A Multiparty Videoconferencing System Over an Application-Level Multicast Protocol

Chong Luo, *Member, IEEE*, Wei Wang, Jian Tang, *Member, IEEE*, Jun Sun, *Member, IEEE*, and Jiang Li, *Senior Member, IEEE*

***Abstract*—Increased speeds of PCs and networks have made media communications possible on the Internet. Today, the need for desktop videoconferencing is experiencing robust growth in both business and consumer markets. However, the synchronous delivery of high-volume media content is still a big challenge under a current heterogeneous Internet environment. In this paper, we present a multiparty videoconferencing system based on a peer-to-peer (P2P) solution. The contribution of our paper is twofold. On the one hand, we design an application-level multicast scheme which intends to tolerate the heterogeneity in videoconferencing applications. Design tradeoffs are analyzed and our decisions are made based on extensive experimentation. On the other, we design a five-layer architecture for implementing a multiparty videoconferencing system. This architecture makes a clear-cut distinction between different functional modules and therefore provides rich flexibility in feature adaptation. We believe that our work can be a helpful reference in other efforts on building desktop videoconferencing systems.**

***Index Terms*—Application-level multicast, multiparty videoconferencing, peer-to-peer networking.**

I. Introduction

COMMUNICATION is an essential part in our daily lives. For quite a long time, the communication between geographically distant individuals has been limited to mails and telephones. With the advances in technology, people of today are provided with videoconferencing, which embraces text, audio, and video in real-time two-way communications.

Manuscript received November 15, 2006; revised July 29, 2007. This work was carried out at Microsoft Research Asia. The associate editor coordinating the review of this manuscript and approving it for publicatrion was Dr. Klara Nahrstedt

C. Luo is with the Institute of Image Communication and Information Processing, Department of Electronic Engineering, and the Shanghai Key Laboratory of Digital Media Processing and Transmissions, Shanghai Jiao Tong University, Shanghai, China, and also with Microsoft Research Asia, Beijing, China (e-mail: chong.luo@microsoft.com).

W. Wang is with the School of Electronics and Information Engineering, Sichuan University, Chengdu, China (e-mail: weiwang@scu.edu.cn).

J. Tang is with the Microsoft Research Asia, Beijing, China (e-mail: jtang@microsoft.com).

J. Sun is with the Institute of Image Communication and Information Processing, Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China, and with Shanghai Key Laboratory of Digital Media Processing and Transmissions, Shanghai Jiao Tong University, Shanghai, China (e-mail: junsun@sjtu.edu.cn).

J. Li is with the Microsoft Research Asia, Beijing, China (e-mail: jiangli@microsoft.com).

The work presented in this paper was carried out in Microsoft Research Asia.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2007.907467

In the recent decade, we observed increasing needs for desktop videoconferencing, especially multiparty videoconferencing, in both SOHO (Small Office and Home Office) users and end consumers. Knowing from the studio-based conferencing which requires dedicated ISDN lines and expensive equipments, desktop videoconferencing runs on normal PCs and works even with broadband connections. For these reasons, desktop videoconferencing has gained popularity, however, it is also facing many rigorous challenges beyond those of studio-based systems.

- *Addressing and connectivity:* The shortage of network addresses leads to the widespread use of the network address translator (NAT). This presents a challenge in both addressing and connectivity, since an IP address is no longer a unique identification. The studio-based solution over IP tackles this problem by using a worldwide dialing system and by employing a gatekeeper to provide address translation, call control, and routing services. In desktop systems, a lightweight solution which provides similar functionalities is needed.
- *Heterogeneous network conditions:* Desktop videoconferencing systems should support not only Ethernet users, but also home users with broadband connections. However, their network conditions, especially bandwidth resources, differ greatly. An Ethernet user is able to deliver high-quality video to dozens of users, while a broadband user may have difficulties in sending low-quality video to more than two receivers. This raises a big challenge when we want to bring all types of users into one multiparty conference.
- *Real-time requirements:* Videoconferencing is an application for real-time two-way communications, and therefore has a very stringent latency requirement. It has been shown that it will become objectionable if the latency of a voice communication exceeds 300 ms. A similar requirement applies to video communications. Though an IP-based videoconferencing system can hardly always meet this requirement in the real Internet environment, it should try to reduce the latency as much as possible.

We surveyed the state-of-the-art desktop videoconferencing systems, and found that systems for the business market and the consumer market tried to resolve these problems through different approaches. Commercial software, mostly taking the form of web conferencing, resorts to additional servers to solve the above problems. Its media streams are relayed by a central server or a collection of switching centers, as introduced in WebEx [13] MediaTone and Raindance [9] SwitchTower technology. Although using server relays can relieve the bandwidth

stress at end users, it introduces a long delay, often on the order of seconds, for data and video communications. Our experiments on WebEx show that the video signal can be delayed for a few seconds even when the two communicating parties are close-by, e.g., in the same LAN. While a delay for a few seconds is still tolerable in data exchange, it is objectionable for video communications.

In the consumer market, videoconferencing is interpreted as voice/video chatting in most cases. Dominant systems include MSN Messenger [14], AOL Instant Messenger [1], Yahoo Messenger [15], PalTalk [8], ICUII [3], and iSpQ [5]. The companies usually provide them as free software or service and, therefore, do not deploy a large number of servers to relay the media data. Some software [14], [1], [15] allows only one-to-one video communication; some others [8], [3], [5], although allowing a user to simultaneously view multiple videos, are only based on independent point-to-point sessions. Usually, when the number of participants exceeds three, the voice/video communication is provided at a wretched quality. Moreover, if two communicating parties cannot reach each other directly because of the network configurations, the software provides little support in routing the communication.

From the survey, we found that neither commercial videoconferencing systems nor consumer-oriented video chatting systems took advantage of user network resources. In fact, when a user does not have enough bandwidth to feed all his receivers, he can request other peers who have spare network resources to help relay the data. Such kind of peer-to-peer (P2P) solution has demonstrated its efficiency in many file sharing and VoIP applications, such as Kazaa [6] and Skype [12]. However, videoconferencing applications have more stringent latency requirements than file sharing applications, and demand more bandwidth than voice communications. Therefore, existing P2P solutions for asynchronous data delivery or for low-bandwidth synchronous communications cannot be directly applied to videoconferencing. In view of this, we designed a P2P solution that was tailored to videoconferencing applications. This kind of solution is also called application-level multicast (ALM) or end system multicast (ESM) in literature [17], [21]. Based on our proposed ALM solution, we have designed and implemented a small-scale multiparty videoconferencing system.

The rest of the paper is organized as follows. Section II provides an overview of our multiparty videoconferencing system, and illustrates the five-layer design of the client application. In Section III, we analyze several tradeoffs in the design of an ALM scheme. These tradeoffs are further studied through extensive experimentation. The simulation methodology and evaluation results are presented in Sections IV and V. Finally, we conclude in Section VI.

II. System Overview

Our multiparty videoconferencing system is P2P in nature; however, in order to avoid the management and security issues associated with P2P applications, we adopt a hybrid system architecture. A server is deployed in the public domain and is used for user registration and conference setup. We consider that such a hybrid approach can take the advantages of both Client/Server and P2P architectures while eliminating their disadvantages.

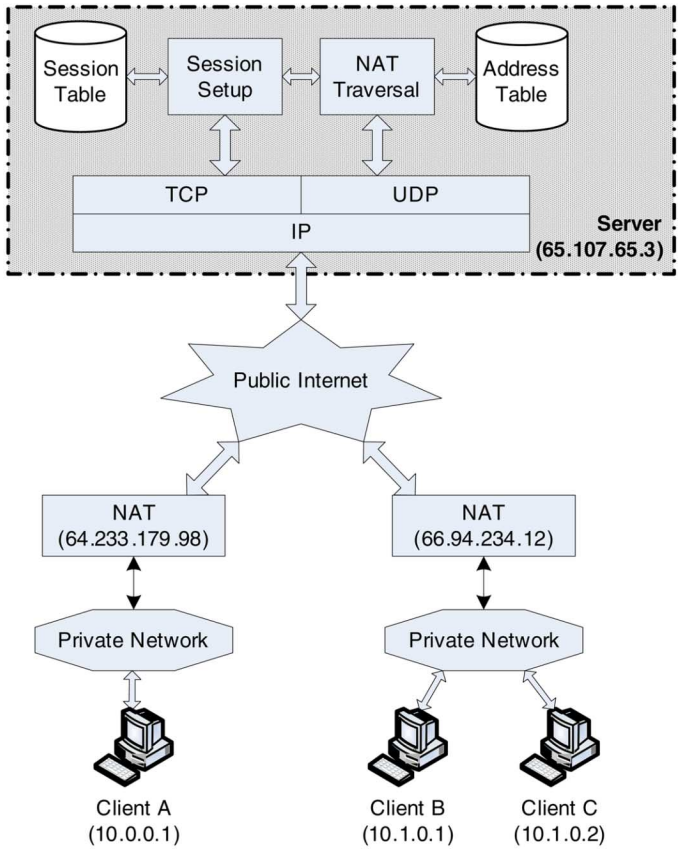


Fig. 1. System architecture.

A. Server Implementation

Referring to the current consumer-oriented multiparty videoconferencing systems, we find that, a majority of them adopts a hybrid approach too. However, their servers are solely responsible for user registration and authentication, and take too little responsibility in supporting multiparty conference and in improving the end-to-end connectivity. In our system, the server has two additional functionalities: 1) act as the man-in-middle for two communicating parties who are behind different NATs so as to improve their connectivity; 2) track the members of each conference and speed up the user joining process. These two functions are illustrated in Fig. 1.

1) NAT Traversal: In recent years, NAT is gaining popularity as a method to alleviate IPv4 network address shortages. It involves re-writing the source and/or destination addresses of IP packets as they pass through a router or firewall. While Client/Server applications can work transparently with NAT, most P2P systems fail completely or require special solutions to become NAT-enabled [18]. For P2P applications, the most critical problem introduced by NAT is the incoming-connectivity problem: the end host in a private realm does not have a stable IP address/port that can be reached by other hosts in the public realm or different private realms.

Engineers and researchers have explored many ways to traverse NAT. The most well-known technique is UDP hole punching, which was first publicly documented by Dan Kegel. Later, this idea is incorporated into a few experimental protocols, such as STUN [23] and ICE [22]. In our system, we

implement the basic version of STUN, and let the server act as the man-in-middle.

In addition to UDP hole punching, an idea from the ICE protocol is borrowed by our system: we require every client to retrieve as many usable addresses as possible before it logs on to the server. These addresses include the private address, the public address negotiated with the NAT if it is UPnP-enabled, and the IP multicast address if the client is in a multicast-enabled network. After the client logs on to the server, it informs the server of all the locally retrieved addresses, and then sends a few UDP probing packets from which the server can discover its external address. The address information is saved in the *Address Table*.

2) Session Setup: The session setup module takes the responsibility to provide mutual awareness of members in the same conference. It uses TCP as main communication channel for reliability. When a user initiates a new session, and registers the session with the server, the server will keep a record of this session and the invited member list in its *Session Table*. Then, whenever a member joins this session, the server informs him of all the existing members and their IP addresses. The server also informs all the existing members of the newly joining member and his IP addresses.

These IP addresses are retrieved from the *Address Table* and contains many address types, such as internal, external, UPnP, and IP multicast addresses. As we have mentioned earlier, when a client receives such a set of addresses, he will try them all, and choose the best one that works. After this point, the server is no longer involved in any data communications in that session.

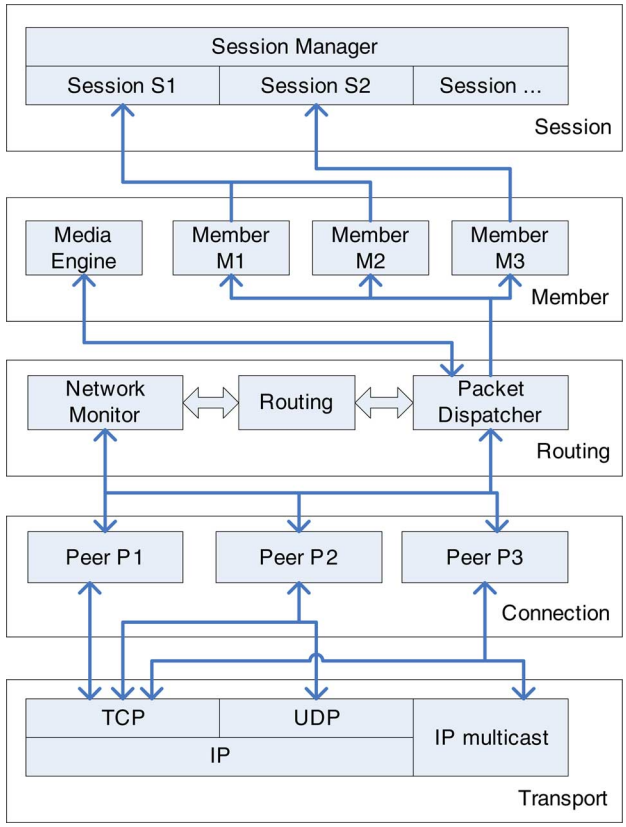


Fig. 2. Client implementation.

B. Client Implementation

We consider the implementation of client application as one of our main contributions. The client is implemented in a five-layer architecture, which spans both network layer and application layer. Fig. 2 shows the five-layer design. The arrows indicate how the streams flow across multiple layers.

1) Transport: Our system is built on top of TCP and UDP. UDP is a light-weight protocol suitable for media stream transmission, while TCP is a reliable protocol suitable for control message delivery. End systems in the same conference always try to build both TCP and UDP channels with other systems.

In addition, our system utilizes IP multicast when it is available. To be specific, each conference session is associated with an IP multicast address, which is randomly chosen by the conference initiator. This address is registered on the server and passed to all participating members when they join in. Upon receiving this address, every member tries to join this multicast group and if succeed, he will distribute his information through this channel. On the other hand, if another member receives the information of this member on the multicast channel, he knows that two of them are in the same multicast-enabled domain. Therefore, both of them will discard UDP channel, and use IP multicast for all the subsequent data communications.

2) Connection: The connection layer consists of a list of peer modules. Each peer module corresponds to an communicating party, and it is looked on as an abstract socket. By saying so, we mean that, when the client application wants to send a packet to a specific peer, it only needs to pass the packet to the corresponding peer instance, and specify whether this packet needs reliable transmission. Then, the peer instance will select an appropriate channel to transmit this packet. Usually, each peer module maintains two channels: TCP for reliable control message transmissions and UDP/IP-multicast for delay-sensitive media communications. The type of available connections is transparent to all above layers.

3) Routing: The routing layer is the core layer of our system. It consists of three modules: network monitor, application-level routing, and packet dispatcher. The network monitor periodically measures network dynamics, such as end-to-end latency and available bandwidth, and provides these information to the routing module. We have designed our own light-weight bandwidth estimation technique [25], but other techniques can also be used as long as they provide the same set of programming interface.

The routing module is the core of the core. It executes the application-level multicast (ALM) algorithm, and computes delivery paths to every subscribed receivers. The ALM algorithm design will be detailed in the next section. For the moment, we can treat it as a black box, which accepts inputs from the network monitor and provides output to the packet dispatcher. Besides, it also records the delivery paths of other members, so that it knows how to relay packets for others if needed.

The packet dispatcher is the busiest module in this architecture. Every outgoing and incoming packet goes through this module. For every packet, the dispatcher first checks whether the local site is the destination. If so, it forwards this packet up to the relevant member module. Then, it checks whether the local

site needs to relay this packet to any others. If yes, it consults the routing module for the next passing hop and pushes the packet down to the corresponding peer module.

Together, these three modules form an application-level router (ALR). Just like an IP router which forwards IP datagrams to the networks it connects to, an ALR forwards the application-defined packets to the peers it connects to.

4) Member: The member layer consists of one media engine and a list of member instances. The media engine generates local streams, and passes them to the packet dispatcher in routing layer. There is a one-to-one mapping between the member module in the application layer and the peer module in the connection layer. The member module keeps application-level properties of a member, such as the ID and the friendly name. It also remembers the sessions which the member is participating in.

The member module collects the media packets generated from the corresponding member, and is responsible for assembling A/V packets if channel coding is used. Then, it forwards the packets to the session in which it is actively participating.

5) Session: One special feature of our system is that it allows a user to attend multiple conferences. In a typical scenario, when a user is participating in a conference, another user calls in. With our system, the user can put the first session on hold, and accept the second one. Then, he can switch between these two conferences freely. In order to avoid confusing, we regulate that there is only one active session at any point of time.

Each session, together with the attendee list, is maintained by the Session module. Thanks to the clear separation between different layers, the connection and routing layers are totally unaware of which session(s) a member is participating in. In other words, even when a peer is present in two or more sessions, only one peer instance is maintained.

C. Control Protocol

Our system is a distributed system in nature. Therefore, it needs a control protocol to coordinate the conference members when network conditions or conference parameters change. Here, network conditions mean end-to-end latency and available bandwidth. Change of conference parameters includes the following.

- A member joins or leaves a session.
- A member puts a session on hold or resumes a session.
- A member requests to view or stop viewing some other member's video.
- A member starts or stops broadcasting his audio/video.
- A member changes his audio/video bit rate, etc.

Each event will trigger a series of status updates and message exchanges. For the brevity of this paper, we will not go into the details of how we handle these events. Instead, we will discuss the core part of the control protocol: the application-level routing.

In our system, each member monitors and measures its own network conditions, and periodically broadcasts the information to other members. Hence, every conference member has complete application-level topology information, including the connectivity, latency, and available bandwidth between any two end hosts. Based on these same status data, each conference

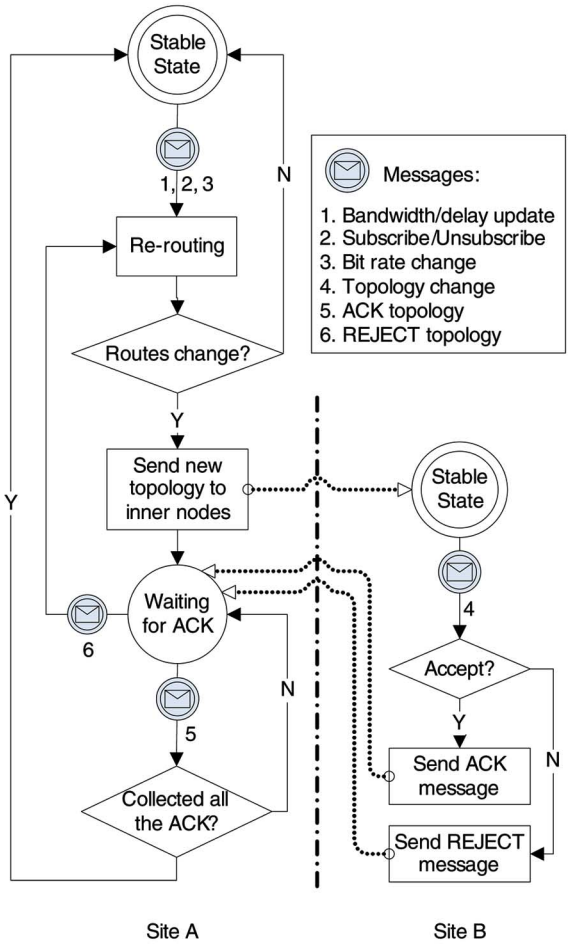


Fig. 3. Source-specific per-stream routing control.

member independently computes the ALM trees of its own streams. Then, if relay is needed, it will inform the relay nodes so that they can make the concerted effort to route the stream.

Fig. 3 describes the basic control process. In this example, site A is distributing audio/video stream, and site B helps in relaying A's data. When any of the following events happen, the data source (Site A in this example) needs to recompute its multicast tree. If the tree changes, it will forward the new topology to relay nodes.

1) Receive the updated end-to-end latency or available bandwidth information. Usually, when there is no abrupt change in the network, the tree topology will not change.
2) Receive subscription/unsubscription request. A member will receive this request when another member joins/leaves a conference, holds/resumes a session, or simply wants to view/stop viewing his video.
3) The user decides to change the stream quality (i.e., bit rate). Our system allows a user to change his media stream quality when, for example, he wants to upload some photos to an e-album during the conference.

If the newly computed multicast tree is different from the old one, the data source needs to send the new topology to the relay nodes, and waits for their acknowledgement. After all the relay nodes have accepted the new topology, the data source goes back to the stable state. However, if any of the relay nodes rejects

the topology, the ALM tree has to be recomputed. Rejection only happens in two rare cases: 1) there is an abrupt change of network condition after last topology update; 2) another data source has just requested this node to relay data and occupied its bandwidth resource.

Our system is unique in providing such a negotiating mechanism between data source and relay nodes. In previous ALM protocols [17], [21], the application didn't have much control over the bandwidth usage; it only relies on the underlying transmission protocol (e.g., RTP) to provide best-effort delivery. Such a strategy is prone to overloading "hinge" nodes, and degrades system performance as a result. On the other hand, the negotiating mechanism provided in our system ensures that no relay node will get overloaded, and media streams can be transmitted to subscribers in its full quality.

III. Application-Level Multicast Routing

The basic idea of application-level multicast (ALM) is to push the multicast related functionalities, such as routing and packet duplication, from IP layer to application layer. Such a scheme can work independently of the underlying hardware.

During the last decade, there has been a surge of ALM designs, tailored to various types of applications. We observed that different Internet applications had their distinct requirements on bandwidth, latency and scalability. In this paper, we focus our discussion on ALM protocols designed for small-scale videoconferencing applications, which are bandwidth-demanding and latency-sensitive although they require less scalability.

A. Related Work

In the literature, we found three pieces of representative work on ALM protocol design for videoconferencing applications.

End System Multicast (ESM) [17] is a pioneer among them. It is a fully distributed protocol, and adopts a mesh-first strategy in building multicast trees: end systems first self-organize into a rich connected graph (mesh), then data delivery trees are generated on top of the mesh based on the distance vector (DV) protocol. In their subsequent work [16], the authors explored the feasibility of enabling conferencing applications using ESM. In particular, the multicast tree is built on shortest widest paths, contrasting to the shortest paths used in conventional DV protocol. The main deficiency of this work is that it does not consider the effect of building multiple delivery trees over the same mesh.

Differing from Narada, ALMI [21] is a centralized protocol. It takes multiple data sources into consideration by using a shared data delivery tree. This tree is formed as a minimum spanning tree (MST) based on the end-to-end measurements, i.e., round trip delays in this context. Although MST achieves the optimal performance in terms of network resource usage, its end-to-end performance is left unexplored in [21]. Besides, in a shared tree, the network load of leaf nodes and inner nodes differs greatly when multiple data sources exist. As a result, the inner nodes are prone to being overloaded.

In contrast to Narada and ALMI, the protocol for multisender 3-D video conferencing [19] explicitly addresses the multisender requirement in a videoconferencing application. It assumes that a Rendezvous Point (RP) exists, and the RP is responsible for computing multicast trees for all data sources. The novelty of the work lies in the double-algorithm approach for managing the soft join (subscription to a stream) requests. The key shortcoming of this protocol, as the authors pointed out, is its static nature with regard to the network conditions.

From this brief review, we can see that an ALM protocol design involves many tradeoffs. In the rest of this section, we will first discuss two basic tradeoffs and give our choices. Then, we will bring forward another consideration about incorporating IP multicast into an ALM protocol.

B. Shared Tree versus Source-Specific Trees

In designing an ALM protocol for multiparty videoconferencing, a very basic tradeoff is between using a single MST and using multiple source-specific trees for multisender data delivery. By definition, the MST has the optimal network resource usage, which is defined as $\Sigma_{i=1}^{L} d_i * s_i$ where L is the number of links in the multicast tree, d_i is the delay of link i, and s_i is the stress of link i [17]. However, for real-time two-way communications, what we care the most is the end-to-end performance. MST doesn't have any superiority in this context. On the contrary, research [24] has already shown that shared trees do not have as good delay properties as source-specific trees.

Besides, shared tree produces less throughput. In a N-party videoconference, the shared tree has $N-1$ edges. In other words, all the media traffic is concentrated on $N-1$ transmission links. This brings a very heavy load to these links. Through simple deduction, one can obtain that the minimal uplink bandwidth required for each node is $N \times (d-1)+1$ times of stream bit rate, where d is the node degree. According to this equation, the required uplink bandwidth grows linearly with node degree, and the step is N. In contrast, source-specific trees distribute the network load on a maximum of $N(N-1)/2$ links, thus greatly reduce the stress on any single link.

Based on the above two considerations, we favor source-specific trees to shared trees. This choice is also supported by the experiments to be presented in the next section.

C. Centralized versus Distributed Multicast Tree Construction

We observed two strategies for constructing source-specific trees in previous work. ESM [17], [16] constructed multicast trees based on the distance vector protocol. In particular, the routing protocol in [16] prioritized bandwidth metric over delay matric, and used a kind of the *shortest widest path* algorithm. In contrast to the distributed DV protocol used in [17] and [16], the protocol for multisender 3-D videoconferencing [19] assumed that a rendezvous point (RP) existed, and it was responsible for calculating multicast trees for all senders. Just because of this, the tree construction algorithm can apply a constraint on bandwidth usage. To be specific, it was assumed that each node v in the network graph had a maximum out-degree $\mathrm{outd}_{\max}(v)$. Therefore, the resulting multicast trees should satisfy: for each node v, $\mathrm{outd}(v) < \mathrm{outd}_{\max(v)}$.

We favor the second approach to the first one. The main reason is that the first approach, lacking of a coordinating mechanism, cannot balance the bandwidth usage intelligently. It relies on the transport network to perform best-effort delivery. As a result, the network is prone to congestion especially when

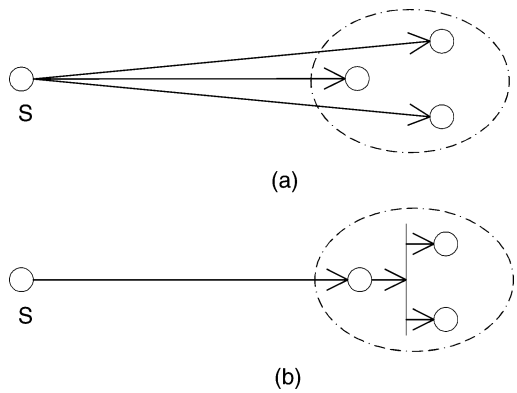


Fig. 4. Data delivery with and without IP multicast. (a) Delivery without IP multicast and (b) delivery with IP multicast.

the stream bit rate is at the same magnitude as the available bandwidth. In the experiment section, we will demonstrate this point. Our another interesting finding is that, the widest path first algorithm cannot achieve higher throughput than the shortest path first algorithm, sometime even lower.

D. Incorporating IP Multicast

IP multicast is the most efficient mechanism to enable one-to-many data delivery, since it ensures no duplicate packets on the physical link. Currently, most LANs, including educational institutes and large-scale corporations, have deployed IP multicast. However, because of some inherent problems, IP multicast is not generally available to average end users. This has been the motivation of the research on ALM. Nevertheless, we can incorporate IP multicast into an ALM protocol and make use of IP multicast as much as possible.

In a multiparty videoconference, it is quite often that two or more members are from the same multicast-enabled LAN. If these members all request video from a specific member s, then s only needs to send one video copy to one of the requestors, who can then relay the data to all the others through IP multicast. This point is illustrated in Fig. 4. Suppose the bandwidth of s permits more than three upload links. If we apply the original Dijkstra's algorithm, the data will be sent in multiple Unicast [Fig. 4(a)]. This is apparently not as reasonable as the delivery scheme shown in Fig. 4(b), where two long transmission links are avoided by making use of IP multicast.

In order to leverage IP multicast, we make extensions to ALM tree construction algorithm. The extension is quite general, and can be applied to other ALM algorithms without much modification. Without loss of generality, we illustrate it through the extension to application-level Dijkstra's shortest path first (SPF) algorithm. In order to take bandwidth constraint into consideration, we first filter the links whose available bandwidths are smaller than the data transmission rate. The following is the formal problem definition and solution description.

Problem Definition:

- Given (V, E, C, δ, s)
  - — (V, E)-directed graph, $s \in V$ is the source node.
  - — $\delta()$-cost function. Each edge $e = (u, v)$ in E has a non-negative cost there does not exist $\delta(u, v)$.
  - — C-set of multicast-enabled LANs (or clusters). $C = \{c_i\}$ and $c_i = \{v_i\}$, if $u, v \in c_i, \delta(u, v) = 0$.

```
Dijkstra_IPM(G, s)
01  for each vertex u ∈ G.V()
02      u.setd(∞)
03      u.setparent(NIL)
04  s.setd(0)
05  S ← ∅
06  Q ← G.V()
07  while(!Q.IsEmpty())
08      u ← Q.extractMin()
09      S ← S ∪ u.cluster()
10      for each w ∈ u.cluster()
11          for each v ∈ w.adjacent() do
12              if(v.getd()> w.getd()+dist(w, v))
13                  v.setd(w.getd()+dist(w, v))
14                  v.setparent(u)
15                  for each x ∈ v.cluster()
16                      if(x.getd()> v.getd())
17                          x.setd(v.getd())
18                          x.setparent(v)
```

Fig. 5. Extended Dijkstra's algorithm for routing with IP multicast.

- Find the shortest path tree T rooted at s that satisfies for any cluster $c \in C$: if $s \notin c$, there is one and only one edge $e = (u, v)$ in T where $u \notin c$ and $v \in c$; otherwise, $\text{nexists}e = (u, v)$ in T where $u \notin c$ and $v \in c$.

Our Solution:

The problem can be solved by a three-step process: first, all the nodes in the same cluster can be looked on as a single cluster node; then, we apply the original Dijkstra's algorithm to the graph with special cluster nodes; at last, we reverse the cluster nodes into ordinary nodes and add necessary edges to the generated multicast tree. The details can be described as follows:

Step 1: Generate (V', E, δ').

- if $v \in c, v$ is replaced with c in V^{ϵ};
- the cost function is changed to

$$\delta'(x,y) = \begin{cases} \delta(x,y), & \text{if } x,y \in V \\ \min_{u \in x} \delta(u,y), & \text{if } x \in C, y \in V \\ \min_{v \in y} \delta(x,v), & \text{if } x \in V, y \in C \\ \min_{u \in x, v \in y} \delta(u,v), & \text{if } x \in C, y \in C. \end{cases}$$

Step 2: Use the original Dijkstra's algorithm to compute the shortest path tree T' on (V', E, δ').

Step 3: Generate T from T', starting from $T = \varnothing$. For each edge $(x, y) \in T'$:

- if $x \in V$ and $y \in V, T \leftarrow T \cup (x, y)$;
- if $x \in C$ and $y \in V, T \leftarrow T \cup (u, y)$, where $u = r \big| \min_{r \in x} \delta(r, y)$;
- if $x \in V$ and $y \in C, T \leftarrow T \cup (x, v) \cup (v, w_i)$, where $v = t \big| \min_{t \in y} \delta(x, t), w_i \in y$;
- if $x \in C$ and $y \in C, T \leftarrow T \cup (u, v) \cup (v, w_i)$, where $(u, v) = (r, t) \mid \min_{r \in x, t \in y} \delta(r, t), w_i \in y$.

Implementation:

Fig. 5 shows the pseudo code that implements our proposed algorithm.

E. Our Scheme

In our multiparty videoconferencing system, we used a distributed-centralized approach to construct source-specific multicast trees: on the one hand, the delivery paths of a specific stream are computed at the source in a centralized manner; on the other, there is no rendezvous point in the system, instead, each source calculates its own multicast tree in a distributed manner.

Both available bandwidth and delay are considered in multicast tree construction, and our objective is to optimize data transmission delay under bandwidth constraints. Usually, if a node has sufficient uplink bandwidth for all the receivers, its ALM delivery will degrade itself to multiple Unicast. This is because that direct transmission usually (though not always) introduces less delay than relay (triangle inequality).

Details of our previous tree construction algorithm can be found in [20]. Now we are working on a refined version which incorporates IP multicast and simplifies tree construction operations. We would like to mention that, thanks to our flexible system design, it takes little effort to replace the ALM algorithm.

IV. Experiment Evaluation

The evaluation objective is to verify our design decisions in ALM tree construction. We use collected real Internet datasets to drive the simulation. In this section, we will introduce the data preparation and evaluation methodology. The experimental results will be presented in the next section.

A. Data Preparation

Currently, there are several efforts toward the real Internet data collection, such as the Route Views Project [11], Rocket-Fuel Project [10], and skitter Project [2]. We choose to use the data from another project, "An Internet topology for network simulation" [4], hosted by Jason Liu. This project combines the datasets collected by the above multiple mapping tools, and results in a router-level U.S. network map with about 44 223 nodes and 68 681 (bidirectional) links.

This project also seeks to assign link attributes (i.e., bandwidth and delays) using information available from each ISP. There are five link types in this dataset; their assigned bandwidth capacities are: 45 M, 155 M, 622 M, 2.448 G, and 10 G. In reality, however, not all of the bandwidth is available. In our experiment, we assume only a small portion (0.5%) can be used by our videoconferencing application.

In addition to the router-level data, we also need the last-mile information of end hosts in order to conduct the simulation. From OECD (Organization for Economic Co-operation and Development) broadband statistics [7], we know that, in United States, there are 6.5 DSL subscribers, 9.0 Cable subscribers, and 1.3 other type of subscribers per 100 inhabitants. Therefore, we generate end host connection types and their uplink/downlink bandwidth according to Table I.

Given a fixed number of conference participants N, we generate 1000 test sets, each of which is created through the following procedure.

1) Randomly select N nodes (routers) from the original graph.
2) Attach an end host to each router. Host connection type and last-mile bandwidth is generated according to Table I.

TABLE I
Connection Type and Typical Uplink/Downlink Bandwidth

Connection Type:	DSL	Cable	LAN
Percentage:	39%	54%	7%
Uplink Bandwidth:	200-400K	700-900K	10M
Downlink Bandwidth:	600-1,200K	700-900K	10M

3) For each (host, host) pair, calculate the shortest path between them using the Dijkstra's algorithm, and record all the nodes and links on the path.
4) Trim off all the other nodes and links not recorded in previous steps.

B. Experiment Methodology

The IP-level topology is not directly used for simulation. The reason is that, in practice, conference members can only obtain application-level measurements, such as the round-trip time (RTT) and available bandwidth to a specific peer. Therefore, in order to simulate the real situation, we compute the end-to-end latency and bandwidth in each test set, and provide this information as the input to the ALM algorithm. In other words, the IP-level topology is transparent in the simulation.

We then execute the algorithm and work out one or multiple data delivery trees. Quality of the trees is evaluated along the following dimensions.

- *Total tree cost:* This metric is defined in [21]. For shared tree, it is the sum of delays on each link of the MST. For source-specific trees, in order for a fair comparison, it is defined as the average cost of all trees rooted at each member. Intuitively, this metric describes the time duration that an ALM delivery scheme occupies the network resource. Thus, this metric is also referred to as *network resource usage* in [17].
- *Average end-to-end delay:* This is one of the most important metrics that dominate user's experience. The end-to-end delay between host A and host B is calculated by summing up the link delay along the delivery path from A to B in the ALM tree. For an N-member conference, this metric is averaged among $N(N-1)$ end-to-end measurements.
- *Average throughput:* This metric evaluates how well a transmission scheme makes use of the bandwidth resource. Large throughput translates to high quality video. In order to measure this metric, we assume that: 1) videos are compressed by a scalable video codec, and the bit rate is between [0, 1000] Kbps; 2) All streams share the bandwidth fairly. For example, if two delivery paths share a physical link whose bandwidth is 1500 Kbps, then each of them gets the video transmitted at bit rate of 750 Kbps.

V. Experimental Results

A. Shared Tree versus Source-Specific Trees

The first experiment studies the performance tradeoffs between shared tree and source-specific trees (SST). Here, the shared tree is built as a MST based on the delay metric, while SST is constructed by shortest path first routing. Figs. 6–8 gives

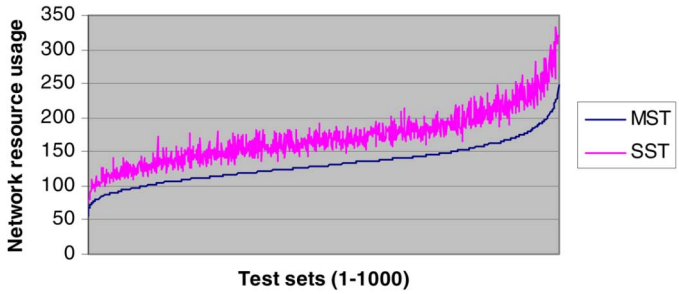


Fig. 6. Network resource usage of MST and SST.

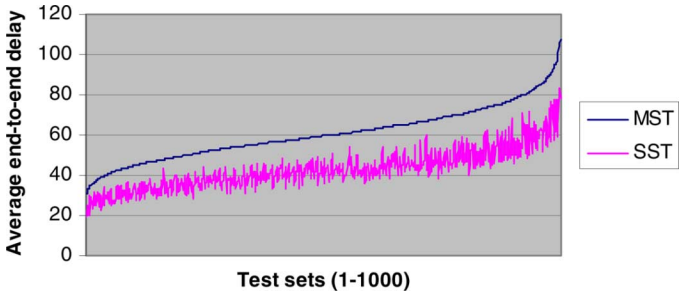


Fig. 7. Average end-to-end delay of MST and SST.

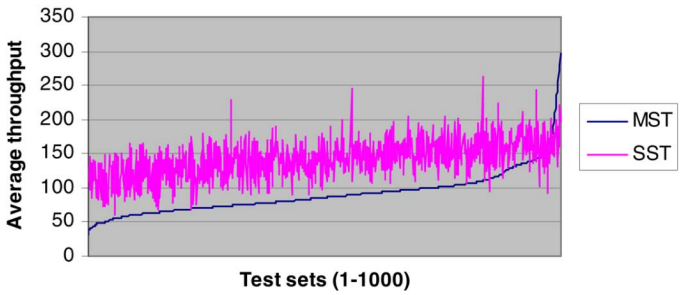


Fig. 8. Average throughput of MST and SST.

performance comparison on three evaluation metrics when conference size is five. Both algorithms are executed on 1000 test sets. The results are sorted in ascending order based on the MST performance. We keep the correspondence of results on the same test set, so the performance curve of SST is a bit oscillatory.

The results are consistent with our analysis in previous section. By definition, MST has the smallest total tree cost among all the application-level delivery schemes. Fig. 6 shows that MST has smaller tree cost than SST. On average, MST reduces the tree cost of SST by 22.3%. However, Figs. 7 and 8 show that MST doesn't have as good end-to-end performance as SST. On average, the throughput of SST is larger than that of MST by 66.7%, while the end-to-end delay of SST is 29.4% smaller.

Fig. 9 shows the comparison of MST and SST along different conference scales. For each experiment setting, we execute both algorithms on 1000 test sets. The data points shown on this figure is averaged among these test sets. We can see from the figure that, as conference size gets larger, SST gets more performance advantage over MST: when there are only three members in the conference, SST achieves 33.3% more throughput than MST, and the average end-to-end delay is 15.2% less; when conference size grows to ten, the throughput achieved by SST is 2.37 times of (137.2% more than) that achieved by MST.

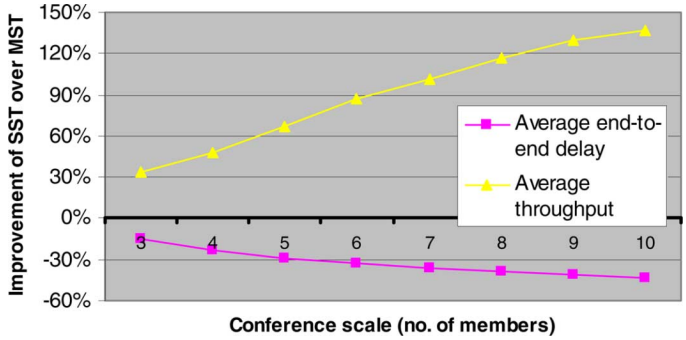


Fig. 9. Performance improvement of SST over MST on average end-to-end delay and throughput.

B. Multicast Tree Construction

We try to answer the following two questions in multicast tree construction through experimentation.

- Which tree construction framework is better, using DV protocol or using a centralized algorithm?
- In the framework of using DV protocol, does the *shortest widest path* routing achieve better performance than *widest shortest path* routing?

One advantage of a centralized algorithm over the DV protocol is that we can explicitly put on the bandwidth constraints, and keep track of bandwidth usage. We are interested in how this explicit bandwidth control could improve ALM algorithm performance. Therefore, we compare the DV protocol (based on shortest path first routing) with a simple centralized tree computation algorithm. The centralized algorithm constructs an ALM tree step by step using the following procedure: for a viewing request from member r to member s, s checks which member in its multicast tree has enough bandwidth to r. If such member exists, s will ask one of them (possibly s itself) to relay data for r. Usually, the member who brings the smallest delay to r is selected. If no member has available bandwidth, this request will be rejected. After the relay node is decided, s will update its bandwidth record, and process the next viewing request based on updated bandwidth information. Our experiment shows that such a simple centralized algorithm can achieve much higher throughput than the DV protocol.

Here, we use another metric *Number of satisfied requests* to evaluate the throughput performance. We fix the stream bit rate before executing both algorithms. Then, for the centralized algorithm, we count how many viewing requests are accepted. For the DV protocol, we count how many streams can be transmitted end-to-end without serious quality degradation. Specifically, we fix the bit rate at 100 Kbps, after executing the DV protocol, we count how many end-to-end paths have 100 Kbps throughput or larger.

Fig. 10 shows the comparison between the two strategies. In this set of experiment, the conference size is five. In a typical scenario, every member wants to see all the other's video, then there are 20 viewing requests (5×4) in total. We can see that, by using the centralized algorithm, in more than 75% of the 1 000 test sets, all the viewing requests are accepted. In the worst case of these test sets, more than half of the viewing requests (ten) are accepted. In contrast, if we use the DV protocol, less

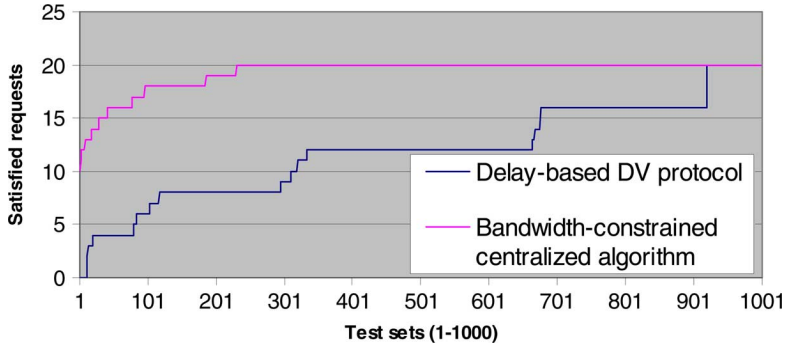


Fig. 10. Number of satisfied requests by two strategies.

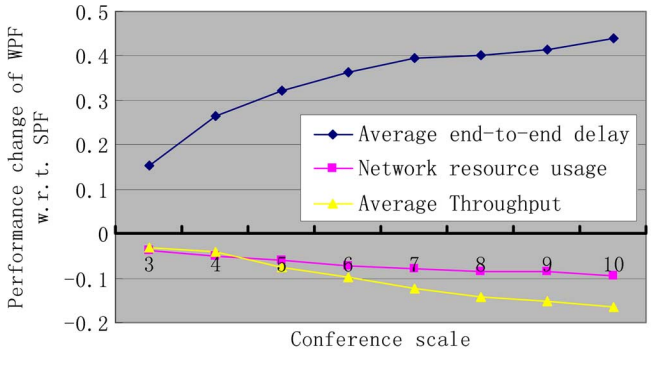


Fig. 11. Performance change of WPF w.r.t. SPF.

than 10% of the test sets can accept all the viewing requests. Moreover, only 65% of the test cases can accept 12 requests or more. Twelve requests (4×3) can translate to a four-member conference. Therefore, we can say that under the same network condition, a simple centralized greedy algorithm can support a five-member conference with a higher satisfaction rate than using DV protocol to support a four-member conference.

The second question interests us because in [16] the authors discussed the problem of dealing with dual metrics in multicast routing. In [16], the authors selected the *shortest widest path* in the DV protocol: each member tries to pick the widest path to every other member. If there are multiple paths with the same bandwidth, the member picks the shortest one. By using this metric, the authors prioritize the bandwidth metric over the delay metric, so we can call it widest path first (WPF) routing. Intuitively, WPF should achieve better throughput performance than *shortest path first* (SPF) routing. Very interestingly, we found that using WPF doesn't result in higher throughput. In most test cases, its average throughput is even slightly worse than using SPF routing.

Fig. 11 compares WPF with SPF along the three evaluation metrics on variant conference scales. Again, each data point is the average value of 1 000 test sets. It's not surprising that SPF has better end-to-end delay performance than WPF. When conference size is five, the average delay achieved by using WPF is 32.3% longer than that achieved by SPF. When conference size is ten, this number goes up to 44.0%. What surprises us is that the average throughput of WPF is smaller than that of SPF. By carefully examining the test cases and the resulting multicast trees, we find that WPF tends to use a smaller set of transmission links. In other words, the links used by SPF are more diverse, so that traffic loads are better dispersed. This also explains why WPF uses less network resource than SPF. Based on these results, we could answer the second question and conclude that SPF is a better choice than WPF under the framework of using DV protocol.

C. Performance Enhancement by Incorporating IP Multicast

We propose to use IP multicast whenever it is available. From the theoretical analysis in the previous session, we know that our proposed algorithm should consume less network resource and produce larger throughput than the algorithm without making use of IP multicast. However, evaluating our algorithm is difficult because it depends too highly on how conference members are located and clustered. In the extreme case, all the members are connecting to different ISP (Internet Service Provider), then our algorithm will not result in any performance improvement.

In order to provide a clear idea of how IP multicast could improve the performance of an ALM algorithm, we conduct experimentation in a five-member conference. There are seven different clustering combinations, one of which is that five members are with one ISP. Another combination is that four members are with one ISP and the fifth in the other; this is denoted as 4-1. Similarly, other combinations are denoted as 3-2, 3-1-1, 2-2-1, 2-1-1-1, and 1-1-1-1-1. In this experiment, we pass over the two extreme combinations (5 and 1-1-1-1-1) because they do not provide much information.

For the other five clustering combinations, we generate 1000 test sets for each. The data preparation procedure is a little different from the one described earlier:

1) Select M nodes (routers) from the Internet topology, where M is the number of clusters.
2) Compute the shortest paths between these M nodes, and trim the unrelated nodes and links.
3) Attach intended number of end hosts to each routers.
4) Add IP multicast channel in each cluster; the typical channel attribute is 10 Mbps bandwidth and 1ms delay.

Then we execute the tree construction algorithm with and without IP multicast. Fig. 12 shows the performance comparison on resource usage and average throughput. The average end-to-end delay of the two algorithm is almost identical. We see from the figure that, under all the five clustering settings, the algorithm with IP multicast (AwIPM) uses less network resource and achieves higher throughput than the algorithm

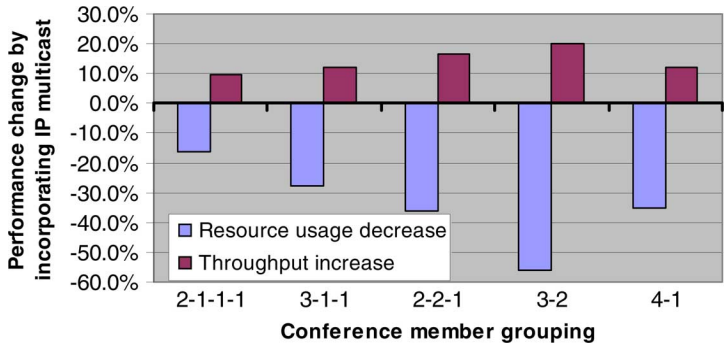


Fig. 12. Performance comparison betweeen the algorithm with and without IP multicast on resource usage and average throughput.

without IP multicast (AwoIPM). The cluster combination (3,2) achieves the best performance among others, where AwIPM uses 55% less network resource than AwoIPM and meanwhile achieves more than 20% average throughput.

Incorporating IP multicast into a centralized bandwidth-constrained algorithm can achieve similar performance. This supports our proposal to use IP multicast whenever it is available.

VI. Conclusion

A. Summary

With emerging broadband technology, desktop videoconferencing is striding into the mainstream, and an increasing need for multiparty videoconferencing exists. The main challenge in designing such a system is to accommodate the heterogeneity among Internet users. This heterogeneity exists in the types of Internet connections (direct connection to the Internet or behind firewalls and NAT), in the amount of governable network resources (e.g., available bandwidth), and in the requirements of software features.

We have solved the connectivity problem between different types of the Internet users by employing a server-aided conferencing architecture. A server is placed in a public Internet domain, and acts as middleman for multiple NAT-ed users. Such an architecture can improve the connectivity by a large degree. A more important contribution of our work is the five-layer design and implementation of the client application. This layered design makes a clear-cut distinction between different functional sets, and allows us to define a suite of interfaces between related modules. Therefore, we are able to provide rich flexibility not only for the implementation of functional modules but also for the feature set that we offer. Specifically, our system can transmit data over TCP, or UDP, or IP multicast; it allows multi-Unicast transmission or ALM transmission; it provides the feature of attending multiple simultaneous sessions and the ability to switch between them.

Observing that the high bandwidth and low-latency requirements of videoconferencing applications do not tolerate the heterogeneity of user network resources, we have designed an ALM solution for multisender real-time communications. We have addressed three important tradeoffs in the design of an ALM scheme, and have made our own choices through extensive experimentation. We have found that 1) source-specific trees have better end-to-end performance than shared tree, and as conference size grows larger, the advantage becomes more notable; 2) distributed DV protocol cannot make use of bandwidth resource as intelligently as a centralized bandwidth-constrained tree construction algorithm, and in the framework of using DV protocol, shortest path first routing generally has better performance than widest path first routing; 3) incorporating IP multicast into an ALM scheme can greatly improve end-to-end performance and network resource usage. These findings provide important guidance in our system design.

B. Discussion

The described system has been implemented and is now available for trial inside our company. We have received many valuable feedbacks from users as well as other colleagues who are interested in this application. Next we discuss a few directions for future work which are enlightened by these feedbacks:

1) Allowing Non-Participant Relay: In the current design, we limit data relay within conference participants. We consider that only participants have natural motivation to relay data for others. However, this design does not fully exploit network resources and is insufficient when:

1) a majority of peers are behind NAT. Our system employs a server in the public domain to facilitate UDP hole punching between peers. However, if a majority of peers in a conference are behind symmetric NAT or cannot establish direct connections because of firewall settings, a large portion of data traffic will be relying on a small number of participants;
2) all participants have limited bandwidth resource. When this rare situation happens, our current system cannot do much except that current participants invite a broad bandwidth user to join the conference.

These situations can be tackled if we allow nonparticipant relay. Nonparticipant includes dedicated media servers or selected super peers. Employing dedicated media servers is simple in architecture, but needs initial investment. Utilizing super peers is cost effective, however, it is very challenging to select the right super peers to keep the transmission delay and total system cost at a reasonable level. We believe this challenging topic worths further study.

2) Automatic Video Quality Adaptation: Our system allows users to choose a few video compression settings, such as bit rate, frame rate, and key frame interval. It even allows users to

adjust these settings when the conference is under way. However, we find that most users do not touch these settings, and they almost never change them during a conference session.

While we believe that video quality adaptation helps to cope with network dynamics and heterogeneity, we did not incorporate an automatic strategy in the current system. Consider such a scenario: a DSL user, who has 200 Kbps uplink bandwidth and 400 Kbps downlink bandwidth, is having a conference with four LAN users. Video bit rates of the DSL user and LAN users are 128 Kbps and 512 Kbps (using nonscalable codec). With our ALM routing, the DSL user can distribute his video to all the others through peer relay. However, he cannot see any others' videos because of downlink bandwidth limitation. In such a case, shall we automatically decrease the LAN users' video bit rates so that the DSL user can see all of them, or we just trade the bad experience of the DSL user for high quality videos that other LAN users can enjoy? More user studies need to be carried out in order to make such a decision, or scalable video codec needs to be used. Incorporating automatic video quality adaptation and scalable video codec will be one of future directions.

3) Reducing Less Frequently Used Features: In telephone system, people are able to handle multiple connections by holding and resuming operations. We provide a similar service in our videoconferencing system: a user can put an ongoing session on hold if he receives an incoming invitation, and he can switch between multiple simultaneous sessions just like what people can do in telephone systems. However, our users reported that they seldom used this feature. We think the reason is that people use videoconferencing at a much lower frequency than they use telephone, and most of the conferences are scheduled. We are thinking of reducing such less frequently used features in future versions of our system.

In summary, we have built a multiparty videoconferencing system based on peer-to-peer technologies, and have tried to provide good user experiences despite of network heterogeneities. We believe that our work can be a helpful reference to other efforts towards the same goal.

REFERENCES

[1] [Online]. Available: http:www.aim.com

[2] CAIDA skitter Tool [Online]. Available: http:www.caida.org/tools/measurement/skitter/

[3] ICUII Live Video Chat Program [Online]. Available: http:www.icuii.com

[4] An Internet Topology for Simulation [Online]. Available: http:www.crhc.uiuc.edu/jasonliu/projects/topo/

[5] iSpQ Video Chat and Webcam Chat Community [Online]. Available: http:www.ispq.com

[6] Kazaa File Sharing [Online]. Available: http:www.kazaa.com

[7] OECD Broadband Statistics [Online]. Available: http:www.oecd.org/

[8] PalTalk Live Online Chat Community [Online]. Available: http:www.paltalk.com

[9] Raindance Web Conferencing, Web Meeting, and Internet Video Conferencing [Online]. Available: http:www.raindance.com

[10] Rocketfuel: AN ISP Topology Mapping Engine. [Online]. Available: http:www.cs.washington.edu/research/networking/rocketfuel/

[11] Route Views Project [Online]. Available: http:www.routeviews.org/

[12] Skype Interent Calls [Online]. Available: http:www.skype.com

[13] WebEx Web Conferencing, Online Meetings, and Video Conferencing [Online]. Available: http:www.webex.com

[14] Windows Live Messenger [Online]. Available: http:messenger.msn.com

[15] Yahoo Messenger [Online]. Available: http:messenger.yahoo.com

[16] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay muilticast architecture," in *Proc. ACM SIGCOMM 2001 Conf. Applications, Technologies, Architectures, and Protocols for Computer Communications*, Aug. 2001, vol. 31, pp. 55–67.

[17] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM SIGMETRICS 2000 Conf. Measurement and Modeling of Computer systems*, Jun. 2000, vol. 28, pp. 1–12.

[18] A. Ganjam and H. Zhang, "Connectivity restrictions in overlay multicast," in *Proc. 14th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, Jun. 2004, pp. 54–59.

[19] M. Hosseini and N. Georganas, "Design of a multi-sender 3D videoconferencing application over an end system multicast protocol," in *Proc. Eleventh ACM Int. Conf. Multimedia*, Nov. 2003, pp. 480–489.

[20] C. Luo, J. Li, and S. Li, "Digimetro—An application-level multicast system for multi-party video conferencing," in *Proc. IEEE Global Telecommunications Conf. (GlobeCom)*, Nov. 2004, vol. 2, pp. 982–987.

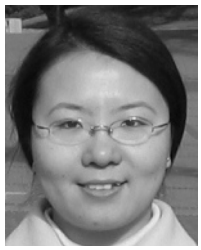
[21] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "Almi: An application level multicast infrastructure," in *Proc. 3rd Usenix Symp. Internet Technologies and Systems (USITS)*, Mar. 2001.

[22] J. Rosenberg, Interactive Connectivity Establishment (ICE): A Methodology For Network Address Translator (NAT) Traversal for the Session Initiation Protocol (SIP) July 2004, work in progress..

[23] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy, STUN—Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs) Mar. 2003 [Online]. Available: http:www.faqs.org/rfcs/rfc3489.html

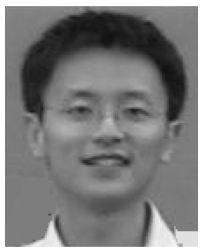
[24] L. Wei and D. Estrin, "The trade-offs of multicast trees and algorithms," in *Int. Conf. Computer and Communication Networks*, Sept. 1994.

[25] M. Zhang, C. Luo, and J. Li, "Estimating available bandwidth with multiple overloading streams," in *Proc. IEEE Int. Conf. Communications (ICC)*, June 2006.



Chong Luo (M'05) received the B.S. degree in computer science from Fudan University, Shanghai, China, in 2000 and the M.Sc. degree in computer science from National University of Singapore, Singapore, in 2002. She is currently pursuing the Ph.D. degree in the Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China.

She is an Associate Researcher with Microsoft Research Asia, Beijing, China. Her research interests include distributed multimedia systems, peer-to-peer networks, and wireless sensor networks.



Wei Wang received the B.S. degree in electronic engineering from Sichuan University, Chengdu, China, in 2003. He is currently pursuing the Ph.D. degree in the School of Electronic Engineering, Sichuan University, Chengdu, Cnina.

His research interests include networking and security.

Jian Tang (M'05) received his B.S., M.S., and Ph.D. degrees from Tsinghua University, Beijing, China in 1999, 2002 and 2005, respectively.

He is now with System Research Group at Microsoft Research Asia, Beijing, as a post-doctoral Associate Researcher. His research interests include networking, peer-to-peer, system software and media streaming.

Jun Sun (M'03) received the B.S. and M.S. degrees from the University of Electronic Science and Technology of China and Shanghai University, China, in 1989 and 1992 respectively, and Ph.D. degree from Shanghai Jiao Tong University (SJTU), Shanghai, China, in 1995.

He is now Professor with the Institute of Image Communication and Information Processing, SJTU. His general interests include DTV coding and video communication.

Jiang Li (SM'04) received the B.S. degrees in both applied physics and applied mathematics from Tsinghua University, China, in 1989, the M.S. degree in optics from Physics Department, Zhejiang University in 1992, and the Ph.D. degree in applied mathematics from the State Key Laboratory of Computer Aided Design and Computer Graphics, Zhejiang University, China, in 1998.

He joined the faculty of Zhejiang University in 1992.He is currently Research Manager of the Media Communication Group, Microsoft Research Asia, Beijing, China. He invented bilevel video, portrait video, and watercolor video, which are suitable to very low bandwidth network. He released Microsoft Portrait, the first video communication software prototype on Pocket PC and Smartphone. He is now leading Media Communication Group on research of mobile video communication, multiparty conferencing, multiview video, and peer-to-peer streaming.

Before joining Microsoft, Dr. Li was Associate Professor at Physics Department, Zhejiang University.