

Enabling Video Conferencing between VIC and NetMeeting

Yuan Ji Hu Wang Zongming Fei
Department of Computer Science
University of Kentucky, Lexington, KY 40506

Abstract—Real-time multimedia conferencing tools over the Internet have become attractive in the past few years. Many tools, such as VIC and Windows NetMeeting, have been designed to hold Internet conferences. However, these two video conferencing tools cannot communicate with each other. In this paper, we design and implement a tool which is based on VIC but can set up conferences with Windows NetMeeting.

Keywords: video conference, multimedia, RTP, H.323

I. INTRODUCTION

A. Windows NetMeeting and H.323

Windows NetMeeting is one of the most popular video conferencing tools. It uses H.323 protocol suite to realize the Internet conferencing over packet-based networks. Such H.323 conferencing tools also include Intel VideoPhone and Netscape Conference.

H.323 [1] is a collection of International Telecommunication Union (ITU) recommendations. It is based on Internet Engineering Task Force (IETF) Real-Time Protocol (RTP) and Real-Time Control Protocol (RTCP), with additional protocols for call signaling, audio and video communication, and data transmission. Specifically, H.255.0 call control [2] defines how a user establishes a connection with a remote user. H.255.0 Registration, Admission, Status (RAS) performs registration, admission control, bandwidth changes, and status between endpoints and gatekeepers. H.245 [3] controls the exchange of end-to-end control messages such as capabilities of endpoints. Audio and video codecs are used to define how audio and video signals are encoded or decoded for communication between conferencing parties. Moreover, H.323 specifies T.120 services for data communication and conferencing within and next to an H.323 session. Most importantly, T.120 supports means that data handling can occur either separately or in conjunction with H.323 audio and video [4].

The architecture of H.323 is shown in Fig. 1 [4]. This figure also shows interfaces for the network, audio and video equipment.

For optimal performance over the Internet, NetMeeting specifies H.263 and G.723 as default codecs. NetMeeting can also negotiate with other codecs, such as H.261 or G.711, depending on the requirements of other H.323-compatible products. Thus, NetMeeting is able to easily interoperate with other H.323 standards-based products.

To set up a call, NetMeeting makes a TCP connection through the H.323 standard signaling port, port 1720, and sends a call setup message to a remote user. After

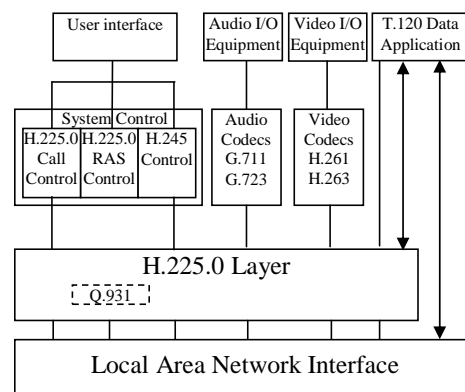


Fig. 1. H.323 Architecture

the remote user replies with a call connection message, NetMeeting reads the TCP port number from the message and make another TCP connection through this port. By using this connection, NetMeeting checks the remote user's audio and video capabilities and informs its capabilities to the remote user. NetMeeting will choose the capabilities owned by both sides to compress audio and video data. After checking the capabilities, NetMeeting and the remote side negotiate a UDP port and start a RTP session to transfer audio and video signals.

Since Microsoft and more than 120 other leading companies support and have implemented H.323 in their product [4], H.323 is becoming a standard for audio and video conferencing over the Internet.

B. VIC

VIC, on the other hand, is another real-time multimedia application for video conferencing. VIC is designed by Lawrence Berkeley National Laboratory. It is a flexible and extensible architecture to support heterogeneous environments and configurations. For example, in high bandwidth settings, multi-megabit, full-motion JPEG streams can be sent using hardware assisted compression; while over lower bandwidth environment, aggressive low bit-rate coding can be carried out in software [5].

VIC uses RTP for transmitting video signals. Although VIC supports point-to-point conferencing using standard unicast IP addresses, it is primarily intended as a multi-party conferencing application. Features unique to VIC include an Intra-H.261 video encoder and voice-switched viewing windows. Compared with H.261, intra-H.261 uses only a subset of H.261 specification, thus the complexity of rendering images is reduced. Voice-switched windows allow viewing windows to be configured to

“follow the speaker”. In this manner, more than one window can be configured to display the most recent speakers at the same time [5].

These features of VIC make it a widely used conferencing tool. For example, Argonne National Laboratory chooses VIC for its world-wide group communication project - Access Grid.

An overall structure of VIC is given in Fig. 2 [6]. First, a data stream is received from networks. Then, a demuxer divides the data stream into different data streams which will be sent to a software decoder or a hardware decoder. After the data streams are decoded into video signals, these video signals will be rendered in a window or an external video output. Usually, the video signals will be played in a window.

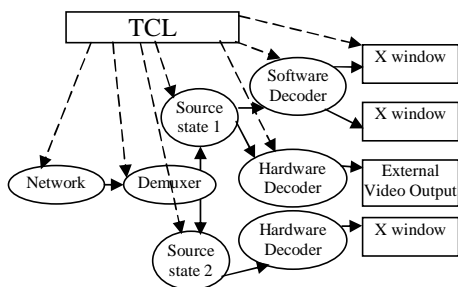


Fig. 2. Structure of VIC

Note that VIC only supports the video portion of multimedia conferences. Audio, whiteboard, and session control tools are implemented as separate applications.

C. Incompatibility

Unfortunately, NetMeeting and VIC cannot establish connections with each other. The reason is that they use different methods to initialize conferences. NetMeeting negotiates a UDP port with a remote party and then starts a RTP session using this port. In contrast, before VIC sends RTP packets, neither does VIC send messages nor negotiates a UDP port with a remote user. Two VIC parties have to know the UDP port before a session begins. VIC can always send RTP packets no matter whether the remote party exists or not.

The rest of this paper is organized as follows. We present our design model in section II and describe implementation details in section III. Related work is discussed in section IV. Finally, section V concludes this paper.

II. DESIGN

To enable video conferencing between the NetMeeting and VIC, we have two design choices: either including the signaling functionalities of H.323 into VIC or allowing NetMeeting to accept VIC connections.

The first choice is considered better since VIC is an open source tool and any modifications are permitted. Furthermore, we believe that our strategies would also allow VIC to easily interoperate with other H.323 products.

Three functionalities should be included in VIC. First is H.323 capabilities. VIC should be able to send call setup messages, read call connection messages, and check the capabilities of a remote party. In other words, VIC has to understand H.323 protocols. So, our new model is as following. When a signaling message comes through a port other than the H.323 standard signaling port, 1720, VIC performs as usual. When a message comes through port 1720, VIC should be able to establish an H.323 session with the remote party. Fig. 3 gives the design model of the new VIC. Besides the H.323 capabilities, VIC also needs audio capabilities. Original VIC can only send, receive and render video signals. Therefore, the transmission and rendering of audio signals should be included. Finally, an interface is necessary for users to start a call and to quit the program.

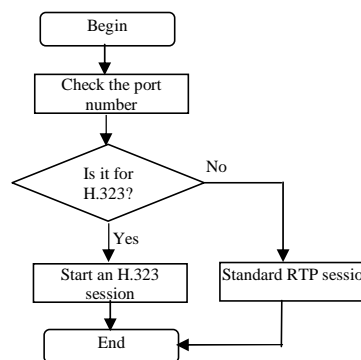


Fig. 3. Structure of the new VIC

Some procedures need to be done to establish an H.323 session. Before trying to create one, VIC initially sets its audio and video capabilities. Then it checks whether audio and video devices exist or not. VIC removes all audio capabilities if no audio device exists. At the same time, it sets a fake device as the video device and keeps the video capabilities if there is no video device. Next, VIC tries to set up an H.323 session. If the H.323 session is set up successfully and at least one of the audio capabilities exists, VIC opens an audio channel. If the H.323 session can not be set up, VIC terminates. After setting up the audio device, VIC uses the video or the fake video device to capture video signals. The received video signals are rendered in X Windows. The fake video device is only used when no video device exists. Finally, VIC terminates after the H.323 session ends.

III. IMPLEMENTATION

Whenever we modify VIC, we keep its original functions and capabilities unchanged. For example, the new VIC is started by the same method as original VIC. Thus, its conferencing capabilities via an RTP session remain the same. We only embed the H.323 functions into VIC.

VIC is originally implemented by a low version Tcl/Tk. Instead of using Tcl/Tk, we choose GTK+ to implement user interfaces on the Linux platform.

A. Processes and threads

The function of starting an H.323 session is implemented by two processes and three threads. The two processes are GTK+ user interface process and H.323 process. The GTK+ user interface process is used to display user interfaces. The H.323 process controls an H.323 connection. Specifically, it initializes and terminates a connection. The H.323 process contains three threads, namely, main thread, call thread, and user thread, to handle H.323 events. These threads must be synchronized since they operate on the same resources. In our model, the synchronization requires two semaphores.

A main thread first initializes some parameters and starts a user thread. Next, the main thread waits for a termination signal from the user thread. Once the signal is received, the main thread terminates. The above operations are shown in Fig. 4.

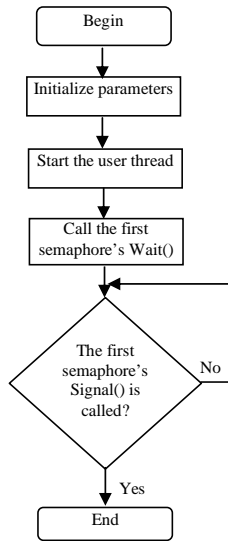


Fig. 4. Main thread

A user thread starts a call thread and waits for a termination signal from the call thread. The user thread keeps waiting until the signal is received. Once the signal is received and the user thread also receives a proceeding signal from the call thread, the user thread establishes an H.323 session. Otherwise, the user thread does not do anything. Finally, it sends a termination signal to the main thread and terminates. Fig. 5 illustrates the above operations.

A call thread determines if a proceeding signal should be sent. If the proceeding signal is sent, an H.323 connection will be established by the user thread. On the contrary, if the proceeding signal is not sent, the call thread only sends a signal to terminate the user thread. An H.323 session will not be established without a proceeding signal from the call thread. Fig. 6 shows the details of a call thread.

A main thread is in charge of the course of establishing an H.323 connection. In the main thread, G.711 (audio codec) and H.261 (video codec) are always added as

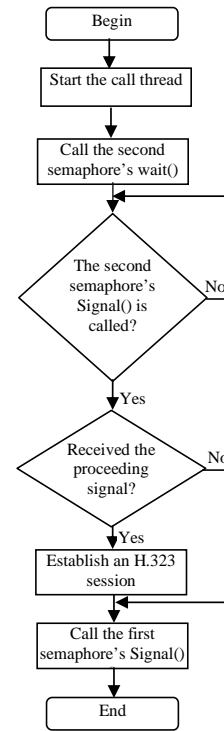


Fig. 5. User thread

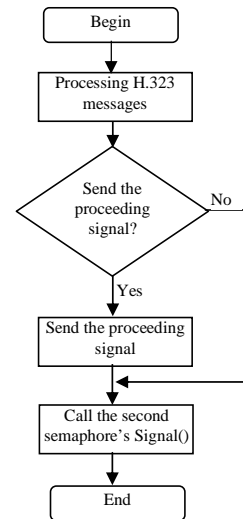


Fig. 6. Call thread

basic capabilities. In most cases, sound card works fine on Linux, so G.711 usually remains.

B. Classes in our implementation

The main data structure in our implementation is classes. The following are some of the classes implemented.

First is VicProcess class. It only has a constructor, a destructor, and a member method. When an instance of this class is created, a main thread will be created.

Second is VicEndPoint class. It inherits methods from the H323EndPoint class which is defined in OpenH323

library. However, some inherited methods are not enough to accomplish the H.323 related tasks. We revise six methods, and their functionalities are described below. `OpenAudioChannel` actually opens an audio channel and handles errors. It reports an error message if no sound device exists. `OpenVideoChannel` opens a video channel. If there is no video device or the video device is occupied by other programs, it not only reports an error message, but also replaces the video device with a fake video device. Moreover, this method sets video frame size, video grabber port, and determines video format. `CreateConnection` creates an H.323 connection and returns the connection. `OnIncomingCall` records some data about a call. These data include calling time, local user name, and remote user name. `OnConnectionEstablished` changes the status of the call to notify the user that the connection has been set up. Similarly, `OnConnectionCleared` notifies the user when the connection has been torn down.

Third is `VicConnection` class, which is derived from `H323Connection` defined in `OpenH323` library. Two important inherited methods are involved. One is `OnStartLogicalChannel` and the other is `OnCloseLogicalChannel`. They inform a user that a logical channel for audio and video has been opened or closed, respectively.

The fourth class is `XwinVideoDevice`, which is derived from `H323VideoDevice` defined in `OpenH323` library. An method in this class, `SetFramSize`, sets frame resolution for rendering images and opens a window to display signals.

In addition to above classes, we use a method, `create_VicMW`, to create the user interface of an H.323 session. Two other methods, `on_QuitButton_clicked`, and `on_ContinueButton_clicked`, are used to define the operations when quit or continue button is clicked on the user interface.

C. Modification to the original VIC code

In the original VIC code, there is a scanner class, namely, `video4linux`. This class is initialized outside of the main thread of the H.323 process so that the grabber will be occupied by this scanner class before the main thread starts. As a result, the H.323 part can not get video devices. To solve this problem, we need to have this scanner class initialized within the main function. We revise `video4linux.cpp` by setting the scanner class as an extern variable so that the scanner class can be "seen" within the main thread. In our implementation, we move the code for creating an instance of the `video4linux` scanner into the main thread. This makes the scanner class initialized inside of the main thread. Therefore, we can control when the scanner class should be initialized.

IV. RELATED WORK

Several other studies have been conducted to solve the compatibility among different Voice-Over-Internet (VoIP) protocols. These solutions fall into three categories [9], Time Division Multiplexing (TDM), Single Protocol Architecture, and Protocol Translation.

TDM approaches introduce latency in real networks. The single protocol architecture approach is also not preferable since these different protocols will coexist together in a foreseeable future. For instance, on one hand, H.323 products are used mostly in PC-based conferencing. On the other hand, SIP products are popular in Internet telephony. Therefore, the protocol translation approach is getting more attention. Ho et al. has designed and implemented a gateway to support the interoperability between Session Initiation Protocol (SIP) and H.323 [7]. A Generic Conference Control Gateway (GCCG) is used to translate messages and media streams between endpoints. Similarly, a signaling gateway (SGW) is proposed in [8]. SGW servers as a interpreter between SIP and H.323. It translates messages of call setup, user registration, and session description between SIP and H.323 clients. Furthermore, Kumar et al. proposes a multi-signaling protocol architecture to handle the interoperability [10]. Without using a central gateway, an endpoint management architecture is proposed in which each endpoint is equipped the capabilities to translate different protocols such as SIP, H.323, and Media Gateway Control Protocol (MGCP). Compared to them, our work focuses on the implementation that enables the interoperability of two Internet conferencing tools: VIC and NetMeeting.

V. CONCLUSION

In this paper, we have described a mechanism to enable VIC communicate with NetMeeting. A detailed design is also given. Our implementation has been tested in a practical, real world environment.

VI. ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation under Grants CCR-0204304 and EIA-0101242, and a grant from the Kentucky Science and Engineering Foundation as per Grant Agreement #KSEF-148-502-05-139 with the Kentucky Science and Technology Corporation.

REFERENCES

- [1] ITU-T Recommendation H.323, "Packet-base Multimedia Communication Systems".
- [2] ITU-T Recommendation H.225.0, "Call Signaling Protocols and Media Stream Packetization for Packet-based Multimedia Communication Systems".
- [3] ITU-T Recommendation H.245, "Control Protocols for Multimedia Communication".
- [4] Windows NetMeeting Resource Kit, Chapter 11: Understanding the H.323 Standard. <http://www.microsoft.com/windows/NetMeeting/>
- [5] VIC - Video Conferencing tool: <http://www-nrg.ee.lbl.gov/vic/>
- [6] Steven McCanne et al., "Vic: A flexible frame work for packet video", ACM Multimedia, San Francisco, CA, 1995.
- [7] Jiann-Min Ho et al., "A conference gateway supporting interoperability between SIP and H.323", ACM Multimedia, Ottawa, Canada, 2001.
- [8] K. Singh, H. Schulzrinne, "Interworking between SIP/SDP and H.323". In proceedings of the 1st IP-Telephony Workshop, Berlin, Germany, 2000.
- [9] Cisco Systems Inc. White Paper, "Understanding Voice Packet Protocols".
- [10] Anoop Kumar K. et al., "A multi-Signaling protocol architecture for voice over IP terminal", In proceedings of IEEE INFOCOM, Hong Kong, 2004.