 data unit (we use *unit* for the circuit switching case and in case of packet switching, we refer the unit as a packet), there is an MC (Multiplex Code) field which identifies the multiplex descriptor used for this unit. However, in practical experiments, we have found that current 3G mobile

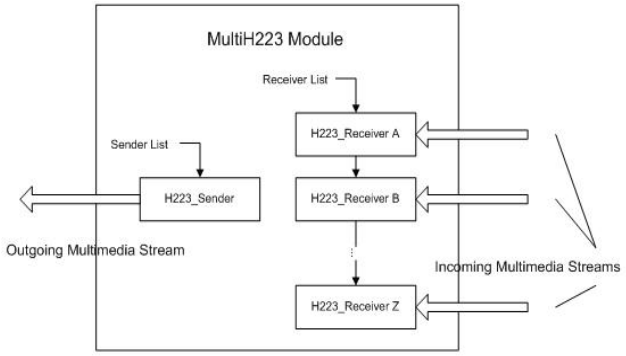

Figure 6. Linked lists approach of MultiH223 Module

terminals only use 3 different multiplex descriptors (MC = 0, 1, 2) at most. Thus, we adapt MC = 15 for conference management. Table 1 lists the major control messages defined to accomplish the most common management functions.

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 the participating clients timely. A terminal can send `CONNECTDEFAULT` message to the host to indicate its intention to join in the conference. The host has to perform the admission control decisions according to terminal information, such as ID or the constraints of the conference, e.g., the maximum number of participants. Upon the admission control, the host sends back `CONNECTED` or `CONNECTION_REJECTED_F/I/B` (Table 1) message to admit or decline the requesting terminal.

During the conference, a terminal may withdraw from it at anytime and this terminal may be the host of the conference. Thus, in order to manage all participants in the conference and help the conference goes smoothly, we need to re-elect a host of the conference. Since the re-election cost is promotional to the number of re-election, our strategy is the less re-election the better. To accomplish this requirement, host re-election is performed based on signal strength of a participated terminal, terminal capacity and predicted terminal withdrawing time where signal strength weighted the most because the weaker the signal the more likely disconnect from the network.

Due to the limited bandwidth in the 3G environment, the flow of video streaming need to be controlled throughout the whole conference to avoid data congestion and collision. In out implementation, for video channel scheduling, we adopt round robin mechanism. Under this mechanism, time slot (token) is reserved explicitly for each terminal and, therefore, a terminal must broadcast its video stream when a token is granted to it. The host is responsible for determining the length of a time slot based on current network quality. The advantage of introducing this mechanism is that

TABLE 1.
Major Control Messages Defined for Conference Management

Command	Description
`CONNECTDEFAULT`	Request to establish a connection to the remote terminal.
`CONNECTED`	Accept a connection request from 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

bandwidth of 3G network can be fully utilized and the flows are controlled.

For the voice channel scheduling, the conference system allows only one terminal to speak at any time, and the voice is generally broadcast to all participants in 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 it has the "token". However, the way of token passing differs 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 the speaking, the terminal has to send an indication `TALK_X` to the host to return the token. Moreover, to be realistic, the speech is interruptible once another terminal request to speak, but that terminal should return the "token" to previous terminal immediately after its speech.

V. SYSTEM EVALUATION

■ *A System Interface*

Our prototype system is implemented on both PC platform and 3G handsets. 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. 7 shows the user interface of our system, conferencing windows are located on the left hand side while conferencing controls are put on the right hand side. 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 described in Sec. 4, only one terminal is permitted to speak at any time.

■ *B. Some Experimental Results*

With the original descriptor, the processing cost of multiplexing one packet can be divided into two parts:

- C_m -- the cost of filling bytes into a packet, i.e. the cost of memory operations;

Figure7. GUI of the prototype system – anyConference

- C_n--the cost of handling the nested descriptor structure, i.e. the cost of recursive function calls.

Thus, a number of experiments have been carried out in PC (WinTel, P4 Processor) Platform to evaluate the parameters C_m and C_n. The procedures of these experiments have been conducted as follow:

1. Define system start time as 0ms.
2. Record system time while starting filling bytes into a packet.
3. Record system time after repeating step 2 for one million times.
4. Record system time while starting handling the nested descriptor structure
5. Record system time after repeating step 4 for one million times.

The results for step 3 and step 5 are shown in table 2 and table 3 respectively: average time taken for handling a nested descriptor structure (C_n) for one million times is 0.0472s and the average time taken for filling bytes into a packet (C_m) for one million times is 0.353s.

■ *C. Analytical Result*

In this session, the performances of original and serialized multiplex descriptors are compared. 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 we assume $C_i \approx 16 * C_n$. Thus, for multiplexing of k packets, the total cost can be calculated as:

$C(k)'=C_i+k*C_m \approx 16*C_n+k*C_m$.

From the above experimental results, we can estimate that: $\frac{C_n}{C_m} \approx \frac{0.0472}{0.353} = 13.4\%$. For the multiplexing of k packets,

$$\frac{C(k)'}{C(k)} = \frac{16 * C_n + k * C_m}{k * C_n + k * C_m} = \frac{16 * 0.134 + k}{0.134k + k}$$

$$\xrightarrow{k \to \infty} \frac{k}{0.134k + k} = 88.2\%$$

.

TABLE 2.
Time taken for filling bytes into a packet for one million times

	Start Time	End Time	Time Spent
1	1031 ms	1078 ms	47 ms
2	1031 ms	1078 ms	47 ms
3	1031 ms	1079 ms	48 ms
4	1031 ms	1078 ms	47 ms
5	1031 ms	1078 ms	47 ms
6	1031 ms	1078 ms	47 ms
7	1031 ms	1078 ms	47 ms
8	1031 ms	1078 ms	47 ms
9	1031 ms	1078 ms	47 ms
10	1031 ms	1079 ms	48 ms
		Average	47.2 ms

TABLE 3.
Time taken for handling a nested descriptor structure for one million times

	Start Time	End Time	Time Spent
1	1078	1431	353 ms
2	1078	1431	353 ms
3	1079	1432	353 ms
4	1078	1431	353 ms
5	1078	1431	353 ms
6	1078	1431	353 ms
7	1078	1431	353 ms
8	1078	1431	353 ms
9	1078	1431	353 ms
10	1079	1432	353 ms
		Average	353ms

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

■ *D. Actual Performance*

With the MultiH223 module and the 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 WLAN of IEEE 802.11, the performance of video and audio communication is satisfactory and stable.

VI. CONCLUSIONS AND FUTURE WORK

We have presented an efficient implementation of control and data multiplexing protocols H.245 and H.223 for enhanced 3G324M, hence migrated these to support multi-point video conferences (*anyConference*) in 3G environments. With serialization approach to the multiplex descriptors, support for multiple media streams control and simple conference management functions, our H.245 and H.223 implementations make 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.

Currently, we are investigating the transcoding and compatibility between 3G324M terminals through heterogeneous environment. We'll try to make our protocol stack and conference system compatible across 3G324M, WiFi and H.323. Thus, the video conference can be performed over both 3G mobile and internet networks, with support of MCUs and gateways.

VII. 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 3G324M - 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 x 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 kbps, 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.
[15] M.-C. Yuen, J. Shen, Weijia Jia and B Han, Simplified Message Transformation for Optimization of Message Processing in 3G324M Control Protocol, Proc. of 2005 International Conference on Computer Networks and Mobile Computing (ICCNMC 2005)

Weijia Jia received his PhD degree from Polytechnic Faculty of Mons, Belgium in 1993.

He is the chairperson of Anyserver Lab of City University of Hong Kong. His research interests include Distributed Systems, Internet Computing, Networking & Reliable Communication Protocols, Anycast and Multicast for Mobile Computing

Fung Po TSO received his Eng. degree in computer engineering and information technology from City University of Hong Kong in 2005.

His research interests include 3G technology, wireless mesh network, embedded system design.

Lizhuo Zhang received his M.S of computer science from Central South University, China in 2001.

In 2006, he joined Anyserver Lab., City University of Hong Kong as a senior research associate. His research interests include digital image processing, telecommunication, wireless communication.