Available online at www.sciencedirect.com
SCIENCE DIRECT®
Signal Processing: *Image Communication* 20 (2005) 743–754
SIGNAL PROCESSING: IMAGE COMMUNICATION
www.elsevier.com/locate/image

Peer-to-peer multipoint videoconferencing on the Internet

M. Reha Civanlar, Öznur Özkasap*, Tahir Çelebi

Department of Computer Engineering, Koc University, 34450 Istanbul, Turkey

Received 29 April 2005; accepted 2 May 2005

Abstract

A peer-to-peer architecture for multipoint videoconferencing is presented. Each conference participant may have asymmetric and dissimilar bandwidth connections to the Internet. The solution does not require additional hardware, as in multipoint control units, or network infrastructure support such as multicast. Without creating any additional demand on the networking and computing resources needed for a point-to-point videoconference, 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. The architecture of the prototype system along with the details of the protocol optimization is discussed. Several performance results are presented.

Keywords: Multipoint videoconferencing; Peer-to-peer

1. Introduction

Large bandwidth demands of video transport and associated high networking costs are among the leading reasons for the current unpopularity of videoconferencing in general and multipoint (MP) videoconferencing in particular. Problems with implementing MP videoconferencing can be observed, for example, in the video enhanced instant messaging (IM) applications. Recently, several IM implementations started to allow participants of chat rooms establish pair-wise video communications along with text-based chat. Using this technology in a multi-party video-chat, where everyone wants to see everyone else, however, increases both the send and the receive bandwidth demands for all participants for each added party to the MP conference. For low bandwidth connections, e.g., connections over ordinary modems, the bandwidth is barely enough for transporting video between two participants and clearly will not be sufficient as the number of participants increases. Consequently, users with modem connections cannot participate in MP videoconferences. Considering that low bandwidth

*Corresponding author.
E-mail addresses: rcivanlar@ku.edu.tr (M.R. Civanlar), oozkasap@ku.edu.tr (Ö. Özkasap), tcelebi@ku.edu.tr (T. Çelebi).

0923-5965/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.image.2005.05.003

connections still constitute a significant percentage of the Internet access lines, a solution to this problem is needed for the wide-scale usability of MP videoconferencing.

One approach to reduce the bandwidth demand of an MP videoconferencing system is to use network-based devices called multipoint control units (MCU) [1]. An MCU is a central device which has a larger bandwidth Internet connection than a regular participant and it receives all participants' video signals and disseminates them. An MCU prepares a video representation, e.g., collages of low-resolution video and sends it to all participants. However, their costs and management complexities make MCUs suitable only for larger business applications.

Another way to reduce the bandwidth requirement 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 [6]. Unfortunately, due to the fact that wide-scale deployment of native mode multicasting on the global Internet has still not been realized, this option is not a viable choice either.

In this paper, we present a new fully distributed peer-to-peer (P2P) approach for multipoint videoconferencing. Our initial assumption, namely 1-1 I/O constraint, dictates that a participant can only receive, send and produce a single video signal at any given time. Then, we further consider the case where all participants can receive and produce a single video signal, but send/relay one or more video signal(s) at any given time. This approach takes the heterogeneity of the Internet environment into consideration. We built a prototype of our system by extending an H.263-based, point-to-point video telephony implementation to MP video. The resulting system can be used for MP videoconferencing 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. Our system has no need for additional hardware, as in MCUs, or underlying network support like multicast. Besides, our solution is fully distributed and does not suffer from single point of failures. With almost no additional demand on the networking and computing resources needed for a point-to-point videoconference, our system can make it possible to extend a point-to-point conference to an MP videoconference, 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 videoconferencing through an end-system-based multicast implementation has been reported before [4]. However, such systems assume availability of larger upstream bandwidths for each participant. In [11] another P2P solution for MP videoconferencing 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.

The article is organized as follows. In the next section, an overview of P2P media distribution approaches is presented. Section 3 presents the main algorithm for a distributed P2P MP videoconferencing arrangement for participants with a single video input/output. Section 4 discusses the implementation details of our prototype system. Section 5 addresses optimizing the resulting system for delay. A solution for the case where some participants have multiple video outputs is discussed in Section 6. Finally, Section 7 presents results and conclusions.

2. Related work

Most of the recent techniques for P2P media streaming on the Internet utilize multicast model at the application layer. The main benefit of implementing application layer multicast is overcoming the lack of large-scale IP multicast deployment at the network layer. In addition to P2P models, there exist other application layer multicast solutions adapting an overlay-based (infrastructure level) approach. Overlay-based approaches require the existence of dedicated servers as opposed to P2P ones. They are scalable since the receivers can get the content from the dedicated servers acting as software routers, which

would decrease the bandwidth consumption at the source as well.

Most of the publications on P2P media distribution so far focus on streaming applications. Capacity analysis of a P2P media streaming system, and the two key problems of these systems, namely the media data assignment for a multi-supplier P2P streaming session and the fast amplification of the P2P streaming capacity, are discussed in [21]. Various methods have been proposed to address the asynchrony of user requests and the synchronous nature of the multicast paradigm [9]. A recent study [8] proposes a layered P2P streaming mechanism for on-demand media distribution. Asynchrony of user requests and heterogeneity of peer network bandwidth are the two main issues that this work points out. As the solution, cache-and-relay and layer-encoded streaming techniques are proposed. The solution has been shown to be efficient at utilizing peers' bandwidth, scalable at saving server bandwidth consumption, and optimal at maximizing streaming quality of peers.

ZIGZAG is a P2P system developed for single-source media streaming to a large group of participants on the Internet [20]. It proposes an efficient failure recovery and control protocol for maintaining the application layer multicast tree against network dynamics and unpredictable client behaviors. Failure recovery is performed locally, and control overhead is constant. In addition, the system addresses the issue of minimizing end-to-end delay from source to receivers. The study includes a theoretical analysis together with a simulation study and comparison with NICE protocol [2], an application layer multicast technique for P2P streaming. NICE and ZIGZAG are different in their multicast tree construction and maintenance mechanisms. They both focus on large P2P networks. The system in [20] is based on one of the first P2P streaming techniques called "chaining" [18]. It allows a source to server chain of clients using a single data stream. In chaining, the delivery tree is built rooted at the source, and it uses the source's bandwidth efficiently by utilizing other peers' bandwidth to stream media to others.

CoopNet [13] proposes a hybrid approach integrating the client-server and P2P models for both live and on-demand streaming media. For on-demand media, clients make use of caching content that they viewed recently. For live media streaming, clients form a distribution tree rooted at the source. In particular, CoopNet considers the problem of server overload by a large number of requests from clients. In such a case, it utilizes P2P model and clients cooperate to distribute content to decrease the load on the server. In addition, CoopNet uses a multiple description coding method for the media content, which improves robustness and balances load among peers.

Another study [4] explores the impact of application requirements on end system multicast architecture in which end systems in a multicast group self-organize into an overlay structure. In the context of audio and videoconferencing applications, techniques proposed are incorporated into Narada [5], which is a self-organizing and self-improving protocol that adapts to network dynamics. Narada constructs overlay structure in two phases. First, it maintains a connected graph called mesh among the peers, and then constructs a shortest path spanning tree on the mesh whenever a source wants to transmit content to a set of receivers. Each member has a limited number of neighbors on the mesh determined by that member's connection bandwidth to the Internet. This limitation regulates the fan-out of members when constructing trees, and reduces the overhead when running routing algorithms on the mesh. Benefits of the two-phase approach are that group management is maintained on the mesh rather than on trees per source, and a mesh is more resilient to the failure of members. Mesh-based approach is also motivated to better support multi-source applications. Constructed tree for a video source aims to optimize metrics such as stress (number of identical copies of a packet carried over a physical link), relative delay penalty (ratio of the delay between two members along the overlay to the unicast delay between them), and resource usage. Results indicate that end system multicast is a promising approach for enabling conferencing applications in dynamic and heterogeneous Internet settings. The study demonstrates the necessity for self-organizing protocols to adapt to both latency and bandwidth metrics. In

practice, end system multicast is widely and successfully used for single source video streaming on the Internet. Although its applicability to multi-source videoconferencing is mentioned, to the best of our knowledge, its practical usage in this context is not that common and related performance results are not reported. Different than this study, we target multi-source videoconferencing sessions in particular.

Finally, [11] proposes an approach for multi-sender 3D videoconferencing using end-system multicast, which has a problem domain similar to our study. It maintains a conference session protocol for small number of participants where each participant can act as video source for the rest, and frequent changes are handled quickly. It also offers a method of soft-join which involves locating a new participant into the multicast tree as fast as possible. Unlike video sharing, multicast tree construction in [11] is done via rendezvous points (RP) acting as central machines. This makes decision-making fast and reliable. However, due to the nature of centralized solutions, it suffers from single point of failures. In addition, multicast tree construction involves controlling all multicast trees which increases operational complexity and cost. While constructing multicast trees, the RP can choose one idle/available participant for acting as a reflector node (node that relays other participant's video without watching) without its consent.

Our approach differs from the prior work discussed above in a number of ways. First of all, it is not based on an application layer multicast model or overlay-based approaches requiring the availability of dedicated servers. Our solution employs a fully distributed model utilizing P2P principles for the dissemination of media content. Hence, problems of single point of failure and server overload are avoided. As discussed in the following section, our main assumption dictates that each conference participant can produce and receive only one video signal at any given time and send/relay one or more video signal(s) based on its computing power and Internet connection speed. This, together with the P2P media dissemination model, leads to balanced overhead at the participants. A solution should be efficient in utilizing participants' bandwidth and scalable at saving media sources' bandwidth consumption. Our system achieves this via the spanning tree configuration and management of conference participants. Besides, our approach does not allow resource sharing without a participant's consent.

3. Peer-to-peer approach for video transmission for participants with single video I/O

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 an 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 cannot act as an intermediate peer passing a video signal to another peer, and an intermediate peer cannot send its own video signal to anyone else.

As shown in Fig. 1, in an 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.

3.1. 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

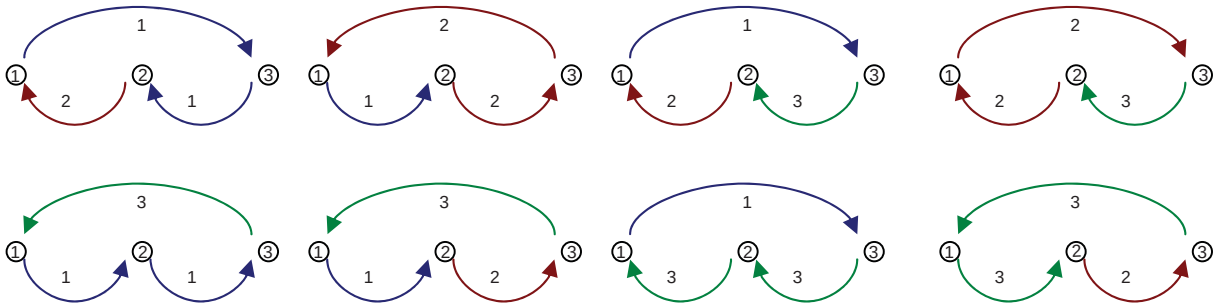


Fig. 1. In a P2P MP videoconference with three participants each participant can see any other participant. (The numbers on the arrows indicate which participant's video signal is being transmitted over the corresponding link.)

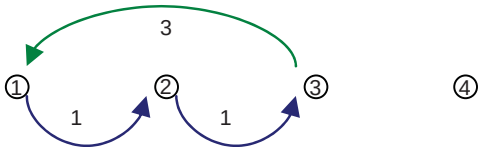


Fig. 2. Participant number 4 cannot see participant number 2 in this configuration.

are handled by the appropriate protocols as described in the Internet multimedia architecture. 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 on the basis of an object called a "*chain*." In this context, 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.

3.1.1. 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 ($\bar{\mathbf{e}}$) 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.

3.1.2. 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 eight 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

{ <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 >}

(a)

{<1,2,3, ē >, <3,1, ē >, <4,e>}

(b)

Fig. 3. Chains corresponding to the configurations in Figs. 1(a) and 2(b).

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 cannot 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.

```
Participant k receives a video request
   message from participant 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
```

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 cannot 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 the 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.

4. Implementation

Our P2P MP videoconferencing prototype has five main modules as demonstrated in Fig. 4. The first module is for conference management. Although a standard conference management application can be used for this purpose, in this prototype, a simple conference management application of our own with basic functionalities such as conference creation and joining in and leaving a conference is developed. Another responsibility for this module specific to our application is to establish the unique session IDs for each conference participant.

The second module, P2P manager, communicates with the conference manager to send current chain status (to be displayed on the screen), and specific events like user login/logout and peer crashes. A user interacts with the system using the graphical user interface (GUI) module which is implemented as an integral part of the conference manager in the prototype system. Video requests from other participants are given to the system through the GUI.

The P2P video manager is responsible for deciding how to multicast its local video. For this purpose, it includes a unit to construct the chains. P2P manager does the remote updates by sending configuration messages as well. Another responsibility of P2P manager is listening to the keep-alive messages of the participant in the same video chain. If it detects a crushed participant, (i) and if it is the head node, then it determines a new chain and implements the required changes by sending configuration messages, (ii) else, it notifies the head node about the crushed peer. P2P manager also runs the delay measurement unit which has a vital importance for the chain optimization process.

The video module handles video capture, compression/decompression, display 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 bit streams 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. Video device is configured by the P2P video manager to stop/start capture/display video.

P2P gateway application is a simple socket-based program which just sends and receives

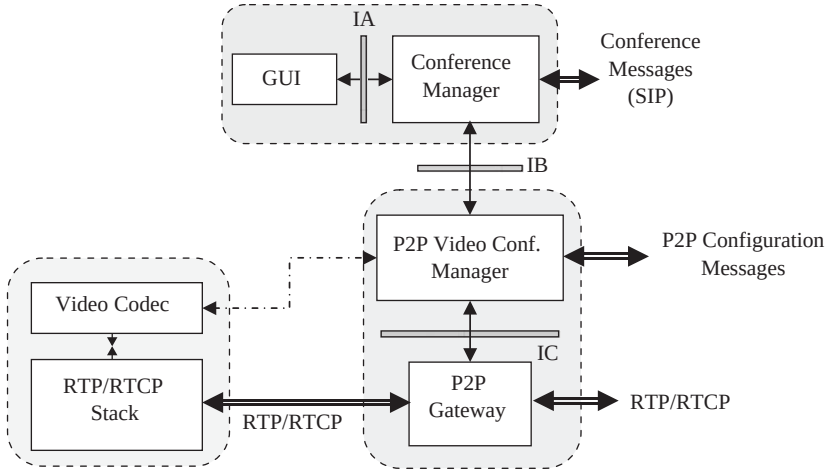


Fig. 4. General structure of the P2P MP videoconference prototype.

packets to and from pre-determined UDP sockets. It sends the local packet from the video unit to their destinations, and it may re-send (relay) the packets it receives to another destination or pass them to the video module depending on its current role in a chain.

5. Chain optimization

The chain construction algorithm described above and in [7] does not consider the network delays between the participants. This may result in longer than necessary overall delays. For instance, assume that three users located at different national networks participate in a conference session. At a given time, user@US may have a chain as shown in Fig. 5(a) which has a long detour. Using the chain given in Fig. 5(b) instead can decrease the overall delay significantly. This reordering may also help to reduce the total packet loss rate because the longer routes will be more susceptible to congestion. To address this issue, we added a step to the chain construction method to search for the optimal chain orderings minimizing the total delay on a chain.

We ignore the computing power parameter of a participant and concentrate on the network delay in re-constructing the chains. Since we need the network delays between peers to determine the optimal chain, we implemented a delay measurement protocol which is based on a clock synchronization algorithm described in [19]. The delays are measured for a participant as soon as it registers to a conference. All participants except the new registered one wait for a random time (to eliminate collisions) and run the delay calculation for the newly registered participant. All participants' delay matrices are updated after the delay measurements are complete.

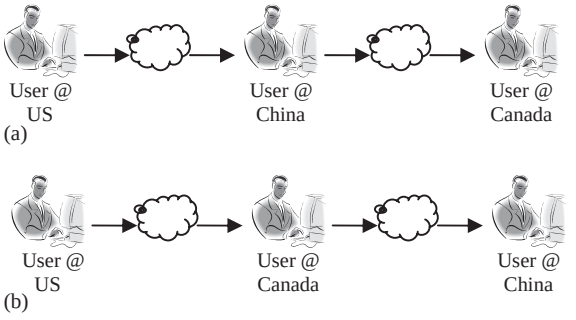


Fig. 5. Chain reconstruction example: (a) erroneous ordering; (b) reconstructed ordering.

We implemented the chain-order optimization as part of a more general solution described in the next section where some of the peers have the capability to transmit more than one video signal. The solution is based on a two-stage heuristic and may not be necessary for short chains where a full enumeration is viable.

A separate thread is reserved for chain re-construction and it runs the optimization algorithm mentioned above periodically. Whenever a participant joins into a chain, it is tried to be located immediately without regarding the optimized network delays. The re-construction for optimized delays is performed later when the chain head becomes available. This concept is called soft-join.

Delays are assumed to be static and they are calculated only once at the conference login. However, they may change during a conference session. The delays reported by the RTCP messages may be used to assist in handling varying delays. So as a further improvement, the delay optimization process may be run when the delays reported by RTCP change dramatically.

6. P2P MP videoconferencing for participants with multiple video outputs

Single I/O model and its optimization offer a good solution for MP videoconferencing for many practical cases where all participants are connected over low bandwidth links or do not want to handle more than one video. However, it is natural to expect that some of the participants may have larger access bandwidths and computing powers to source more than one video signal at a given time. Clearly, having such participants can help in cases where the single I/O system cannot fulfill all video requests.

In the literature, finding a multicast tree for a multi-sourced conference where the number of

node outputs is limited is known as the minimum cost degree-constrained spanning tree problem and identifying such a tree is computationally NP-hard [17]. Thus, several heuristics have been offered to find a qualified solution under time or computing power constraints. One such solution is an iterative search for eliminating degree-constraints while considering the total delay [10]. Another one is a randomized approach that mainly focuses on the delay problem without disturbing the I/O constraints [15,16]. The iterative solution, which has been implemented and ported to the current application [3], is described in the next section.

6.1. Two-stage heuristic

We consider the two-stage heuristic algorithm reported in [10] for the solution of the generalized MP videoconferencing problem. The first stage involves constructing a minimum spanning tree (MST) from a complete graph (this graph is constructed using global delay matrix described in the optimization part above) without considering the degree constraints. The second stage modifies the spanning tree to satisfy the given I/O constraints (degrees) for all participants. The details of the algorithm and its procedures are outlined in Fig. 6. In the first step, an MST can be calculated in polynomial time using greedy algorithms like Kruskal's [12] or Prim's [14]. After implementing the first step, the constructed tree is processed to adjust the I/O constraints of the nodes. This step involves selecting a node that violates the I/O constraint disconnecting the highest-cost edge from this node. Next, the sub-tree created by this operation is re-attached to the current tree. This operation goes on until all nodes have valid number of I/O connections. This heuristic is not guaranteed to find a solution under all cases. That is, when it fails to find a solution, it does not mean that no solutions are available. However, we will declare no solution in these situations. As stated in the previous section, optimization of the single I/O case can also be handled by this algorithm.

```
Find an efficient MST using Kruskal algorithm
Call degree-resolve() method for calculated MST

degree-resolve(){
        1- Select a participant (let p) from the calculated MST that does not
        satisfy the degree constraint. If two or more candidates exist then select
        one randomly
        2- Remove an out-edge link (let l) which has the largest delay from p.
        This step divides our spanning tree into two sub trees
        3- Try to attach the second tree (which does not include the source) to
        the first one. While combining trees, consider low-delay and degree
        constraints
        4- Repeat this procedure until each participant satisfies the degree
        constraints defined for themselves.
}
```

Fig. 6. Algorithm for two-stage heuristic.

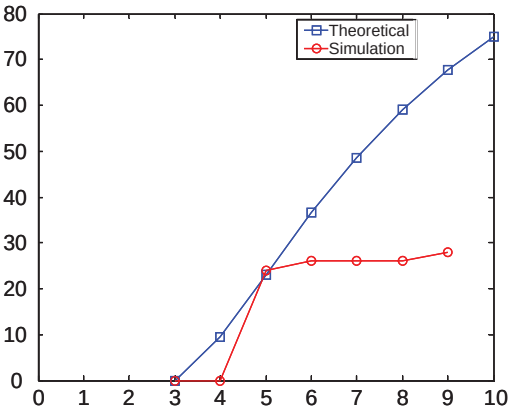
The advantages of this approach are that it is simple and easy to use. Results of this algorithm show that it is feasible to use when the participant count is less than or equal to 20 [10].

7. Results and conclusions

We implemented and tested the feasibility of a system for P2P multipoint videoconferencing. It can be used to extend almost any point-to-point videoconferencing system to a multipoint one with minimal increase in the complexity and no additional hardware support.

The system architecture allows us to use off-the-shelf software components for most of the system elements. We tried to stay compliant with the standards whenever possible.

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. Fig. 7 shows theoretical and simulation results obtained for the percentage of unachievable configurations, i.e., configurations for which one or more participants cannot receive video from another participant of their choice (as a percentage of all configurations) as a function of the number of conference participants for the single I/O case. The theoretical values are determined by generating all possible requests for the given number of users, determining if a solution is available, and enumerating the solutions. When compared to the theoretical values, the simulation results show that, as the number of participants gets



Simulation results

	All Requests	*Granted Requests*	*Rejected Requests*	*Others*
4	788	780	5	3
5	809	597	196	16
6	1001	725	264	12
7	1210	789	283	38
8	1176	859	304	13
9	1600	1139	438	23

	All State States	*Valid States*	*Invalid States*	*Deadlock States*	*Dead lock-free States*
3	27	27	0	0	54
4	256	232	24	66	630
5	3125	2405	720	1692	7928
6	46656	29616	17040	34840	113240
7	823543	42383	400260	712110	1827588

Theoretical results

Fig. 7. Theoretical and simulation results for the single video I/O case: unachievable configurations' percentage (y-axis) for a given number of conference participants (x-axis). N is the number of participants.

larger, the percentage of unachievable configurations does not increase as much. This is because the number of feasible configurations is larger than the infeasible ones when a random request is generated starting from a feasible configuration (as compared to starting from a completely disconnected configuration).

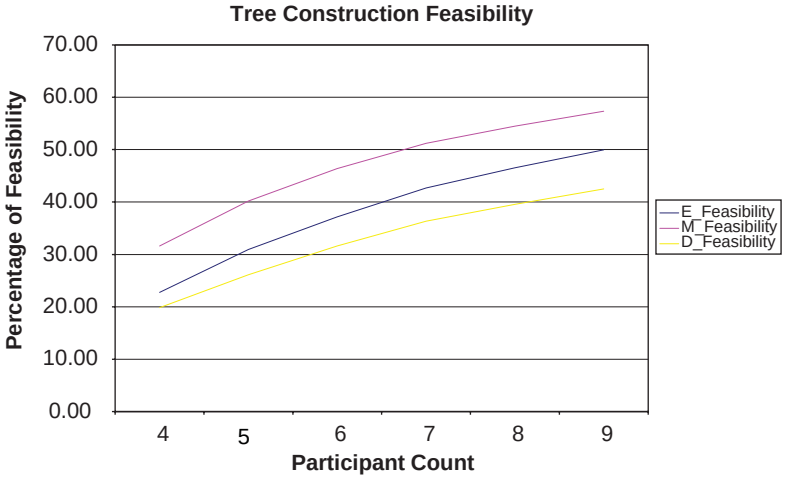
The computational burden caused by the P2P application on the system is observed to be insignificant (less than 1% CPU time on a 2.4 GHz Pentium).

Another runtime test for the application is calculating the network traffic and memory load in the system stemmed from the application. Fig. 8 shows the run-time cost of the P2P application for 15 s while both incoming and outgoing channels are busy. CPU idle time (upper line) has been measured and it shows that our P2P application leads to a 5–7% computational load. Another logged criterion is network utilization. Just for the sake of dial-up users, the P2P prototype encodes three frames per second which are MPEG4 encoded using ffmpeg codec (freely distributed

Fig. 8. Run-time cost of the P2P application: measured parameters are CPU idle time (upper line) and UDP bytes per second (lower line).

@ www.sourceforge.net). For this case, the video (UDP) transmission load is almost 60–70 Kbps.

We ran a formal verification study confirming our protocol. We verified the feasibility of the optimization algorithm that we used for reducing



Feasibility benchmark of 3 methods; Kruskal MST, Enumeration of all chains, Two Phase Heuristic

(a)

(in msecs)	*ETREE*	*DTREE*
4	1.25	1.42
5	1.39	0.48
6	4.39	0.63
7	31.87	0.62
8	207.8	0.31
9	1621.55	1.1

CPU times in milliseconds for algorithm run times.
Benchmarked algorithms are (E)numeration and (D)egree Contrained using Two Stage Heuristic

(b)

Fig. 9. (a) Chain feasibilities and (b) CPU-cost test results. Test machine is Intel (R) Pentium (R) 4 CPU 2.60 GHz, 512 MB of RAM, MS Windows XP Pro v.2002.

the delay on long chains (two-stage heuristic) performing some tests on randomly generated chains and the results are displayed in Figs. 9(a) and (b). For each test (that means for each entry of the horizontal axis), 5000 sample chains have been generated and all of the algorithms have been forced to find a solution. Horizontal axis shows the participant count and the vertical axis shows the feasibility percentage. Fig. 9(a) depicts the comparison of the three algorithms offered for tree construction. First algorithm Kruskal MST (upper line) has the best results since it does not take into account the degree constraints. On the other hand, for a chain, the best solution can be found by enumerating (intermediate line) all states and selecting the minimum cost state. Finally, two-stage heuristic (lower line) has a feasibility level nearly approaching 50% for the original cost.

Fig. 9(b) shows the CPU times for the enumeration and the heuristic solutions. As the number of participant increases, enumeration suffers from the time needed to find the best solution. Unlike enumeration, heuristic-based search for chain reconstruction finds a feasible solution in a reasonable time.

We believe that a potential solution for wide-scale deployment of MP videoconferencing is through using P2P systems. Our protocol and prototype demonstrates the feasibility of such approaches. The next step will be organization of large scale trials on the Internet. An extension of our system could be the utilization of scalable video to handle heterogeneous Internet access technologies of the conference participants. For instance, some participants might prefer to view two high-quality videos instead of one. Scalable video techniques may also be used to adapt the video bit rate at each node based on the upstream bandwidth. As another optimization, locating a relay node, with a high bandwidth, close to the video source would be useful to avoid the degradation of the video quality for the successive nodes in the chain.

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] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, Proceedings of ACM SIGCOMM'02, Pittsburgh, Pennsylvania, August 2002.
[3] T. Celebi, M.R. Civanlar, O. Ozkasap, P2P multi-point videoconferencing on the Internet. IASTED PDCN 2005, International Conference on Parallel and Distributed

Computing and Networks, Innsbruck, Austria, February 2005.

[4] Y. Chu, S.G. Rao, S. Seshan, H. Zhang, Enabling conferencing applications on the Internet using an overlay multicast architecture, Proceedings of ACM SIGCOMM'01, San Diego, California, August 2001.

[5] Y. Chu, S. Rao, H. Zhang, A case for end system multicast, in: Proceedings of ACM Sigmetrics, June 2000.

[6] M.R. Civanlar, R.D. Gaglianello, G.L. Cash, Efficient multi-resolution, multi-stream video systems using standard codecs, J. VLSI Sig. Proc. 17 (2–3) (1997) 269–279.

[7] M.R. Civanlar, O. Ozkasap, T. Celebi, Peer-to-peer multipoint videoconferencing, ICIP, IEEE International Conference on Image Processing, Singapore, October 2004.

[8] Y. Cui, K. Nahrstedt, Layered peer-to-peer streaming, Proceedings of NOSSDAV'03, Monterey, California, June 2003.

[9] D. Eager, M. Vernon, J. Zahorjan, Minimizing bandwidth requirements for on-demand data delivery, IEEE Trans. Knowledge Data Eng. 13 (5) (2001).

[10] N. Funabiki, J. Kawashima, A. Kubo, K. Okayama, T. Higashino, A proposal of a two-stage heuristic algorithm for peer-to-peer multicast routing problems in multihome networks, The Fifth Metaheuristics International Conference (MIC-2003), August 2003, pp. 20-1-6.

[11] 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.

[12] J.B. Kruskal, On the shortest spanning subtree of a graph and the traveling salesman problem, Proc. Am. Math. Soc. 7 (1) (1996) 48–50.

[13] V.N. Padmanabhan, H.J. Wang, P.A. Chou, K. Sripanidkulchai, Distributing streaming media content using cooperative networking, Proceedings of NOSSDAV'02, Miami, Florida, May 2002.

[14] R. Prim, Shortest connection networks and some generalizations, Bell System Technical J. 36 (1957) 1389–1401.

[15] G.R. Raidl, An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem, in: C. Fonseca, et al. (Eds.), Proceedings of the 2000 IEEE Congress on Evolutionary Computation, IEEE Press, 2000, pp. 104–111.

[16] G.R. Raidl, B.A. Julstrom, A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem, in: J. Carroll, et al. (Eds.), Proceedings of the 2000 ACM Symposium on Applied Computing, ACM Press, 2000, pp. 440–445.

[17] R. Ravi, M.V. Marathe, S.S. Ravi, D.J. Rosenkrantz, H.B. Hunt, Many birds with one stone: Multi-objective approximation algorithms, in: Proceedings of 25th Annual ACM STOCS, 1993, pp. 438–447.

[18] S. Sheu, Kien A. Hua, W. Tavanapong, Chaining: A generalized batching technique for video-on-demand, Proceedings of the IEEE International Conference on Multimedia Computing and System, Ottawa, Ontario, Canada, pp. 110–117, June 1997.

[19] A.S. Tanenbaum, M.V. Steen, Distributed Systems—Principles and Paradigms, Prentice Hall, New Jersey, 2002, pp. 247–249.

[20] D.A. Tran, K.A. Hua, T. Do, ZIGZAG: An efficient peer-to-peer scheme for media streaming, Proceedings of IEEE Infocom, 2003.

[21] D. Xu, M. Hefeeda, S. Hambrusch, B. Bhargava, On peer-to-peer media streaming, Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), 2002.