Implementation of Real-Time Video Conference System Using high speed Multimedia Communication

Hyen Ki Kim
Department of Multimedia Engineering
Andong National University
388 Songcheon-dong Andong city Kyungbuk
Rep. of Korea
hkkim@andong.ac.kr http://www.andong.ac.kr

Abstract: - Recently, Peer to Peer(P2P) networks become more and more popular. This paper describes an implementation of real-time video conference system using high speed multimedia communication. The proposed real-time video conference system has hybrid Peer to Peer architecture based on a client-server and peer-to-peer, where client-server is used for exchange of account management, client list and status information and P2P is used for the real-time video conference. The proposed real-time video conference system decreases the traffic of server, and can cuts down the load of multimedia communication. Because the multimedia data is decentralized to client by hybrid peer-to-peer network architecture. Also, the proposed system is implemented and tested by the real-time video conference system using communication protocol and application software through high speed multimedia communication

Key-Words: - Real-Time, Video conference, High speed, Hybrid, P2P, Multimedia, Communication

1. Introduction

In the video conference system, multimedia data are processed compression and decompression function for real time simultaneously. Also, high-speed telecommunication network is necessary that can transfer large scale multimedia data such as audio and video. It is now feasible to deliver video streams over the Internet as the rapid development of multimedia and Internet technology. Most of video streaming systems adopt the client-server model and each video stream is carried on a point-to-point real-time protocol(RTP) connection. As a consequence, the video server likely to become the bottleneck of the system, making scalability become the critical issue of video streaming. A client mainly retrieves media data from the media server directly in the client/server extension approach. It only retrieves the media data from other client when the media server is overwhelmed. In multimedia streaming applications like video conference system, multimedia data is transmitted to one or more than one destination process in various types of communication networks[1]. The video conference systems such as skype and AOL IM is one of the applications for broad-band data transmission[2-5].

Because these systems need no dedicated lines between participants, we can communicate with each other at a lower cost. In the RTVC(Real-Time video conference) system, the video and audio data is compressed and decompressed by the Codec for real time video conference[6-9]. Also, high-speed telecommunication network is necessary that can transfer large scale multimedia data such as audio and video. To perform efficiently multimedia data in computer based video conference system, it is essential to have support from a well architecture. Peer to Peer technology support exchange of real-time communication or data between users[10-11]. Peer to Peer system process service as equivalent relation of clients.

This paper describes describes implementation of real-time video conference system using high speed multimedia communication. In the following sections, we begin with the architecture of real-time video conference system, and implementation of multiconference are discussed. Finally we discuss experiment of real-time video conference.

In Chapter 2 of this paper explains the hybrid P2P architecture. Chapter 3 explains the design of multi-party video conferencing system. Chapter 4 shows the implementation and the experiment, and the paper is concluded in Chapter 5.

2. Architecture of video conference system

The level of multimedia data for computer based video conference system is as follows.

· Audio: Stereo of CD level(44.1 KHz, 16 bit quantization)
· Video: 30 frames/sec, full colors(24 bit RGB), 640 pixel x 480 lines/frames
· Graphics: 1024 x 768(resolution), 256 colors

From video conference point of view, software only solution can't fully process the multimedia data mentioned above. So, the partial of function for multimedia data processing require the implementation of hardware[4-6]. From hardware point of view, architecture of desktop computer based video conference system to support multimedia data processing function is Fig. 1.

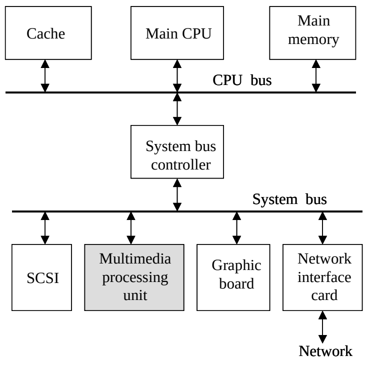


Fig. 1 Architecture of desktop computer based video conference system

From Fig. 1, The function of multimedia processing unit and bus architecture is as follows.

· Multimedia processing unit is capable of capture, playback, encoding and decoding functions of audio-visual data
· The bus is composed of architecture of hierarchical bus according to bandwidth.
· Main CPU is operating with multimedia data independently.

The block diagram of multimedia processing unit is Fig. 2. This block diagram consists of six modules; that is overlay module, audio module, video module, graphic module, network module, controller module.

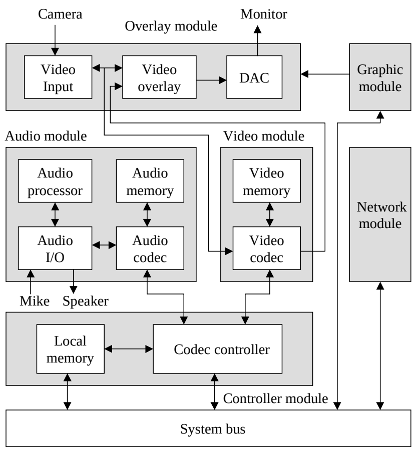


Fig. 2. Block diagram of multimedia processing unit

Overlay module mixes video and graphics data, and audio module must operate recording and playback of audio simultaneously. The video module must separate the function of compression/decompression for real time processing and the controller module is communicate with host for information exchange.

3. Hybrid P2P architecture

The RTVC(Real-Time Video Conference)system uses two separate network architecture models, client-server for the communication with the MCU(Multipoint Control Unit) Presence server for account management, contact lists, and presence status information, and peer-to-peer for the real-time video and audio conferencing and text messaging[4-5].

3.1 Client-Server architecture

The MCU component is a centralized server, which runs on a single machine, and hosts the contact-list (buddy-list) presence information database (XML), which includes lists of usernames, user_IDs(Identification), and passwords for all recognized users. The MCU also receives and publishes status changes to all logged-in users, such as when a user comes online, goes offline, becomes busy, etc. It also handles requests to add and/or remove contacts from a specific users contact list, and update user account information such as username and password. Fig. 3 shows client server architecture of MCU and RTVC clients.

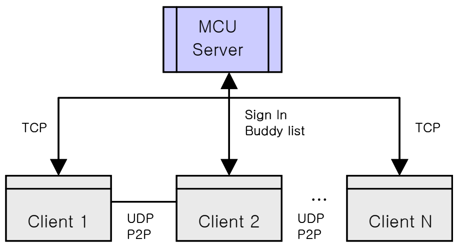


Fig. 3 Client server architecture of MCU and RTVC clients

The RTVC clients have a client-server relationship with the central MCU server. The TCP protocol is used for network communications. When a RTVC client starts up, the user must initially log in to the MCU. The user enters his username and password into a dialog box, along with the static IP address of the remote MCU server. The information is posted to the server, which verifies the information against its accounts database, and sends back either a login success or failure notification. The RTVC client also requests the MCU to send a copy of the user's buddy-list down to it.

3.2 P2P architecture

Since the RTVC client has a local current copy of the IP address and status information for each contact, the peer-to-peer network is used for the actual conference network. When a user selects one or more online contacts and starts a new conference, invitations are sent directly to each remote participant using UDP. The MCU is not involved in this process. Secondary interconnections between all the other conference members are also automatically established when remote participants accept the conference invitation. A custom session control protocol is used to handle the conference invitation, acceptance, refusal, and termination. The MCU server is not involved. Fig. 4 shows the P2P 3-way network of RTVC clients.

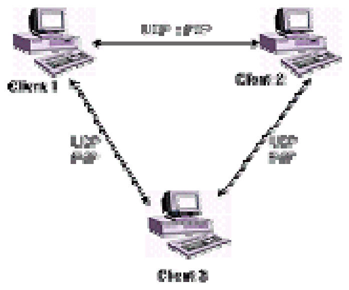
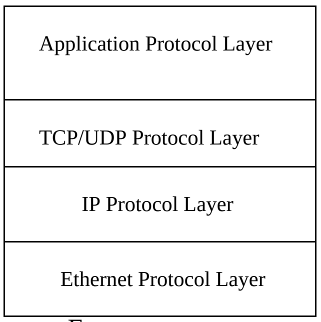


Fig. 4 P2P architecture of RTVC clients

3.3 Network subsystem

Both TCP and UDP protocols are used in the RTVC client, over IP and Ethernet. TCP is used for communications with the MCU server, for login/logout, account management, and contact list presence information. UDP is used for the peer-to-peer conference data transfers, including both the out-of-band session control protocol data and the real-time streaming of video and audio data as well as text messaging data exchanges. Figure 5 shows network layer model.



F

Fig. 5 Network layer model

This category includes the UDP streaming of audio and video data, captured by the local microphone audio capture device and the local webcam video capture device, as well as the related codecs such as the Audio Codec and the Video Codec. It is helpful to define the main network protocol models which are used in the RTVC system. They provide a general categorization of the behavior, role, and responsibilities associated with the different network operations which the system performs. This refers to the TCP client-server model, which is used for login/logoff operations, client account management such as username & password changes, and the ongoing contact (buddy) presence status information which is distributed to all logged-in users.

This category includes the UDP streaming of audio and video data, captured by the local microphone audio capture device and the local webcam video capture device, as well as the related codecs and classes such as the AudioCodec (G723.1) and the VideoCodec (Windows Media Video VCM 9 codec). Data packets sent peer-to-peer between contacts during the conference, which include the session control protocol and the real-time audio and video streaming, and text message data exchange, are handled primarily by the audio-video manager class.

4. Design of video conference system

Initially an online user will become the conference host by selecting one or more online buddies in the user list dialog and starting a call. Where, all calls are handled by same subsystem code, whether 1-to-1 or 1-to-many.

First, an array of multi-party video conference structures is dynamically allocated, since we don’t know ahead of time how many participants any conference might contain. There is a structure for the host and one for each remote participant.

Next, the pointer to the array of multi-party video conference structures is assigned to an element within the multi-conference data structure, which also stores a default conference title string, the username of the conference host, the number of conference participants (including the host), and a randomly generated conference id, thus each conference should have a unique ID.

Finally, the multi-party video conference data is assigned to a member element of the multi-party video conference global structure variable. If the initialization of the global multi-party video conference data structures is successful, then video dialog next calls into audio-video manager to send out the conference invitation packets over the network, directly to each remote participant using the peer-to-peer network model. Again, the MCU server is not directly involved in this process.

4.1 Video conference invitation process

After video dialog calls into audio-video manager, first a check is made to ensure the maximum number of supported participants has not already been reached. Then, the number of participants is retrieved from the global multiconference data. A network packet is formatted which contains both the standard packet header CMD_HEADER as well as extra data appended containing the multiconference data structures. Fig. 6 shows the network packet structure for initial CMD_MC_INVITE packet.

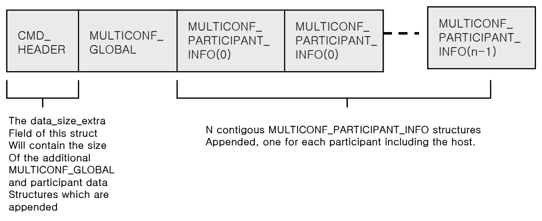

CMD_ HEADER
MULTICONF_ GLOBAL
MULTICONF_ PARTICIPANT_ INFO(0)
MULTICONF_ PARTICIPANT_ INFO(0)
MULTICONF_ PARTICIPANT_ INFO(n−1)
The data_size_extra Field of this struct Will contain the size Of the additional MULTICONF_GLOBAL and participant data Structures which are appended
N contigous MULTICONF_PARTICIPANT_INFO structures Appended, one for each participant including the host.

Fig. 6 Network packet structure for initial CMD_MC_INVITE packet

After the CMD_MC_INVITE packet and extra data is constructed, a loop is entered which retrieves the user presence information for each remote participant, obtains the remote network IP address and port and sends the invitation packet, over UDP using the audio-video manager command.

The invitation packets then go out across the wire to each remote participant. It is important to recognize that the order of receipt of the invitation packets at the remote clients is inherently unpredictable and random, since network latencies and delays may differ along different routes taken. Also, the time for processing of the initial invitation packets at each remote participant, which culminates with the transmission of either an Accept or Refuse response back to the host, is also unpredictable and depends on the human response time of each remote participant, who may or may not even be at the computer terminal when the invitation request is initially received.

4.2 Data Structures and variables

`struct UserPresence`

For each contact (buddy) this holds the network address and port, the contact status (online, offline, busy, etc), and the username and userid. This provides the primary information used for setting up a peer-to-peer session between contacts.

`struct UserPresenceArray`

Used to store a contiguous array of UserPresence structure pointers. Also provides the number (count) of elements in the array.

`Struct MULTICONF_PARTICIPANT_INFO`

For each conference participant, holds the unique participant id, the participant's conference status (connecting, connected, accepted, refused, hung up, timeout), and the associated UserPresence data for the contact.

`struct MULTICONF_DATA`

Holds the general conference data, such as unique conference id, title, and hostname. Also stores number of participants and a pointer to the detailed data for each individual participant which is dynamically allocated for each new conference. Only one conference is allowed to exist at a time.

`struct MULTICONF_GLOBAL`

The topmost structure representing the current conference. holds the overall conference status for the local client, also the size (in Bytes) of the multiconference data structure(s) including the dynamically allocated MULTICONF_ PARTICIPANT_INFO structs.

`enum enMULTICONF_STATUS`

Enumeration containing the valid statuses for a multi-conference.

`enum enPARTICIPANT_STATUS`

Enumeration containing the valid statuses for each participant in a multi-conference.

4.3 Audio-video manager

The audio-video manager is the central repository and container class for all activities related to the peer-to-peer conferencing support. It contains the implementations of the conferencing session control protocol, the network socket thread procedure which receives UDP network requests, the send command methods which provide support for sending the UDP request packets to remote peers. It contains instances of associated classes which provide the audio and video streaming and the codec implementations. It provides the conference management implementation. Fig. 7 shows the block diagram of audio-video streaming subsystem.

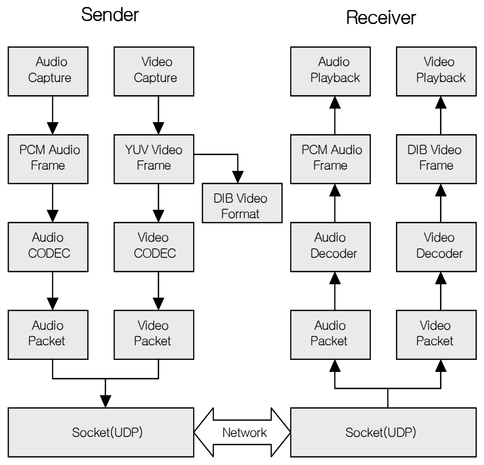


Fig. 7 The block diagram of audio-video streaming subsystem

When a new conference request is created by the host using the user list dialog, the request to start a new conference is sent to audio-video manager. When a remote user receives any inbound network packet over UDP, it is received in socket class of to audio-video manager and indicated up to command class of to audio-video manager if it is a session control protocol command packet. When the status of a remote conference participant changes, such as when they accept, refuse, or hang up a call, the status is updated using update status class of audio-video manager. When the local participant wants to terminate a remote participant's conference channel structure, or hang up the conference and disconnect completely, this is done using end talk class of audio-video manager. These are just a few of the primary roles and responsibilities of the audio-video manager component.

On the conference host system, whenever a remote participant's accept command response is received, the local system will first check that the request is not coming from a participant who already exists in a channel. If it is new, a new entry in the channel structure of audio-video manager is created and initialized, which is where all current connection channels with remote participants are maintained.

A mechanism for automatic resending of network requests is implemented, which along with the application-layer packet Acknowledgement protocol convention, provides the application-layer equivalent of the TCP automatic retransmission and acknowledgement support, at a hopefully lower overhead. Whenever a command packet is sent out, the request is inserted into a Command Queue data structure, which has an associated timer event. Fig 8 shows the flow chart of video conference system.

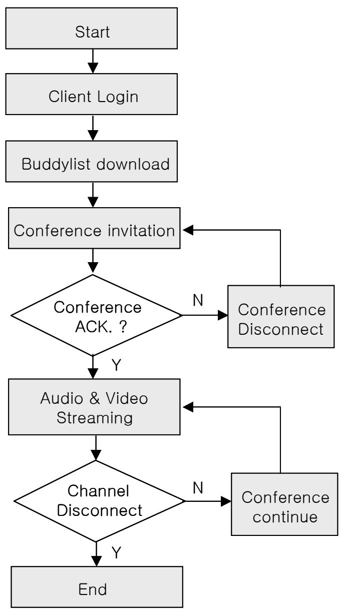


Fig. 8 Flow chart of video conference system

4.4 User interface architecture

At some point in the system design the decision was made to define a user interface architecture which is similar to that provided by the Microsoft Windows Foundation Class (MFC) model, but does not actually use MFC and rather defines a proprietary model. This is called the WndX XSystem and is implemented primarily in the WndX.h and WndX.cpp source files.

In this model, all user interface window and component control classes derive from a hierarchy of classes, the base of which is the CWndX class. Such base classes can be recognized because they use the naming convention CClassnameX, such as CDialogX, CCtrlX, CMenuX, etc. Other UI classes then derive from these base classes, which provide common inherited behaviors for a general type, such as CUserListDlg, which derives from CDialogX, as do all other dialog classes using the CClassnameDlg naming convention.

The WndX system also provides a window management system. All the classes derived from CWndX will be inserted into a window object hash table, which is maintained by the window manager class CWndMgr. Be sure to familiarize yourself with this architecture before creating new UI classes or modifying existing ones!

4.5 User interface components

User List Dialog
ID: IDD_USER_LIST
Associated Class(es): CUserListDlg
Roles & Responsibilities:

- display contact (buddy) list and statuses
- launch Sign In dialog box
- select contacts for call/conferencing and launch new call/conference
- Sign Out, Exit application

Sign In Dialog
ID: IDD_SIGN_IN
Associated Class(es): CSignInDlg
Roles & Responsibilities:

- enter UserName, Password, and MCU Server IP address
- initiate sign in request to MCU server

Main Window (Local Video)
ID: IDD_MAIN_DLG
Associated Class(es): CMainWnd
Roles & Responsibilities:

- display local video
- launch Text Chat dialog window
- launch Setup dialog
- hang up an existing call/conference
- change local audio/video quality settings

Remote Video Window(s)
ID: (dynamically created in CVWin by RegisterClassEx, CreateWindowEx)
Associated Class(es): CVWin
Roles & Responsibilities:

- for each connected (accepted) remote user in a conference, one instance of this window class will be created, and will display the remote video stream.

Multi-conference Status Dialog
ID: IDD_MULTICONF
Associated Class(es): CMultiUserConfDlg
Roles & Responsibilities:

- shows a list of each participant username & participant status, in a two-column format
- the status is updated (by CAVIOMgr:: UpdateMcParticipantStatus and CmultiConfStatus Dlg::UpdateStatus) as new session control protocol packets are received and processed by CAVIOMgr::OnCommand from the SockThreadProc network sockets thread.

Text Chat Window
ID: IDD_TXT_DLG
Associated Class(es): CMsgTalkDlg

Roles & Responsibilities:

- provides basic support for sending and receiving text-based messages. Text messages are sent by default to all participants who have accept the conference invitation. All received text messages are displayed in the same main text window of this dialog, and are identified by the sender (source).

Ring Dialog
ID: IDD_MSGBOX
Associated Class(es): CRingDlg
Roles & Responsibilities:

- a modal pop-up messagebox which is created when an incoming conference invitation is received by CAVIOMgr::OnCommand of type CMD_MC_INVITE.
- displays the remote inviter (host) IP address and allows the local user to Accept or Refuse the conference invitation. Note that based on the user's choice, all additional (secondary) invitations corresponding to this conference ID will be automatically (silently) accepted by the VICQ client.

Setup Dialog
ID: IDD_SETUP
Associated Class(es): CSetupDlg
Roles & Responsibilities:

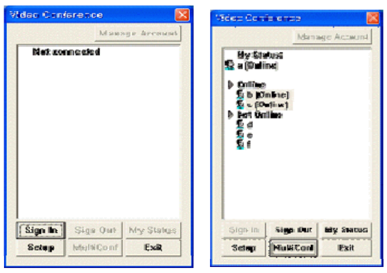
- displays a dialog window which allows user to specify setup preferences for whether a calling sound should play when an incoming call request received customize the calling sound from a list of sound choices enable automatic pop-up of the text chat window when new text messages are received.

5. Implementation and Performance Evaluation

5.1 Implementation

At some point in the system design the decision was made to define a user interface architecture which is similar to that provided by the Microsoft Windows Foundation Class (MFC) model, but does not actually use MFC and rather defines a proprietary model.

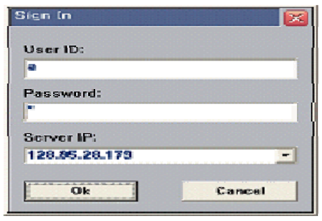
We implemented the MCU and the RTVC client for the proposed system on Windows XP. The implemented RTVC client consisted of about 30,000 lines of the source list that written in Visual C++.NET. After implementing, we tested the desktop video conference system using communication protocol and application software through Ethernet networks. In this paper, we implemented the desktop video conference system based on client-server and peer-to-peer. Fig. 9 shows contact list between RTVC clients for multi-party video conference system. Where, (a) shows initial contact list before connection and (b) shows buddy-list of local and remote client after connection.



(a) (b)

Fig. 9 Contact list between RTVC client for multi-party video conference system: (a) initial contact list and (b) buddy list of local and remote client.

In Fig. 9-(b), client a, b and c connected by online each other. Fig. 10 shows sign in dialog box of a client.


Sign In

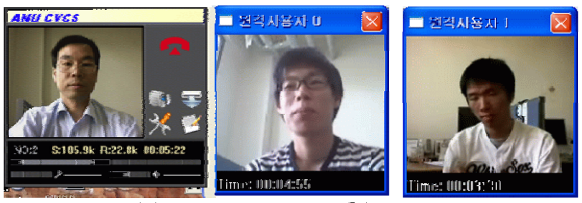
User ID: a

Password: *

Server IP: 128.85.28.179

Fig. 10 Sign in dialog box of a client

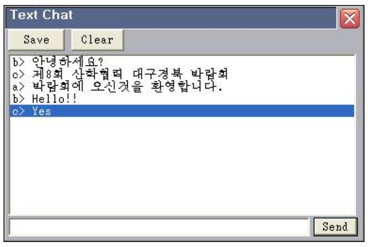
Fig. 11 shows user interface and screen capture for participant display in the client. (a) shows local video of client a and (b) shows remote video of client b in the desktop video conference system.



(a) (b)

Fig. 11 User interface and screen capture for participant display in the client: (a) local video of client a and (b) remote video of client b.

Fig. 12 shows example of text messaging between client a and client b.



```
b> 안녕하세요?
c> 제8회 산학협력 대구경북 박람회
a> 박람회에 오신것을 환영합니다.
b> Hello!!
c> Yes
```

Fig. 12 Example of text messaging between client a and client b. Fig 13 shows status information of multi-party video conference participant

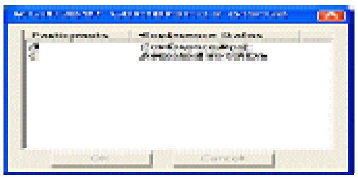


Fig. 13 Status information of multi-party video conference participant

Fig. 14 shows WMV user interface to control of video quality in the video conference.

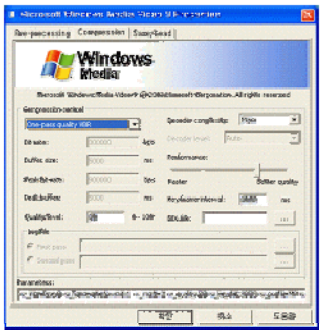


Fig. 14 WMV user interface

Fig. 15 shows total user interface screen of video conference system.

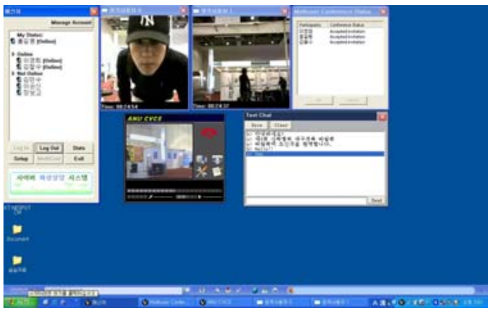


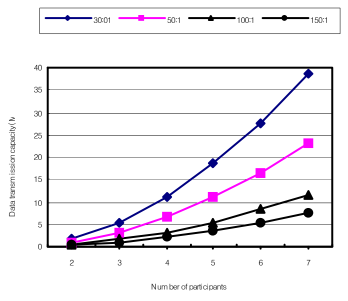
Fig. 15 Total user interface screen of video conference system

5.2 Experiment and Performance Evaluation

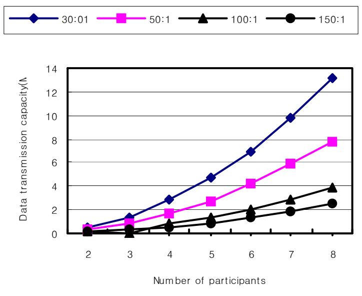
This experiment is used H.263 compression and decompression standard for multi-party video conference, the operating system is used windows XP[12-16]. The test environment for video conference is as follows.

- Number of participant for video conference : N
- Size of video frame : 320 x 240 x 3Byte(QCIF), 160 x 120 x 3Byte(QCIF)
- Compression ratio of video data : 30:1 (H.263)
- Compression ratio of audio data : 20:1 (G.723.1)
- Number of transmission frame per second : 15 frames/sec
- Sampling rate of audio : 8 KHz PCM, 8 bit Mono
- Transmission speed of network : 100 Mbps (Ethernet)

The Fig. 16 shows data transmission capacity according to video compression ratio and number of participant in case of CIF and QCIF in video conference system. In case of CIF for video frame size, the video compression ratio of 30:1 can perform video conference until 3 participants. But the video compression ratio of 150:1 can perform video conference until 7 participants. Also, In case of QCIF for video frame size, the video compression ratio of 30:1 can perform video conference until 7 participants and the video compression ratio of 150:1 can perform video conference until 15 participants. So, the number of possible participant in video conference decide according to compression ratio of video data.



(a)



(b)

Fig. 16 The data transmission capacity according to video compression ratio and number of participant: (a) CIF and (b) QCIF

Fig. 17 shows data transmission capacity according to number of participants of video conference

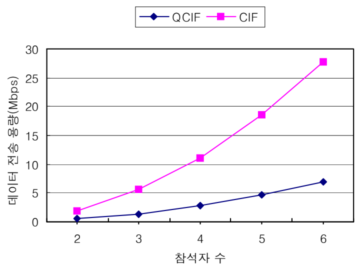


Fig. 17 Data transmission capacity according to number of participants of video conference

The proposed multi-party video conference system using hybrid peer-to-peer decreases the traffic of server, and can cuts down the load of a network.

6. Conclusion

The real-time video conference system is important and useful in a remote discussion with multiple users. In this paper, we describe implementation of real-time video conference system using high speed multimedia communication. Also, we implemented the real-time video conference system and tested the system over the high speed networks through a multi-party video conference. In case of CIF for video frame size, the video conference can perform from 3 participants to 7 participants according to the video compression ratio(30:1~150:1). Also, In case of QCIF for video frame size, the video conference can perform from 7 participants to 15 participants according to the video compression ratio(30:1~150:1). In the future, we plan to apply the embedded devices such as mobile phone or PDA in Windows CE environment.

References

[1] C. Perey and Matthew Feldman, "*Videoconferencing over IP Networks,*" in Broadband Networking, J. Trulove, ed., pp.193-210, CRC Press, 2000.

[2] A Kantarci and T. Tunali, *A Video Streaming Application on the Internet*, ADVIS 2000, LNCS 1909, pp.275-284, Springer-Verlag Berlin Hedelberg, 2000.

[3] Jenq-Neng Hwang, "*Constrained optimization for audio-to-video conversion*", IEEE Transactions on Signal Processing, 52(6):1783-1790, June 2004.

[4] Hyen Ki Kim, "*A Study on the Multimedia Input/Output Model for Web based Video Conference,*" EALPIIT'03, 2003.

[5] M. J. van Sinderen and L. Nieuwenhuis, *Protocols for Multimedia Systems,* Springer-Verlag Berlin Hedelberg 2001.

[6] A Kantarci and T. Tunali, "*A Video Streaming Application on the Internet,*" ADVIS 2000, LNCS 1909, pp.275-284, Springer-Verlag Berlin Hedelberg, 2000.

[7] Masayuki Arai etc, "*Experiment for High-Assurance Video Conference System over the Internet***,**" Proc. of the 7th IEEE International Symposium on HASE'02, 2002.

[8] S. Itaya, T. Enokido and M. Takizawa, "*A High-performance Multimedia Streaming Model on Multi-source Streaming Approach in Peer-to-Peer Networks,*" Proc. of conference on AINA'05, 2005.

[9] Li-wei He and Zhengyou Zhang, "*Real-time whiteboard capture and processing using a*

video camera for teleconferencing", IEEE July 2005.
[10] http://www.napster.com/what_is_nap_ster.html
[11] A. Hudaib and K. Kaabneh, "Hybrid model for people counting in a video stream", 12th WSEAS international conference on COMPUTER, 2008.
[12] http://www.javvin.com/protocolH323.html
[13] A. Takeda and D. Chakraborty etc, "A new scalable distributed authentication for P2P network and its performance evaluation", 12th WSEAS international conference on COMPUTER, 2008.
[14] http://www.vocal.com/data_sheets/g723d1.html
[15] S. HA and J. JIN etc, "Hybrid machine learning to improve predictive performance", ACC'08, 2008.
[16] T. Klobucar, "Evaluation of personalized search for learning resources", Proceeding of the 6th WSEAS international conference on distance learning and Web engineering, 2006.