PEER-TO-PEER MULTIPOINT VIDEOCONFERENCING

M. Reha Civanlar, Öznur Özkasap, Tahir Çelebi

Department of Computer Engineering
Koç University, Istanbul, Turkey
{rcivanlar, oozkasap, tcelebi}@ ku.edu.tr

ABSTRACT

A peer-to-peer architecture for multipoint video conferencing is presented. The architecture targets end points with symmetrical, low bandwidth connections. Without creating any additional demand on the networking and computing resources needed for a point-to-point video conference, this architecture can extend it into a multipoint one. A protocol for a completely distributed implementation has been developed and tested on a prototype system extending a point-to-point video phone to a multipoint one.

1. INTRODUCTION

Mostly due to be the large bandwidth demand of video transport and associated high costs, the use of videoconferencing in general and multipoint (MP) videoconferencing in particular are still not as popular as MP audio conferencing. Problems with implementing MP video can be observed, for example, in the recent instant messaging (IM) applications. Even though several IM applications allow participants of multi-user chat rooms to establish pair-wise video communications, each added party increases both the send and the receive bandwidth requirements for all the end points involved. Over a low bandwidth connection e.g., an ordinary modem connection over a phone line, the bandwidth is barely enough for video communications with a single point. For such connections, which still constitute a significant percentage of global access lines to the Internet, a MP video system implementation that increases the bandwidth demand with the number of participants is definitely not realistic.

One approach to reduce the bandwidth demand of a MP video system is to use additional network based equipment called multi point control units (MCU) [1]. An MCU must have larger access bandwidth to the network than a regular end-point and it acts as a single-point recipient for each participant of a multipoint conference. An MCU prepares a multipoint video representation e.g., a collage of images, that can fit into the bandwidth of a single video source and sends this to each participant. However, the cost and the operation complexity of MCUs constrain their uses mostly to larger business applications.

Another way to reduce the bandwidth demand of MP video at the transmission side is to use multicasting when it is supported by the underlying network. An additional advantage of a multicasting-based solution is the reduced operation complexity due to its distributed nature [2]. Unfortunately, due to the fact that wide-scale deployment of native mode multicasting on the global Internet has still not realized, this option is not a viable choice for many people either.

In this paper, we present an alternative approach to MP video conferencing based on the use of a distributed peer-to-peer (P2P) system. This system has no need for additional hardware, as in MCUs, or network infrastructure support such as multicast. With almost no additional demand on the networking and computing resources needed for a point-to-point video conference, our system can make it possible to extend a point-to-point conference into a MP video conference, where each participant can see one other selected participant under *most* practical cases.

P2P techniques are becoming extremely popular and finding diverse applications. Use of P2P techniques for video conferencing through an end system based multicast implementation has been reported before [3]. However, such systems assume availability of larger upstream bandwidths for each participant. In [4], another P2P solution for MP video conferencing based on a centralized architecture is presented. Our implementation, by focusing on the needs of those users with symmetric, low bandwidth connections, presents a totally distributed solution with no single point of failure and central server maintenance.

We built a prototype of our system by extending an H.263 based, point-to-point video telephony implementation to MP. The resulting system can be used for MP video conferencing among participants with modem connections to the Internet without requiring multicast support or MCUs and it can be used for extending most other point-to-point systems also.

In the next section, our P2P approach, its benefits and limitations are discussed. Section 3 discusses the details of our implementation. Last section presents our conclusions and discussions.

0-7803-8554-3/04/$20.00 ©2004 IEEE.

2. P2P APPROACH

In our P2P approach to MP videoconferencing, we assumed that each participant can produce, send, and receive only one video signal at any given time. This way the networking and the computing resources needed for a MP conference stay the same as that of a point-to-point conference. In order to enable MP video; however, a participant receiving a video signal may have to pass it to another participant who also wants to receive the same video signal. Relaying the received video signal to another participant does not bring any additional computational burden beyond sending the participant's own video signal, but it uses up the entire available upstream bandwidth. Thus, a participant who is sending its own video signal can't act as an intermediate peer passing a video signal to another peer, and, an intermediate peer can't send its own video signal to anyone else.

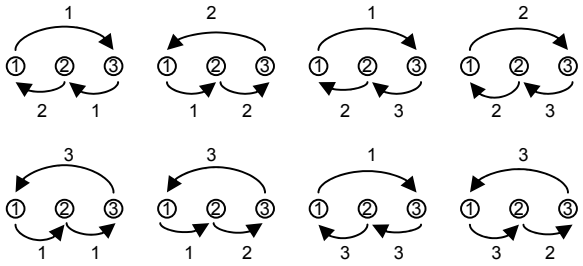

Figure 1 - In a P2P MP videoconference with three participants each participant can see any other participant. (*The numbers on the arrows indicates which participant's video signal is being transmitted over the corresponding link.)*

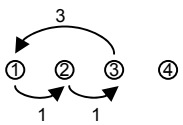

Figure 2 - Participant number 4 can't see participant number 2 in this configuration.

As shown in Fig. 1, in a MP videoconference among three participants, a P2P configuration can always be found such that any participant can see any other participant at any given time. As demonstrated in Fig. 2 for a conference with four participants; however, there are cases where this will not be possible when the number of participants is four or larger. Fig. 4 shows theoretical and simulation results for the number of unachievable configurations, i.e., configurations for which one or more participants can't receive video from another participant of their choice (as a percentage of all configurations) as a function of the number of participants. The configurations for obtaining the theoretical values are determined by generating all possible requests for the given number of users and enumerating the solutions using the algorithm explained in the next section. As the number of participants increases, P2P approach may loose its effectiveness. On the other hand, a conference with a very large number of participants will usually be in the form of a presentation, where most participants want to see the presenter. In such cases, which include remote lectures, the only constraint on the number of participants that can be served with our approach is the delay introduced by each additional pass-through operation.

Another problem caused by the increased delay due to the pass through operations is that extending P2P MP video conferencing to interactive audio conferencing may not be feasible. But, the approach is perfectly suitable for adding MP video to other interactive applications such as instant messaging.

2.1 A Distributed Solution

Our distributed architecture can be used to implement a P2P MP videoconferencing system that can configure itself to allow each participant to see any other participant whenever possible i.e., when the resulting configuration is achievable. In this architecture, we assumed that the usual tasks of setting and controlling a multipoint conference are handled by the appropriate protocols as described in the Internet multimedia architecture [5]. Therefore, tasks such as inviting participants to conferences, distributing and managing the list of active participants, etc., are outside the scope of this description. Also, transmission of any other media associated with the video such as text messages is not addressed but, can be carried out in a similar manner.

The state of our P2P MP Videoconferencing system is defined based on an object called a "***chain***." A chain defines the video flow configuration between the participants of a conference receiving the same video signal at a given time. All the chains involved in a P2P MP videoconference define the conference's current state. The video flow configurations, i.e. chains, change during a session as a response to "***configuration messages"*** sent between the participants.

2.1.1. Chains

A chain is an ordered list of identification numbers of the conference participants who receive the same video signal, in the order they receive it. The first entry in this list, called the "**head**," is the video source for the rest of the participants in the list. Every participant gets assigned a unique identification (ID) number when they join the conference. Each chain has an additional binary valued attribute called "*extensibility*". A chain is extensible (**e**) if its last member is not a head for another chain, non-extensible (**ē**) otherwise.

Each chain member knows the head of the chain and the receiver ID if it relays video. The head knows about the entire chain. A participant who does not send or receive video is a member of an extendible chain containing only its own ID number. Fig.3 shows the chains corresponding to the conferencing arrangements shown in Figs. 1 and 2.

{ <1,3,2,ē> , <2,1, ē > } { <1,2, ē > , <2,3,1, ē > }
{<1,3, ē > , <3,2, ē > , <2,1, ē >} {<2,1,3, ē > , <3,2, ē >}
{<1,2,3, ē >, <3,1, ē >} {<1,2, ē >,<2,3, ē >,<3,1, ē >}
{<1,3, ē >,<3,2,1, ē >} {<3,1,2, ē >,<2,3, ē >}
4. a

{<1,2,3, ē >, <3,1, ē >, <4,e>}
4. b

Figure 3 – Chains corresponding to the configurations in Figures 1 (a) and 2 (b)

2.1.2 Configuration Messages

The **heads** play a central management role for their chains. For example, if a new participant wants to receive video, the head needs to configure its chain by sending the necessary messages to other affected participants. If a chain member looses video, it waits for a "*modify chain*" message from the head and if such a message does not arrive within a timeout period, the head is assumed to be non-functional and the chain is dissolved.

There are 8 configuration messages as outlined below:

i. Video Request Message: Sent by any participant to request video from another participant. The recipient may already be the head of a chain, in which case, the chain needs to be reconfigured if possible. Otherwise, a new chain can be created. A video request message carries the ID of the requestor and an attribute indicating whether the requestor is sourcing video to any other participant or not. A participant already receiving video can't send a video request message unless it releases its current video successfully. Various actions resulting from a video request message are outlined in the pseudo code below.

```
Part. k receives a video request message from part. m
if k is a head
    if k's chain is extensible
        append m to chain
    else
        if m can pass through video
                insert m in chain
        else
                deny the request
else
        if k is a chain member
                if the chain is extensible
                        move k to the end
                        start new chain
                else    deny the request
        else    start new chain
```

Since the extensibility of a chain may change, the head checks this with the last member before adding a new member or reconfiguring the chain.

ii. Video Request Ack/Nack Message: This message is sent as a response to a video request message. If the requestor has to pass the video that it receives to another participant, the ID of this participant is included in the acknowledgement. If the video can't be sent at the current configuration, a *nack* message is sent back.

iii. Release Request Message: When a participant decides to stop receiving, it must send a "*release request*" message to the head. If the "*release request*" message is not from the last chain member, the head reconfigures its chain by sending a "*modify chain*" message to the member sending video to the source of the "*release request*" message. This message includes the ID of the successor if any. If the last participant wants to stop video, the peer before the last is sent a "*modify chain*" message with a zero argument.

iv. Extensibility Check Message: A head sends this message to the last member of the chain to determine if the chain is extensible.

v. Extensibility Ack/Nack Message: Sent as a response to an "*extensibility check*" message.

vi. Modify Chain Message: Sent by the head to only one participant at a time to change its video receiver. This message includes the new recipient's ID.

vii. Keep Alive: Used to determine if a participant is still in the session. This is among the duties of the conference management protocol; however, getting this information from conference management may take too long, resulting in a prolonged loss of video for several members. A participant starts sending "keep alive" messages to its recipient as soon as it joins a chain. The last member sends its "*keep alive*" message to the head. If a participant does not receive a "*keep alive*" message for a pre-determined time period, it notifies the head so that it can make the required changes on the chain. If a participant discovers that the head is non-functional then it notifies all other participants in the chain to dissolve the chain.

viii. Dissolve: Sent by any chain member to other chain members to dissolve a chain when the head is determined to be non-functional.

3. IMPLEMENTATION

Our P2P MP video conferencing prototype has three main modules. The first module is a conference management system. We implemented a simple conference management system ourselves that handles creating a new MP conference, and joining in or leaving an ongoing conference. It is a client/server implementation and the server can run on any one of the conference participants'

machine or on a separate system. Currently, the conference management module and the second module, the "P2P MP Videoconferencing Agent", are implemented as a single process in Java; however, these can easily be separated and the interface between them can formally be specified so that other conference management systems can be used. A logical and standards based choice for this module will be a SIP based solution.

The second module is our main contribution and it implements the P2P MP videoconferencing protocol. We implemented a simple user interface within this module also. The interface between the second module and the video codec is implemented using UDP sockets for interprocess communication.

The video codec module handles video capture, compression/decompression and transmission over the network. In our prototype, we have used a proprietary implementation of an H.263 based video codec. This system is designed to operate over a best effort network, and, hence, can gracefully handle stops in its input video and changes of the video sources during a session without a need to rapidly stop and restart the codec that may be problematic for many video codec systems. The original point-to-point video phone application is extended to handle relaying of the received compressed bitstream when needed.

All the eight P2P MP video configuration messages outlined in Section 2, except the "*keep alive*" message, are transmitted between peers over TCP. The messages are represented as ASCII strings with blank separated fields. The "*keep alive*" messages are transmitted over UDP.

4. CONCLUSIONS & DISCUSSIONS

We verified the feasibility of our architecture and the validity of our P2P protocol using our prototype system with up to seven real participants. We also run automated conference sessions to check the consistency of the system's behavior for larger number of participants and video switching activity. Simulation results obtained so far on the percentage of unachievable configurations as a function of the number of conference participants are depicted in Fig. 4. When compared to the theoretical values, our simulation results show that, as the number of participants gets larger the percentage of unachievable configurations stays constant, or increases very slowly. This is mainly due to the fact that in a dynamic conference setting with requests concurrently triggered by all participants, all possibilities of chain configurations are not equally likely to occur. As a further investigation on the system's behavior and scalability issues, we plan to extend our testbed setting to larger number of participants. The computational burden caused by the P2P application on the system is observed to be insignificant (less than 1% CPU time on a 2.4 GHz Pentium).

We are planning to extend our architecture to handle end points with more than one simultaneous input and output. Another target is to consider the "cost" of a connection in deciding the order of the participants in a chain. This will be particularly important if the system is used over long distance and international connections.

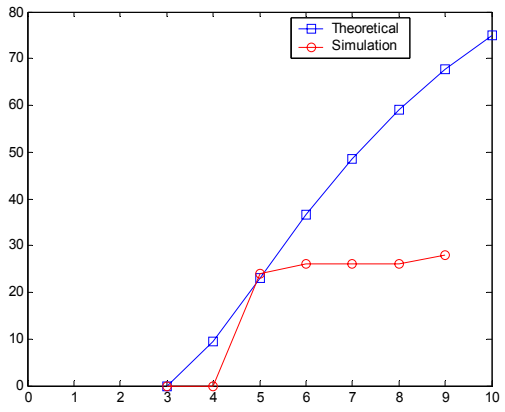

Figure 4 – Theoretical and simulation results for unachievable configurations' percentage (y) of all configurations for a given number of participants (x)

The system assumes a friendly environment and, hence, security concerns are not considered. Clearly, in a non-friendly environment, it is possible to cause unwanted results, ranging from modifying the relayed video to transmitting bitstreams that may overload the decoders, making DOS attacks possible. One way to prevent these can be enforcing authentication in joining conferences.

5. REFERENCES

1. ITU-T Study Group XV – Recommendation H.231, *Multipoint Control Units for audiovisual systems using digital channels up to 1920 kbits/s*, March 1993.
2. M. R. Civanlar, R. D. Gaglianello, G. L. Cash, *Efficient Multi-Resolution,Multi-Stream Video Systems Using Standard Codecs*, Journal of VLSI Signal Processing, 1997.
3. Yang-hua Chu, Sanjay G. Rao, Srinivasan Seshan and Hui Zhang, *Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture*, Proceedings of ACM SIGCOMM'01, San Diego, California, August 2001.
4. M. Hosseini, N. D. Georganas, *Design of a Multi-sender 3D Videoconferencing Application over an End System Multicast Protocol*, ACM, Multimedia 2003, Berkeley, CA, November 2003.
5. Mark Handley, Jon Crowcroft, Carsten Bormann and Jörg Ott, *Very large conferences on the Internet: the Internet multimedia conferencing architecture*, The International Journal of Computer and Telecommunications Networking, Vol 31, No 3, pp 191-204. Elsevier, North Holland, 1999.