Available at
www.ComputerScienceWeb.com
POWERED BY SCIENCE @ DIRECT®

Computer Networks 41 (2003) 363–383

COMPUTER NETWORKS

www.elsevier.com/locate/comnet

LVMSR: an efficient algorithm to multicast layered video

Wushao Wen [a,*,1], Biswanath Mukherjee [b], S.-H. Gary Chan [c], Dipak Ghosal [b]

[a] CIENA Corporation, 10480 Ridgeview Ct., Cupertino, CA 95014, USA
[b] Department of Computer Science, University of California, Davis, CA 95616, USA
[c] Department of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Received 15 February 2001; received in revised form 17 May 2002; accepted 27 September 2002

Responsible Editor: G. Pacifici

Abstract

Layered video is a video-compression technique to encode video data in multiple layers. It typically consists of a base layer and some additional layers that provide enhanced video quality. The multicasting operation of layered video consists of many receivers dynamically joining and leaving different multicast sessions of different layers depending on their network condition. A layered video multicasting system needs to satisfy: (i) bounded end-to-end delay from the video source to each receiver; (ii) minimum total cost; and (iii) minimum delay jitter between the various video streams received by each receiver. The problem of computing such data distribution paths is NP-complete. This paper presents a new heuristic algorithm, called layered video multicast super-tree routing algorithm, with $O(Rn^2)$ time complexity and $O(R^2)$ message complexity, where n is the number of nodes in the network and R is the receiver group size. Our investigation shows that the multicast data paths computed by our algorithm can always satisfy the delay constraint with reasonably low total cost.

Keywords: Layered video; Multicast; Multicast routing algorithm; Delay constraint; Optimization; Quality of service; Mesh network

1. Introduction

Advances in networking and server technologies have made the provisioning of on-demand video services to homes a reality [2,8,11,19]. Video-on-demand (VoD) systems can be divided into two types—the pure video-on-demand (pure VoD) system and the near video-on-demand (near VoD) system. In a pure VoD system, each user is assigned its own dedicated unicast channel from the server. However, a pure VoD system becomes expensive when a large number of concurrent users have to be accommodated. When a video is very popular, using a single multicast channel (i.e., a multicast stream) to serve many users simultaneously offers a more cost-effective solution. A system providing video services via multicast is a near VoD system.

In traditional video multicast, a single source transmits a single video stream to multiple receivers

[*] Corresponding author. Tel.: +1-408-366-4956; fax: +1-408-366-4867.

E-mail addresses: wswen@ece.ucdavis.edu (W. Wen), mukherje@cs.ucdavis.edu (B. Mukherjee), gchan@cs.ust.hk (S.-H. Gary Chan), ghosal@cs.ucdavis.edu (D. Ghosal).

[1] This work was done while Dr. Wen was a Ph.D. student at UC Davis. He is now with CIENA Corp.

via a dedicated transmission network. This scheme works well in a dedicated video-transmission network where all users can access the network with the same bandwidth and quality requirement. However, some video applications may not be able to use such networks, e.g., some interactive video applications which are distributed widely, but only occasionally. Such video applications may use general data networks. However, the data networks' heterogeneity and size make multicast communication a difficult problem in wide-area mesh networks. In such networks, different users can access the network via different bandwidth interfaces. Some users can only access the networks via low-speed connection while others can use relatively high-bandwidth connections. For example, ISDN users may access the network at 128 kbps, digital subscriber line (DSL) users may access the network at 256 kbps, cable modem users may access the network at 1 Mbps, and local-area-network (LAN) users may access the network via higher-speed connections such as Ethernet, which provides 10 or 100 Mbps interfaces. If a video is encoded in a single stream and then is multicast at some fixed rate without considering the user heterogeneity, the VoD system cannot satisfy the different requirements of different users at the same time. To provide appropriate service to all users, only the minimum access bandwidth can be used to encode the video, or multiple video streams with different quality must be used. Furthermore, a video stream requires high bandwidth. If all data is routed in a single stream, some users may not be able to access the service because they do not have enough bandwidth along any single data-path although they may have enough bandwidth via multiple paths with load balancing.

Layered video is the solution to satisfy the user and network heterogeneity. In layered-video encoding/decoding [13,15,18], video data is encoded into a number of layers that can be sequentially combined to provide progressive quality refinement of the received video. In case of insufficient network bandwidth, the network can decide to drop some higher-layer data for some sets of users, or some receivers can choose not to receive some higher-layer data. In these cases, users can still receive the data of the lower layers, which provide reduced quality but continuous video. This is much better than using a single stream in which nothing is played for a frame if any part of its data is not delivered correctly. The layered-video scheme also relieves servers from providing multiple streams for users of different classes, which consume too much network bandwidth and router processing capacity unnecessarily. Therefore, layered-video multicast is an elegant solution to the heterogeneity of networks and users. However, there are some associated overheads. In particular, a receiver must have additional processing capacity for decoding video data since the decoding is based on multiple layers of video data. Furthermore, it may be also required that the data buffer size be also increased to ensure that data from different layers is used to decode synchronously, since data for different layers may be transmitted via different paths, resulting in different delays.

An efficient algorithm to construct multicast data-paths is critical to make the layered-video-multicast scheme successful. The algorithm should be able to minimize the total cost of the multicast data-path, transmit data to all destinations in bounded delay, and minimize the delay jitter of different layers at the receivers. We refer to this problem as the layered-data, bounded-delay, and minimum-cost problem. As we know, the least-cost multicast tree is called a Steiner tree [3]. The problem of finding a Steiner tree is NP-complete [6]. It is clear that the construction of minimum-cost, layered-transmission data-paths with delay constrainst is also a NP-complete problem. In this investigation, we present a new heuristic algorithm, called the *layered-video multicast super-tree routing* (*LVMSR*) algorithm, for multicasting layered video. LVMSR is a distributed algorithm which always satisfies the delay and bandwidth constraints while trying to reduce the total cost. Because this problem is NP-complete and because LVMSR is a heuristic, we remark that data distribution via our constructed multicast data-path may have sub-optimal total cost and sub-optimal delay jitter.

This paper is organized as follows. Section 2 summaries the related work. Section 3 describes our system model and defines the problem. Some keywords are also defined in this section. Section 4 describes our LVMSR heuristic algorithm. Section 5 is devoted to an analysis of various interesting properties of LVMSR. Section 6 presents our simulation model to evaluate LVMSR and some illustrative numerical results. Section 7 concludes the paper.

2. Related work

Multicast communication is a topic of intense research [5,7,9,14–17,20,21]. The "Internet Multicast Backbone", or MBone, has risen from a small, research curiosity to a large-scale communications infrastructure. Layered-video encoding [13,15,18] and distribution [4,9,10,12,15–17,20] are attracting more and more attention. Our study focuses on layered-video distribution over a wide-area mesh network.

Layered-video multicast has been studied by some researchers. Maxemchuk [12] studied the problem of designing multicast networks for video distribution, and developed a heuristic to solve the problem. His proposed heuristic was related to and compared with previous heuristics developed for the Steiner-tree problem and algorithms to design minimum-depth and minimum spanning trees. McCanne et al. [13] developed a receiver-driven layered-video multicast (RLM) protocol for rate-adaptive transmissions. In RLM, a source distributes a hierarchical signal by striping the different layers across multiple multicast groups, and the receivers adjust their reception rate by simply joining and leaving multicast groups. Wang and Hou [20] designed a multicast routing algorithm for distributing layered multicast video to heterogeneous receivers in networks with rate-based link schedulers. The multicast tree constructed by their algorithm fulfills the quality-of-service (QoS) requirements imposed by heterogeneous receivers, both in terms of bandwidth and delay, and simultaneously consumes as little network resources as possible. The proposed algorithm is decentralized and is adaptive to network and membership changes.

However, most previous research does not take the correlation of the multicast sessions of different layers into consideration. Hence, we investigate a QoS-based layered-video multicast routing algorithm, which takes into consideration the correlation of multicast data-paths of different layers. The multicast data-paths constructed by our algorithm attempt to satisfy, if possible, the QoS requirements, e.g., the delay constraint.

3. Layered-video representation and distribution

Let us denote the number of layers generated by a layered-video encoder as L. Layer 1 is the base layer. Any lower l ($l \leqslant L$) layers combined together can be used to represent the original video with a certain quality. The $(l+1)$th ($l < L$) layer provides enhancements to the video that is represented by the lower l layers. The $(l+1)$th layer is valid only if all the lower l layers are valid. If any data of layer j ($j \leqslant l$) is missed, then the $(l+1)$th layer is also invalid for decoding. In our study, we make no assumption on how the original video is compressed and encoded; we only assume that the video is represented in increasingly refined layers. In practice, the base layer provides basic data for those users with minimum available bandwidth. Every additional layer combined with the lower layers can meet the requirements of an additional class of users.

3.1. Network model

We model the network by a connected graph $G(V,E)$. The node set, V, represents routers or switches in the network and the data link set, E, represents the communication links between nodes. There are three parameters corresponding to each communication link e, $e \in E$, as follows:

- $d(e)$ is the delay of link e. It includes the propagation delay, queuing delay, and transmission delay. In this study, we assume that every link's delay is known.
- $c(e)$ is the cost of link e. It represents the charge incurred for using the link, measured in terms of dollars per megabit ($/Mb), or dollars per unit bandwidth ($/Mbps).

- $b(e)$ is the available bandwidth on link e. It depends on the current network state and its value can vary at different times.

In this study, we consider a session with a single multicast source. Receivers may have different capacities (i.e., interfaces with different bandwidths); therefore, they may request different layers of video data. We also assume that the source maintains separate multicast sessions for different layers.

3.2. Problem specifications

This study is aimed at solving the QoS-based multicast routing problem for layered-video distribution. The problem is specified as follows.

For a given network represented by graph $G(V,E)$, its current network state, a video source, and a set of receivers, [2] select an optimum set of multicast-routing data-paths to distribute the layered video, rooted at the source, so that the total cost for the paths is minimum; at the same time, satisfy the end-to-end delay bound from the source to any receiver, minimize delay jitter of received data from different layers for each receiver, and satisfy the receiver request on the number of layers it asks for as much as possible.

The conventional problem to find a minimum-cost multicast tree (Steiner tree [3]) is NP-complete. Therefore, our problem is also NP-complete since the Steiner-tree problem is a subset of our problem. For an NP-complete problem, it is not guaranteed to find an optimal solution; hence, heuristic algorithm must be developed to solve the problem.

3.3. Definitions

To clarify our discussion, we define the following keywords:

- A *super-tree* $\mathrm{ST}(V,E)$ is a graph constructed by merging several trees that have the same root. The trees are merged to form the super-tree according to the following rules:
  1. The super-tree node set V is the union of the node sets of all trees. This can be represented by: $V = V_1 \cup V_2 \cup \cdots \cup V_m$, where V_i is the node set of tree i.
  2. The super-tree edge set is the union of the edge sets of all trees, which can be represent by: $E = E_1 \cup E_2 \cup \cdots \cup E_m$, where E_i is the edge set of tree i. In this definition, we assume that, in the resulting super-tree, edges that have the same end nodes are merged into one edge.
- A *weighted super-tree* $\mathrm{WST}(V,E)$ is a graph constructed in the same way as a super-tree except that all of the original trees are weighted trees and the resulting weight for a edge is the sum of the edge weights from all the trees forming the super-tree.
- A *super-ditree* $\mathrm{SDT}(V,E)$ is a graph constructed in the same way as a super-tree except that all of the original trees used to construct the super-tree are directed trees and the edges that have the same end nodes are merged into one edge only if their directions are the same. A *weighted super-ditree* $\mathrm{WSDT}(V,E)$ is a graph that is constructed from a group of weighted directed trees according to both policies of constructing super-ditree and weighted super-tree.
- A *delay-constrained minimum-cost* (*DCMC*) *path* from a tree T to a node v ($v \notin T$) is the least-cost path from any node in tree T to a node v which has delay less than a delay bound Δ from the root s of tree T to node v. We denote the corresponding path as $P(T,v)$, the cost for the path as $P_{\mathrm{C}}(T,v)$, and the delay on it as $P_{\mathrm{D}}(T,v)$.

Super-tree, super-ditree, weighted super-tree, and weighted super-ditree are special forms of graph that have some tree characteristics. [3]

[2] The receivers here are not actually the end-users. They are routers which take part in the multicast-routing-data-path calculation. They can be edge routers which are directly connected to end-users. Or they can be border-gateway routers which act as virtual receivers, providing proxy service to end-users.

[3] Note that simply joining multiple trees together may not form a tree.

3.4. Problem description

For a layered-video multicast, it is important to design an efficient multicast super-tree. The multicast super-tree is not exactly a tree because different layers of video data can travel through different paths to a destination. When we examine a specific layer of video-data routing paths, they form a tree. However, the routing paths of a set of streams (a layer of video data forms one stream) form a super-tree. A multicast super-tree must satisfy the following requirements:

- *Priority transmission:* The network gives higher priority to lower layers and always reserves necessary network resources to transmit data of lower layers before reserving any resources to transmit data of higher layers. Furthermore, a receiver can decide how many layers of data it would receive according to its own requirements or the available bandwidth. When the available end-to-end bandwidth is low, a higher-layer stream may not be delivered to some receivers even though they request it. We also assume that the network has some mechanisms (such as RSVP, RSVP-TE) to reserve network resources for data transmission.
- *Minimum total cost:* The total cost of the multicast super-tree is calculated as follows:

$$\text{Routing cost}: \ C = \sum_{e \in \mathrm{ST}} c(e), \tag{1}$$

where ST is the multicast super-tree and e is a data link on the super-tree ST.
- *Bounded delay:* The delay of a path $P(s,r)$ from source s to a receiver r ($r \in R$, R is the receiver set) should always satisfy a bounded value Δ, i.e.,

$$\forall r \in R, \quad \sum_{e \in P(s,r)} d(e) \leqslant \Delta. \tag{2}$$

- *Minimum delay jitter:* We define the delay jitter as the sum of different receivers' delay variance, which can be calculated as follows:

$$\sigma_{\mathrm{D}}^2 = \sum_{r=1}^{R} \sum_{l=1}^{L_r} (D_{rl} - \overline{D}_r)^2 \frac{B_l^2}{B_r^2}, \tag{3}$$

where D_{rl} is the delay for layer l at receiver r, B_r is the total bandwidth required by receiver r, B_l is the bandwidth requirement for layer l, and $\overline{D}_r$ is the average data delay at receiver r which is given by

$$\overline{D}_r = \sum_{l=1}^{L_r} D_{rl} \frac{B_l}{B_r}, \tag{4}$$

where L_r is the actual continuous lower layers received by receiver r and M_r is the total number of layers requested by receiver r. [4]
- *User satisfaction:* User satisfaction is measured by the fraction of the weight of the requested video layers by users and the weight of the received video layers. It is given by

$$S = \frac{\sum_{r=1}^{R} \sum_{l=1}^{L_r} B_r}{\sum_{r=1}^{R} \sum_{l=1}^{M_r} B_r}. \tag{5}$$

4. Layered-video multicast super-tree routing algorithm

Based on our above discussion, we propose an algorithm to solve the problem on how to select the layered-video multicast data-paths for a given network $G(V,E)$, the source, and the receivers. Our algorithm, called LVMSR, is used to compute a multicast super-tree, which can deliver as many layers of video data as possible to satisfy each receiver's QoS requirements. The data is distributed with minimum cost while simultaneously satisfying the bounded delay and bounded delay jitter requirements as constraints.

The LVMSR problem is an optimal routing-with-delay-constraint problem. The cost requirement often conflicts with the delay requirement. As a result, the super-tree with the minimum cost may have longer delays for some paths, whereas a minimum-delay super-tree may incur higher cost. A tradeoff must be reached to satisfy the specific requirements. Our algorithm is a heuristic which

[4] M_r can be less than L because some users may choose to subscribe to only several lower layers which they believe are important.

adopts some concepts of the Steiner-tree algorithm. While the Steiner-tree algorithm is used only to compute a multicast tree, our algorithm constructs a multicast super-tree which delivers more than one layer of data to the destinations. Furthermore, the construction of the multicast tree of different layers cannot be made independently because there are correlations between different layers.

In the following discussion, we assume that each node in the network has the information necessary to construct the constraint-based lowest-cost path from itself to any other node in the network. LVMSR is a source-based distributed routing algorithm. The basic idea of LVMSR is that it first constructs the delay-constrained minimum-cost multicast tree for the base layer (layer 1), denoted as T_1. In layered-video multicast, all receivers which request for the same video in a multicast session should subscribe to at least the base layer; hence, this multicast tree includes the source node and all the receivers. Computing such a tree is the standard conditional Steiner-tree problem and there exist efficient heuristics to solve this problem [5,14].

After the multicast tree of the base layer is constructed, the algorithm then constructs the multicast tree for the second layer. The multicast trees for layer 3, layer 4, ..., and layer L are constructed in sequence. The two major components of this algorithm are how to construct a single-layer, delay-constrained, minimum-cost multicast tree, and how to take the correlation requirement of different layers into consideration in deciding on the multicast trees of different layers, i.e., how to take the lower-layer multicast trees into consideration when constructing a higher-layer multicast tree. If the multicast trees for different layers are constructed independently, the delay jitter for different layers might be high, which may incur coordination difficulty and additional resource requirements.

The key steps of the heuristic algorithm are described below.

4.1. The LVMSR algorithm

- *Step 1:* Initialize the multicast super-tree ST which only includes the source node S. Set the current layer $l = 1$, which is the base layer.
- *Step 2:* Mark the tree rooted at source node from the current ST in which each link in the tree has enough bandwidth to transmit layer-l data. If there is more than one path from the source to any node in ST, the lowest-cost path is selected. This is the initial multicast tree T_l for layer-l. Mark the delay for each node in the initial T_l. The source node then sends the information on the initial multicast tree T_l to all the receivers which are not on the tree.
- *Step 3:* Every layer-l's receiver which is not on tree T_l calculates its delay-constrained, lowest-cost path from tree T_l to itself separately and sends this information to the source. The delay-constrained, lowest-cost path is calculated in a simple manner. A receiver calculates the lowest-cost path and the minimum-delay path from every node in tree T_l to itself. Then, every receiver selects the lowest-cost path under the delay constraint from all the paths calculated by the above two methods. [5] If no constrained lowest-cost path exists, the receiver will notify the source and unsubscribe to the video session for the current layer and for all higher layers.
- *Step 4:* The source selects the layer-l receiver which has the minimum-cost path under the delay constraint to tree T_l, and adds the receiver and its corresponding delay-constrained lowest-cost path to T_l. If any node in the selected delay-constrained lowest-cost path is already in tree T_l, [6] the node's parent is changed to the new parent in the newly selected, delay-constrained lowest-cost path. The delay information for every node in the tree is then updated.
- *Step 5:* If all the receivers for layer l are not in the tree, go back to *Step 3*. Otherwise, T_l is constructed successfully. Prune all the edges and nodes that are not leading to any layer-l receiver

[5] This is sub-optimal in finding the constrainted lowest-cost path. We can improve the algorithm by selecting several shortest-delay or minimum-cost paths [1], but our algorithm only uses a single shortest path for either minimum cost or delay. This algorithm guarantees that the delay constraint is always satisfied, if possible.

[6] This implies that there is a better path selected for that node previously if the new receiver is not considered.

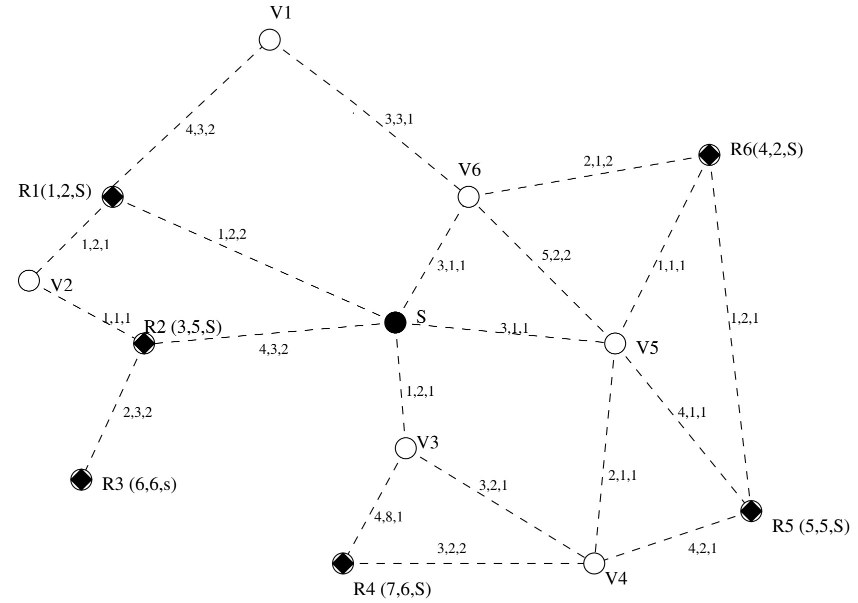

Fig. 1. Example network for LVMSR illustration: S = source; R_j = destinations. Note that the original multicast tree for layer 1 shown here only includes the source node S.

and the resulting tree is the multicast tree T_l for layer l.

- *Step 6:* Merge T_l with old ST, which results in the new super-tree ST.
- *Step 7:* If $l < L$, then $l = l + 1$ and go to *Step 2*; else ST is the required multicast super-tree. Stop.

4.2. Illustrative examples

We use the network shown in Fig. 1 to illustrate our algorithm by constructing a two-layer multicast super-tree. We assume that the required delay constraint Δ from the source to any destination should always be satisfied, with $\Delta \leqslant 6$ in this illustration. In Fig. 1, every link is marked with three parameters (x, y, z), which represent cost, delay, and available bandwidth of the link, respectively. In this example, for simplicity of exposition, we assume that the links are symmetric.

The algorithm first constructs the layer-1 multicast tree. It starts with an initial multicast tree T_1 which only includes the source node S. Every receiver calculates its DCMC path from tree T_1 to itself and sends this information to the source. This step is shown in Fig. 1, where the dashed lines represent data links. Each receiver which is not in the multicast tree is marked with three parameters: *cost, delay, and fork-node* [7] *of the DCMC path from the tree to itself.* For example, $R1(1,2,S)$ means that there exists a DCMC path from tree T_1 to receiver $R1$ at fork-node S. The cost of this path is $P_C(T_1, R1) = 1$ and the delay from source S to receiver $R1$ is $P_C(T_1, R1) = 2$.

After the source receives the information on the DCMC paths from all the receivers, it selects the receiver $R1$ which has the minimum cost among all receivers' DCMC paths from the current multicast tree T_1, and its corresponding DCMC path is added into the multicast tree T_1. Then, the source sends its new multicast-tree information to all receivers which are not in the current multicast tree T_1. These receivers then update their DCMC paths after receiving the new multicast tree accordingly,

[7] A fork-node is the connection node that is the beginning node for a newly selected receiver's DCMC path. It is on the original multicast tree T_l.

and send the updated paths' information to the source. Fig. 2 shows the network state after $R1$ is selected, its DCMC path is included in the tree, and all other receivers finish updating their paths. Fig. 2 shows that $R2$'s DCMC path's fork-node is changed from S to $R1$. And the cost from the current multicast tree to $R2$ is 2.

Following the same procedure, $R2$, $R6$, $R5$, and $R4$ are selected into tree T_1, one by one in each subsequent step. Fig. 3 shows the network state after $R2$, $R6$, $R5$, and $R4$ are included in the multicast tree. In each step, the source always selects the receiver which has a path with the minimum cost under the delay constraint to the current multicast tree and includes its corresponding path in the multicast tree T_1. The DCMC paths of those receivers which are not on the multicast tree are updated accordingly after the new path is added to the multicast tree.

Fig. 4 shows the result after receiver $R3$ is added into the multicast tree T_1. We notice that $R2$ is originally connected to $V2$, but if $R3$ follows the data-path via $R2$'s data-path, then the delay from the source to $R3$ is 7, which is greater than the required delay bound $\Delta = 6$. Therefore, $R3$'s path is S–$R2$–$R3$. $R2$ has already been included in the multicast tree T_1 before; hence, its data-path needs to be rerouted. Its parent node is changed from $V2$ to S. The data link $V2$–$R2$ is no longer useful and thus is excluded from the multicast tree.

After the algorithm successfully adds $R3$, now all receivers are in the multicast tree T_1. Note that this algorithm may encounter rerouting situations while setting up the multicast tree (as illustrated in the previous paragraph); as a result, some of the data links and nodes in the multicast tree may not belong to any of the receivers' data-path at all. For example, in Fig. 4, node $V2$ and link $R1$–$V2$ are not on any receivers' data-path now. In order to reduce unnecessary resource consumption, such nodes and links should be pruned from the multicast tree. Fig. 5 shows the resulting multicast tree T_1 (drawn in thick lines) for layer-1 after pruning, with node $V2$ and link $R1$–$V2$ eliminated. All leaves of T_1 are now layer-1 receivers. And the tree's root is the source node S.

Upon the completion of T_1, the algorithm begins to construct the layer-2 multicast tree T_2. The algorithm first identifies the initial multicast tree for layer-2. The initial layer-2 multicast tree is the sub-

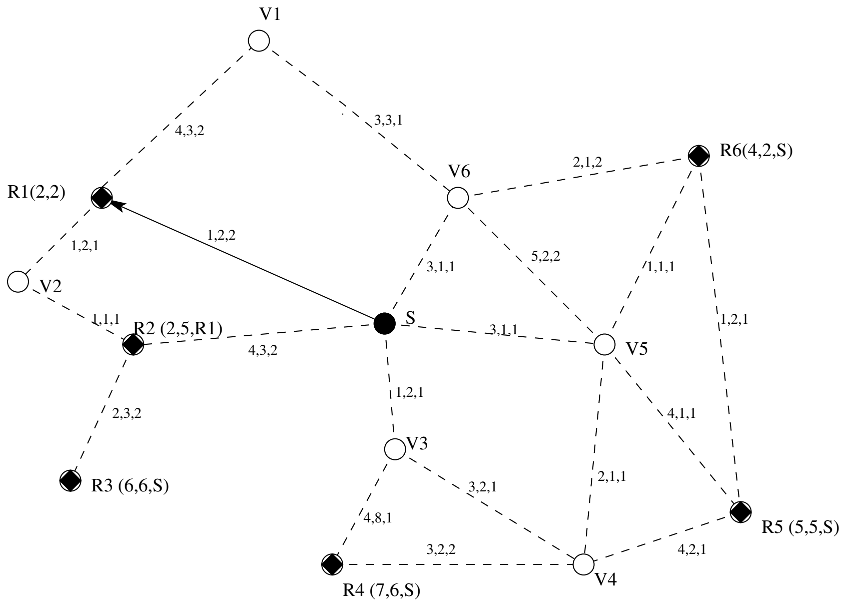

Fig. 2. Layer-1 multicast tree after $R1$ is selected.

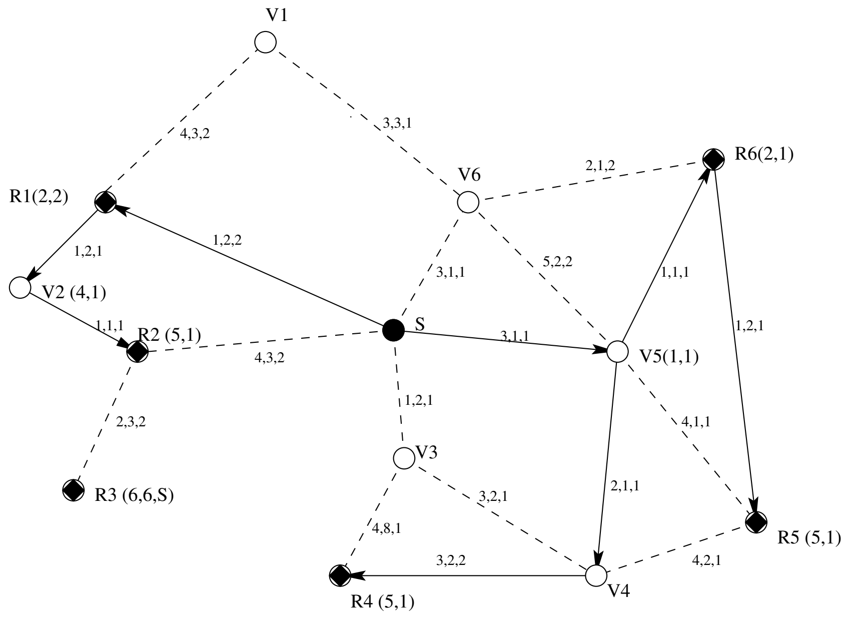

Fig. 3. Layer-1 multicast tree after $R1$, $R2$, $R6$, $R5$, and $R4$ are selected.

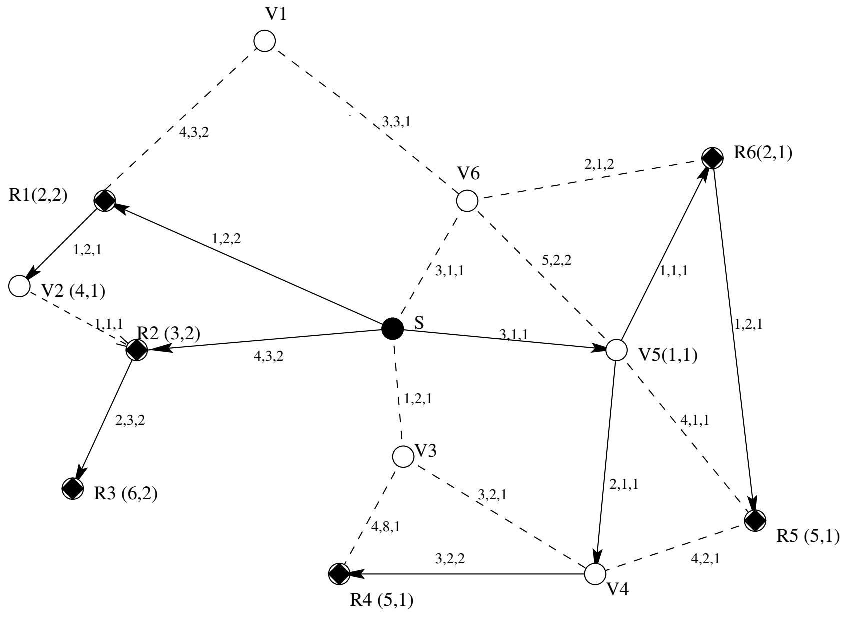

Fig. 4. Layer-1 multicast tree after $R3$ is selected.

tree of layer-1 multicast tree rooted at the source S and including all data-paths which have enough bandwidth to deliver the layer-2 data from the source. Fig. 6 shows the initial multicast tree for layer-2. Some of the data links and nodes of T_1 are not in the initial layer-2 tree because the

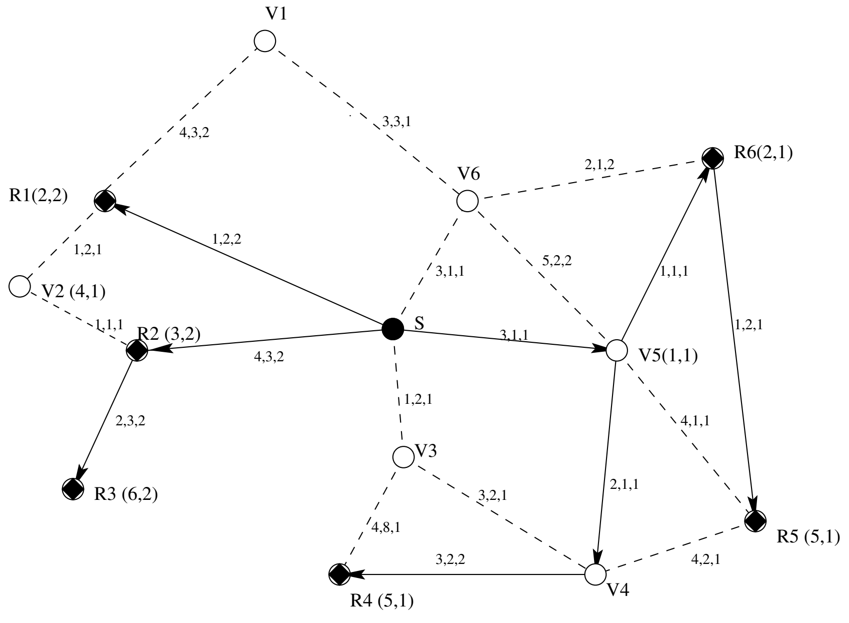

Fig. 5. Layer-1 multicast tree after pruning.

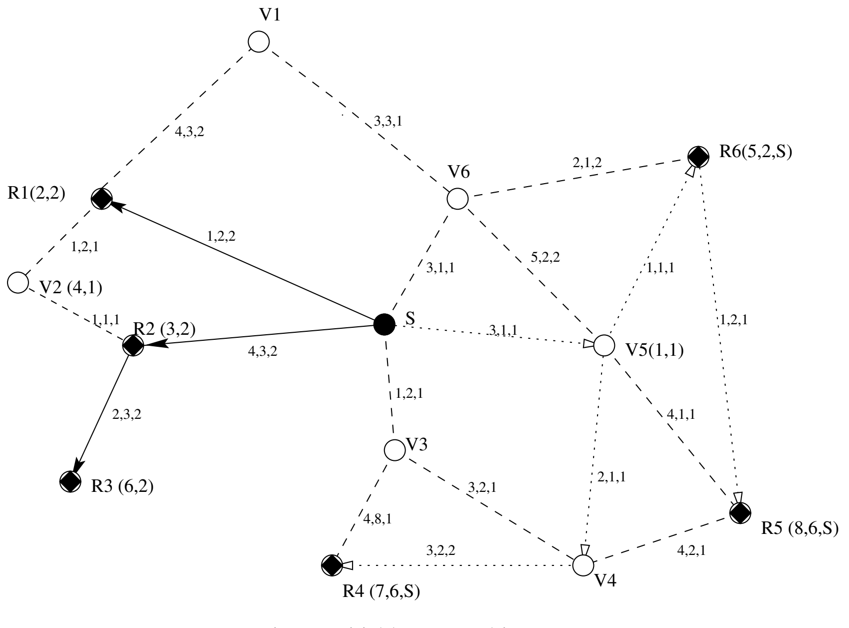

Fig. 6. Initial layer-2 multicast tree.

corresponding data-paths from the source do not have enough bandwidth to deliver layer-2 data. Also note that a leaf of the initial multicast tree is not necessarily a layer-2 receiver. It can be a pure router at this step. Fig. 6 shows that the receivers $R1, R2$, and $R3$ are now already in the tree, and we do not need to perform any calculation for them.

Using a similar procedure as for the construction of the layer-1 multicast tree, node *R*6 is found to have the minimum cost among all DCMC paths from the layer-2 multicast tree to layer-2 receivers not on the tree. Hence, this path is added to the multicast tree, as shown in Fig. 7.

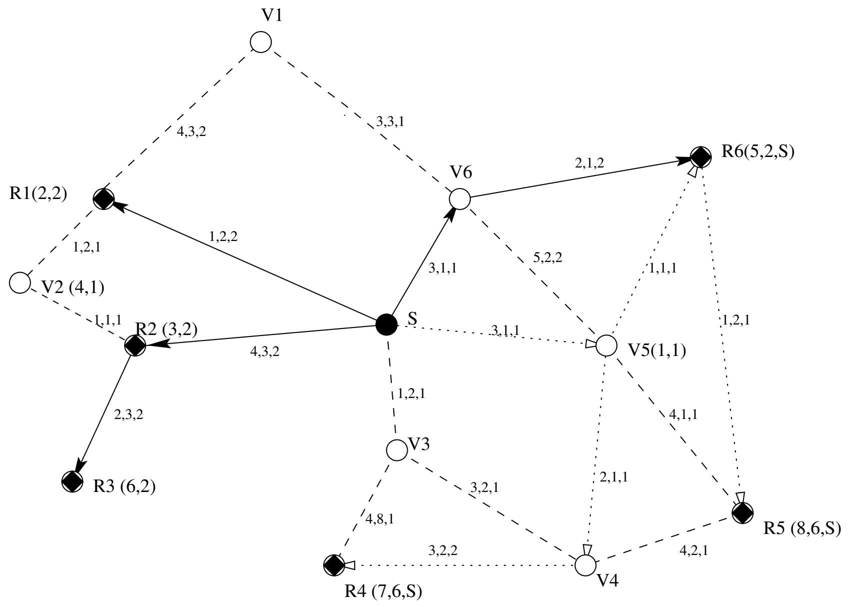

Fig. 7. Layer-2 multicast tree after *R*6 is added.

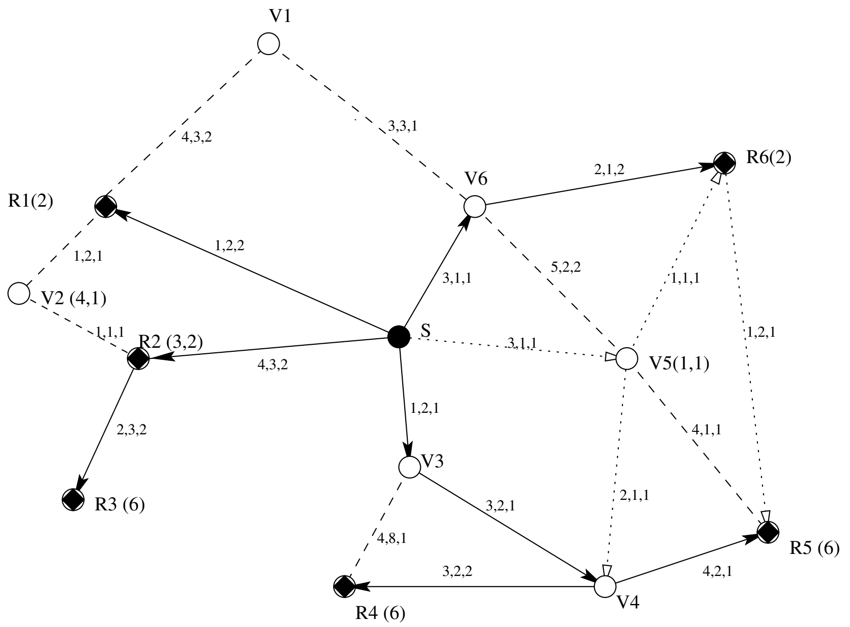

Fig. 8. Layer-2 multicast tree after *R*4 and *R*5 are added (Now, all receivers are on the tree).

Fig. 8 shows the layer-2 multicast tree after adding receivers *R*4 and *R*5 in sequence. At this time, all layer-2 receivers are on the multicast tree. The algorithm then prunes those nodes and data links that are not on any receivers' data-path. In this example, no such node or link exists, so the resulting tree after pruning is the same as in Fig. 8.

Fig. 9 shows the multicast super-tree for both layer 1 and layer 2. The thick lines are data links for both layer 1 and layer 2; the thin lines and dotted lines are data links for layer 1 and layer 2 only, respectively. If we closely examine the entire graph, we find that the data-paths to deliver both layers is not a tree, but it is a super-tree. We also note an important advantage of layered-video distribution by studying this example: the bandwidth of data-paths to receiver *R*5 is less than the total bandwidth necessary to distribute both layers at the same time. If the video data is encoded in a single stream, *R*5 cannot receive the video at all because no data-path of sufficient bandwidth is available. However, after using the layered-encoding scheme and our LVMSR algorithm, the data can now be distributed to receiver *R*5 successfully by using different paths to deliver different layers.

5. LVMSR properties

5.1. Dynamic member management in LVMSR

The LVMSR algorithm is well suited for dynamic member management. It allows easy addition or deletion of receivers at any time. To add a receiver, the new receiver requests the source to send the current multicast super-tree information. The receiver then calculates the DCMC paths for every layer from the super-tree to itself, starting from base layer. Then, the receiver sends the path information to the source. The source includes the newly selected data-paths and the receiver in the super-tree. Deleting a receiver from the multicast session can be done by excluding all paths which are used by the receiver exclusively.

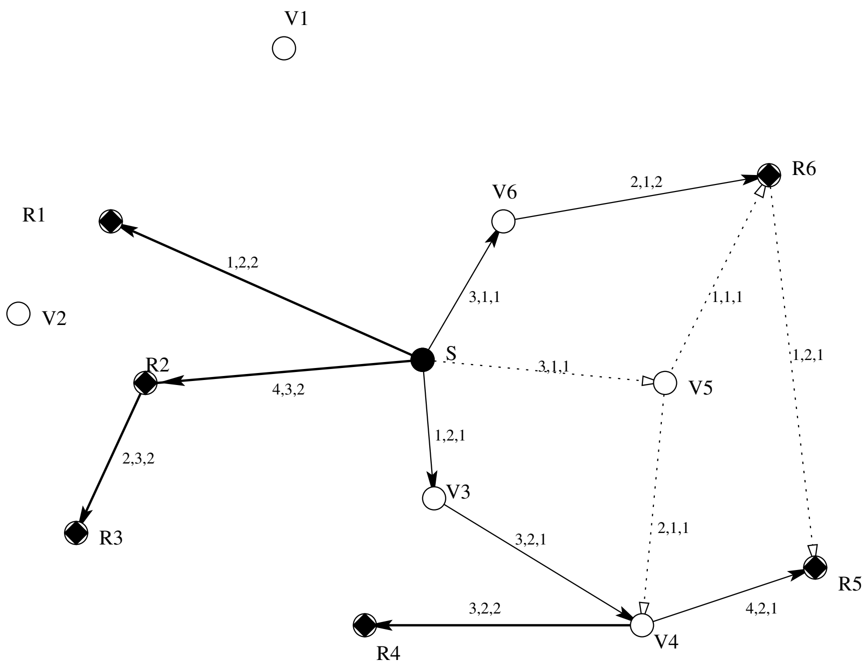

Fig. 9. Multicast super-tree for both layers.

5.2. Properties

The LVMSR algorithm has the following properties:

Property 1. A node i in a newly selected data-path is originally in the current-layer multicast tree if and only if the delay from the source to node i along the newly selected data-path is less than the delay along the old data-path; and using this node as a fork-node for the currently selected receiver violates the delay constraint.

Proof. Suppose the delay for the node in the newly selected data-path is greater than the previous one. Then, the algorithm will select this node as the fork-node for the selected receiver with smaller cost and lower delay. This contradicts the algorithm. Hence, it is impossible that the newly selected path could be selected. □

Property 2. The multicast tree for any layer is a tree. There is only one data path, if it exists, from the source to any receiver in the tree.

Proof. At each step, a receiver and its corresponding delay-constrained data-path are added to the current multicast tree. Any data link that originally connected to a node (except the fork-node) on the newly selected path as well as on the previous multicast tree is discarded. This guarantees that there are no multiple data-paths for any node in the tree. □

Property 3. The multicast super-tree, if it exists, can always satisfy the delay constraint.

Proof. When the algorithm calculates the DCMC path from the current tree to a receiver, it calculates both lowest-cost paths and minimum-delay paths for it. If the receiver is selected, then only the path that satisfies the delay constraint can be selected. [8] Therefore, the data-path for every receiver in every layer can satisfy the delay constraint, if it exists. This proves that the super-tree, if it exists, can always satisfy the delay constraint. □

[8] If the minimum-cost path is not chosen because it violates the delay constraint, the minimum-delay path is considered instead. If the delay along the minimum-delay path is greater than the delay constraint, then no path is available to satisfy the delay requirement.

Property 4. The multicast super-tree constructed by LVMSR is a weighted super-ditree.

Proof. Every data link is from a node near the source to a node near the receivers, and every data link is weighted by cost and delay. Therefore, the super-tree is a weighted super-ditree. □

5.3. Complexity of LVMSR

Our heuristic algorithm is a fully distributed algorithm. It is up to receivers to decide on the delay-constrained, lowest-cost path from a current multicast tree to itself. The source is only responsible for deciding which receiver and the corresponding delay-constrained, minimum-cost path to be included in the multicast tree, based on the information sent to it by the receivers. The source is also responsible for sending the updated multicast-tree information to those unselected receivers.

Let n be the total number of nodes in the network, and R be the number of receivers in the multicast group. For every layer's multicast-tree construction, the source retrieves the initial multicast-tree information and sends it to all receivers. This operation takes one message and $O(n)$ time, where n is the number of nodes. Every receiver v which is not on the multicast tree calculates its constrained lowest-cost path $P(T,v)$ from the current tree T to itself separately. If $P(T,v)$ is calculated using an efficient shortest-path algorithm, e.g., Dijkstra's algorithm, the time complexity of this step is $O(n^2)$. Then, every receiver sends the $P(T,v)$ information to the source so that the source can decide which receiver it will select in the next step. Suppose k receivers, $0 < k \leqslant N_l$ (N_l is the total number of receivers requesting layer-l data), are not in the multicast tree; then, k messages will be sent to the source in this step. After receiving the feedback messages, the source selects the receiver which has the minimum path cost $P_C(T,v)$ and then includes this receiver into the tree. The complexity of selection is $O(k)$. This procedure continues until all receivers are in the multicast tree. Then, the algorithm prunes the multicast tree, which takes $O(n)$ time, and the result is the current-layer multicast tree. Therefore, the total time to construct one layer of the multicast tree is

$$O(T) = O\left(n + N_l n^2 + \sum_{k=1}^{N_l} k\right) = O(N_l n^2). \tag{6}$$

The total number of messages needed to construct the layer-l multicast tree is $N_l(N_l - 1)/2$. So, the complexity for constructing the L layers of the video multicast super-tree is

$$O(T) = \sum_{l=1}^{L} O(N_l n^2). \tag{7}$$

The number of messages needed is $\sum_{l=1}^{L} (N_l \times (N_l - 1)/2)$. In an actual system, L is expected to be a small positive integer, so the time complexity of this algorithm is $O(Rn^2)$ and the message complexity is $O(R^2)$, where R is the number of receivers.

The above analysis does not take into consideration the propagation delay of signals on the data links. The total propagation time will be $O(R)$ RTTs, which is the maximum round-trip time from any receiver to the source. This is because there are R iterations to construct one layer of the multicast tree, and every iteration will take one RTT at most if the intermediate nodes' processing time is negligibly small relative to the RTT.

5.4. Scalability

The LVMSR scheme relies on the source to interact with the receivers to select the multicast data-paths. The interactions might cause some scalability concern. However, as stated before, the receivers in other autonomous systems rely on their border gateways to act as proxy sources. Therefore, the actual multicast super-tree calculation is limited to intra-domain routing. For inter-domain routing, the scheme must still rely on an inter-domain routing protocol, e.g., the Border-Gateway Protocol (BGP). Therefore, LVMSR's scalability is not expected to be a problem.

6. Illustrative numerical results

To illustrate some additional performance characteristics of LVMSR, we run the following simulation experiment. We generate a random network topology with 400 nodes and randomly select N ($N \leqslant 100$) receivers. The network generation is based on the idea of hierarchical nodes and random links [21]. A network is first divided into 10×10 grids, and 20% of the grids are randomly selected as high-density areas and the other 80% are selected as lower density areas. The network plan is divided into four equal-sized wide areas and a center router is placed in the middle of each area. The wide-area center routers are fully connected, and they represent the backbone network. For every dense area (one grid) or every four sparse grid (2×2), an area-center router is placed in its center. These are second-layer routers inside the network, acting as metro-area center routers. A data link is placed between any area center router and its wide-area center router. Also, any two area-center routers may be connected. A link between two area center routers is added by using the following probability density function p_l [5,21]:

$$p_l = \lambda \exp(-d(u,v)/\rho D), \tag{8}$$

where $d(u,v)$ is the distance between u and v; D is the maximum distance between any two nodes that may be connected by a link; λ is used to control the data link densities (a larger value of λ results in higher link densities); and ρ is used to control the density of short data links relative to long data links (a smaller value of ρ results in more short data links). In our simulation experiments, we set $\lambda = 10.0$, $\rho = 0.1$, and $D = 25\sqrt{2}$, which is the longest distance from a node to its wide-area center router.

After placing all layer-2 nodes and the corresponding links, layer-3 routers and links are generated. A layer-3 router is placed randomly in a grid according to the placement probability. When a router is placed into a grid, it randomly selects a position which is not occupied by any router inside the grid. After all nodes are placed in the plane, a data link is added between a layer-3 router and its area center router (layer-2 router). A link between layer-3 routers may also exist. The probability of a link existing between any two layer-3 nodes follows the same probability function p_l given in Eq. (8), but with parameters $\lambda = 0.7$, $\rho = 0.2$, and $D = 20$.

The network topology generated for our experiments using the above method has 400 routers and 1182 links. The average nodal degree in this network is about 3, which is a good approximation for networks in the real world. Fig. 10 shows a network topology that is generated by our simulation. In this network, no duplicate links exist between any two nodes.

The cost of a data link between any two routers is a random number $c(e)$, where $c(e) \in U(50, 70)$ for two layer-1 nodes (wide-area center node); $c(e) \in U(10, 30)$ for two layer-2 nodes (area center routers) or between a layer-1 node and a layer-2 node; and $c(e) \in U(1, 11)$ for any two layer-3 routers or a layer-3 and a layer-2 router. The delay of any data link is assumed to be the length of the data link. The average cost and delay for all data links in this network turns out to be 7.2453 and 7.5540, respectively. We assume that the total bandwidth of a data link is 100 Mbps between any two wide-area centers, and 10 Mbps for all other data links. At any time, a link's available bandwidth is a random number less than its total bandwidth. The available bandwidth for a data link in different directions can be different.

For illustration purposes, let us assume a layered video application with two layers. The bandwidth requirement for each layer is 1 Mbps. Assume also that the system has some mechanism to reserve resources when setting up data-distribution paths and the available bandwidth will not change while setting up the data-paths. The LVMSR algorithm successfully constructs a multicast super-tree. Fig. 11 shows the multicast tree for the first layer. There are 50 receivers in this tree. The delay constraint is $\Delta = 110$ ms in this experiment.

Fig. 12 shows the multicast tree for the second layer. Among the 50 receivers, half of them are randomly selected as receivers requesting data for both layers. We note that most of the nodes and links in Fig. 12 are the nodes and links in Fig. 11

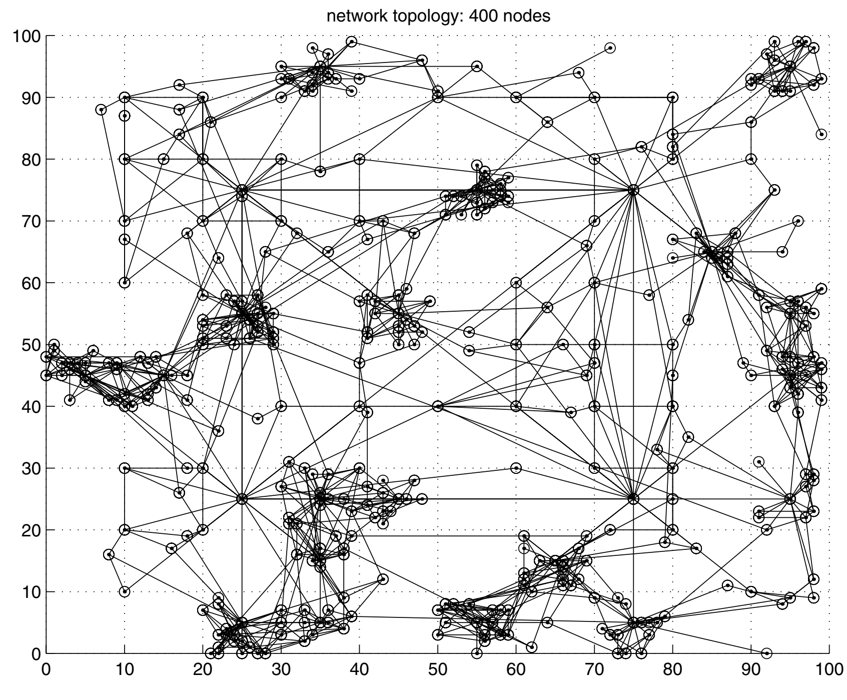

Fig. 10. Network topology in our simulation experiment.

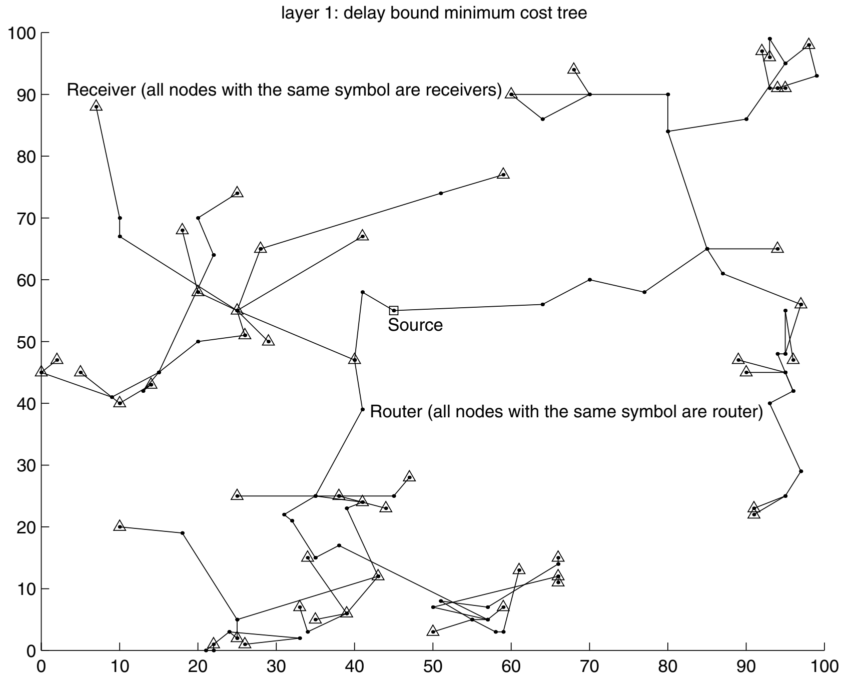

Fig. 11. Multicast tree for the first layer.

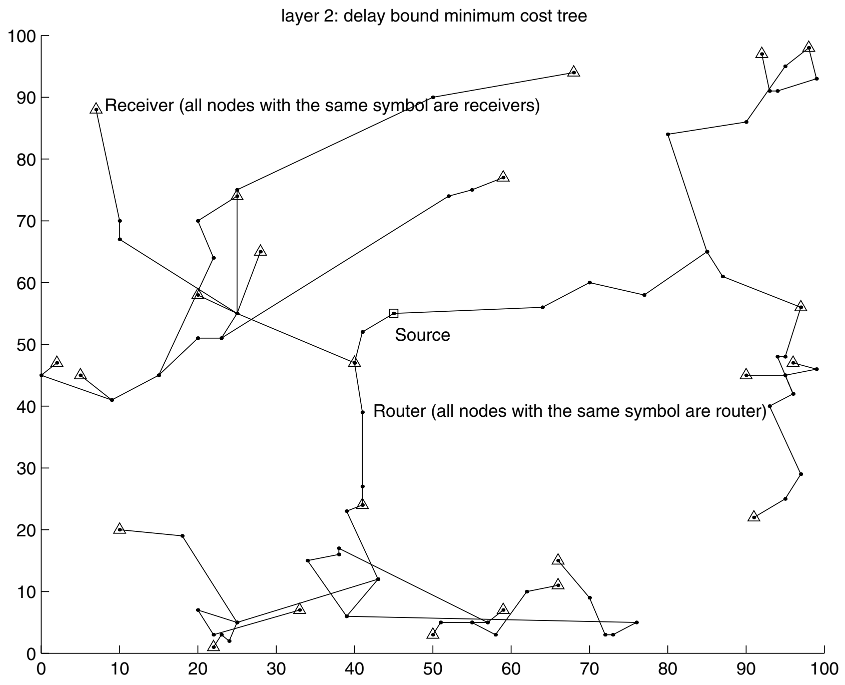

Fig. 12. Multicast tree for the second layer.

also. The multicast trees have some long links because these links have smaller cost than the unselected ones. We also wish to point out that some links overlap because more than two nodes may be in a line in a graph. Thus, some nodes seem to be connected together even though they are not. It was already proved in Section 5 that the multicast path for every layer is a tree. All leaves in a layer's tree are receivers; however, a receiver is not necessarily a leaf in the multicast tree. It could be a routing node and a receiver at the same time.

The multicast paths to deliver all of the data form a multicast super-tree, as shown in Fig. 13, which clearly shows that it is no longer a tree structure. Some receivers may have different data-paths for different layers of data.

Using the above network model, we studied the relationship between the total cost and the delay bound Δ. In this study, we randomly select 50 nodes with half of them requesting both layers. The bandwidth requirements of each layer is 1 Mbps. Fig. 14, which plots the cost vs. the delay bound Δ shows that, when the delay constraint is tight, it costs more to deliver the data. The total cost reduces fast and then becomes relatively flat when the delay bound increases. The total cost changes little when the delay bound is large, because it is no longer an actual constraining factor in constructing the multicasting tree, i.e., the resulting multicast tree for every layer is equivalent to the minimum-cost multicast tree.

We have also studied the average cost per layer per receiver for our algorithm. In this experiment, about half of the receivers in every test requests both layers. The group size is changed from 10 to 100. Fig. 15 plots the average cost rate per layer per receiver vs. group size. The three lines correspond to average cost ($/Mb) per receiver for layer 1, layer 2, and both layers, respectively. Fig. 15 shows that, when the group size is small, the average cost is high. This is because every receiver has a longer data-path from the source as the number of links per receivers is large. When the group size increases, the average cost initially decreases very fast and then it is relatively flat. The average cost rate per receiver is the smallest for

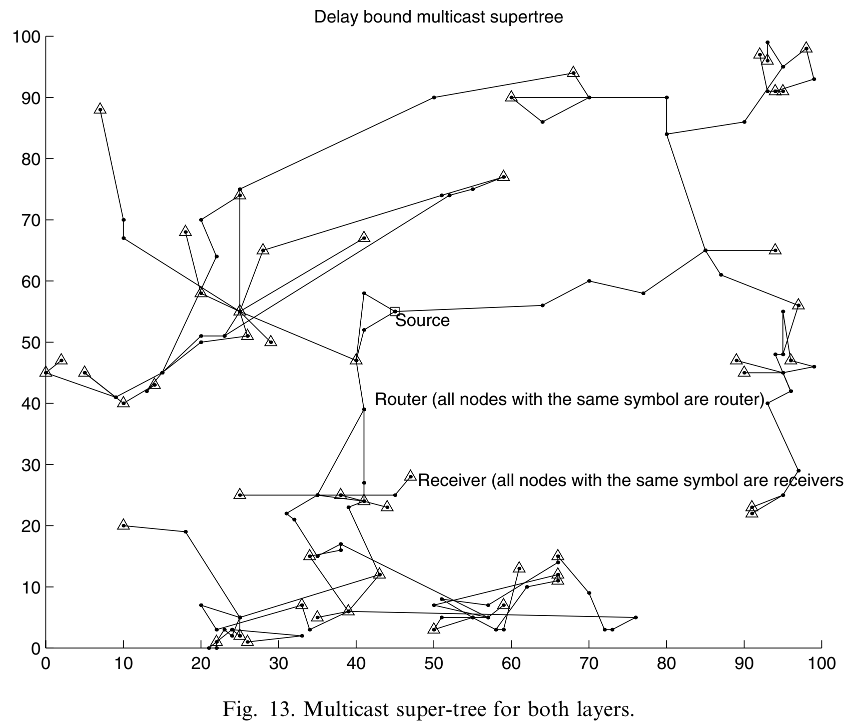

Fig. 13. Multicast super-tree for both layers.

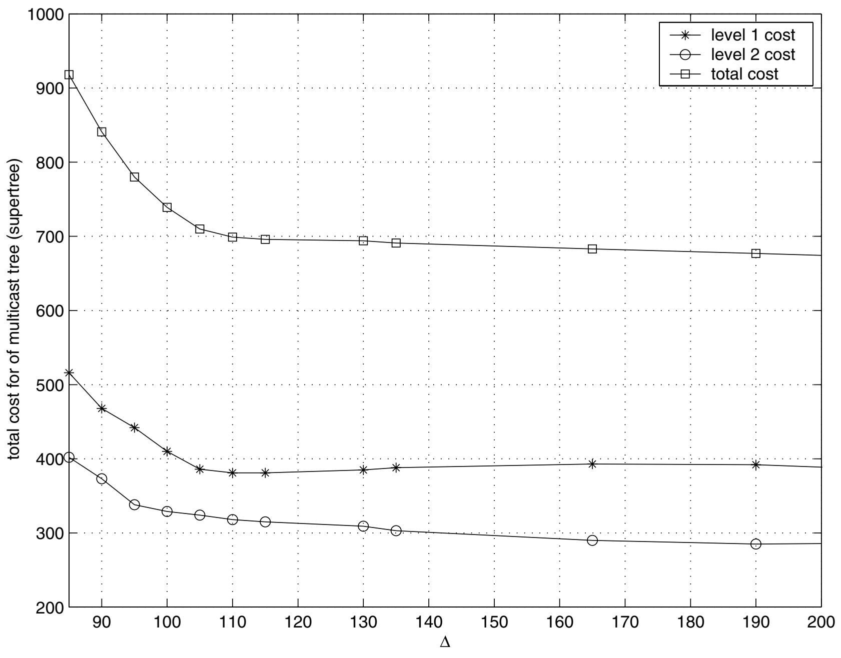

Fig. 14. Total cost vs. delay bound Δ.

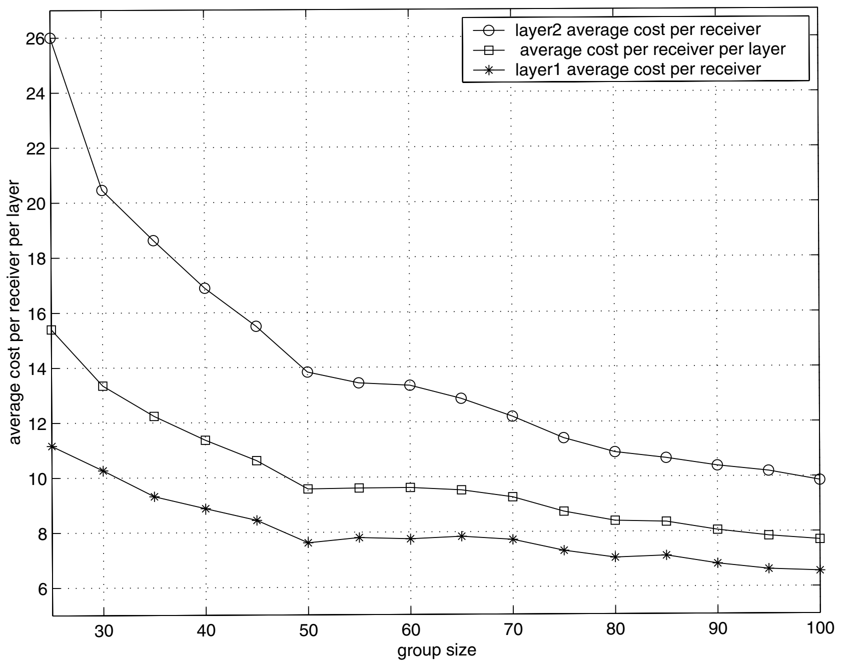

Fig. 15. Average cost per layer per receiver vs. group size.

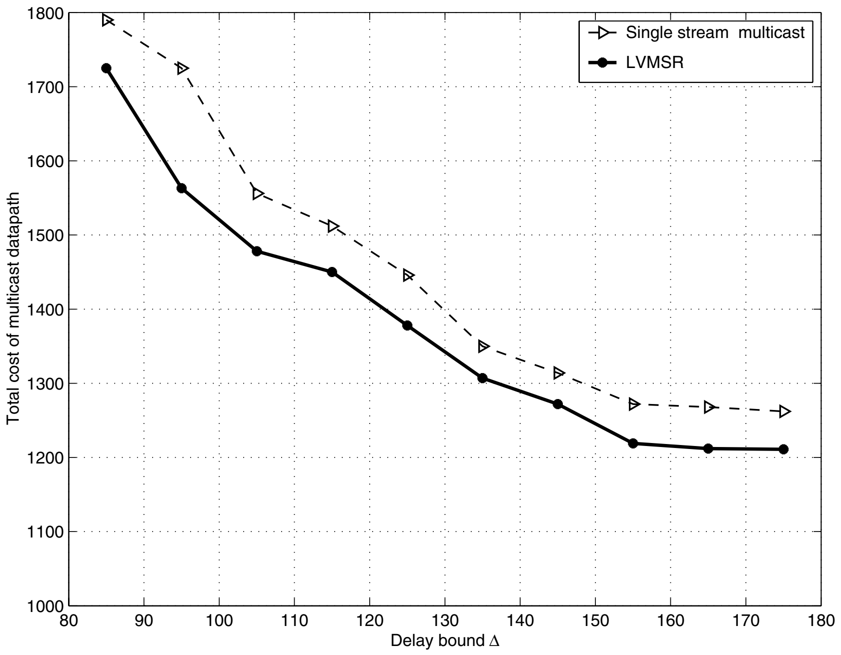

Fig. 16. LVMSR and single-stream multicast comparison.

layer-1 data and the largest for layer-2 data because our algorithm gives higher priority for the layer-1 multicast-tree construction. We also note that, even though the average cost rate per link is \$7.25/Mb in our network topology and the average number of links per receiver is much larger than one because there are some routing nodes in a multicast super-tree, the average cost rate per receiver for a large group size is only about 6.5, 7.7, 10.0 (\$/Mb per receiver) for layer 1, layer 2, and both layers, respectively. The average cost rate per receiver for layer 1 is less than the average of a link's cost rate (7.25), and the average cost rate per receiver per layer is only a little larger than the average cost rate per link. This confirms that LVMSR is an effective algorithm in finding the minimum-cost, delay-constrained multicast super-tree.

We also compare the performance of layered-video muticast by using LVMSR vs. using a single-stream multicast. Fig. 16 compares the total cost of multicast data-path vs. delay bound for our LVMSR scheme as well as the single-stream multicast scheme for the same network as earlier (Fig. 10). For this study, we randomly select 100 nodes as receivers. The results indicate that the overall trends for the single-stream multicast and LVMSR are similar, with the total cost decrease being gradual in the beginning, and then becoming flat when the delay bound is larger, which implies that the delay bound is no longer an actual constraint for selecting the multicast tree. Overall, our LVMSR scheme's total cost is found to be about 5–10% lower than that of the single-stream multicast scheme.

7. Conclusion

We presented a new distributed algorithm, called LVMSR, with $O(Rn^2)$ time complexity and $O(R^2)$ message complexity to construct layered-video multicast data-paths. The multicast data-paths form a multicast super-tree. The super-tree constructed by our algorithm can always satisfy the delay constraint while providing small delay jitter and low cost. By using our LVMSR algorithm, data from different layers may be routed via different paths in case of insufficient bandwidth, which can improve the QoS of the received video. Results from our simulation experiments show that the algorithm performs well in wide-area mesh networks.

References

[1] D. Eppstein, Finding the *k* shortest paths, SIAM Journal on Computing 28 (2) (1998) 652–673.

[2] A.D. Gelman, H. Kobrinski, L.S. Smoot, S.B. Weinstein, M. Fortier, D. Lemay, A store-and-forward architecture for video-on-demand service, Canadian Journal of Electrical and Computer Engineering 18 (1993) 37–40.

[3] S.L. Hakimi, Steiner's problem in graphs and its implications, Networks 1 (2) (1971) 113–133.

[4] Y. Ishibashi, Y. Tachibana, S. Tasaka, Responsiveness of layered multicast and feedback control for video traffic in the Internet, Proceedings of 2000 IEEE International Conference on Communications, vol. 2, 2000, pp. 846–852.

[5] X. Jia, A distributed algorithm of delay-bounded multicast routing for multimedia applications in wide area networks, IEEE/ACM Transactions on Networking 6 (6) (1998) 828–837.

[6] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller, J.W. Thatcher (Eds.), Complexity of Computer Computations, Plenum, New York, 1972, pp. 85–103.

[7] V.P. Kompella, J.C. Pasquale, G.C. Polyzos, Multicast routing for multimedia applications, IEEE/ACM Transactions on Networking 1 (1993) 286–292.

[8] V.O.K. Li, W. Liao, Distributed multimedia systems, Proceedings of the IEEE 85 (1997) 1063–1108.

[9] X. Li, S. Paul, M. Ammar, Layered video multicast with retransmissions (LVMR): evaluation of hierarchical rate control, Proceedings of IEEE INFOCOM 1998, vol. 3, 1998, pp. 1062–1072.

[10] X. Li, M.H. Ammar, Video multicast over the Internet, IEEE Network 13 (2) (1999) 46–60.

[11] T. Little, D. Venkatesh, Prospects for interactive video-on-demand, IEEE Multimedia Magazine (Fall 1994) 14–24.

[12] N.F. Maxemchuk, Video distribution on multicast networks, IEEE Journal on Selected Areas in Communications 15 (3) (1997) 357–372.

[13] S. McCanne, M. Vetterli, V. Jacobson, Low-complexity video coding for receiver-driven layered multicast, IEEE Journal on Selected Areas in Communications 15 (6) (1997) 983–1001.

[14] M. Parsa, Q. Zhu, J.J. Garcia-Luna-Aceves, An iterative algorithm for delay-constrained minimum-cost multicasting, IEEE/ACM Transactions on Networking 6 (4) (1998) 461–474.

[15] N. Shacham, Multipoint communication by hierarchically encoded data, Proceedings of IEEE INFOCOM92, vol. 3, 1992, pp. 2107–2114.

[16] N. Shacham, J.S. Meditch, An algorithm for optimal multicast of multimedia streams, Proceedings of IEEE INFOCOM 1994, vol. 2, 1994, pp. 856–864.

[17] R. Sriram, G. Varghese, G. Chandranmenon, A rearrangeable algorithm for the construction of delay-constrained dynamic multicast trees, Proceedings of IEEE INFOCOM 1999, vol. 3, 1999, pp. 1073–1080.

[18] D. Taubman, A. Zakhor, Multirate 3-D subband coding of video, IEEE Transactions on Image Processing 3 (5) (1994) 572–588.

[19] F.A. Tobagi, Distance learning with digital video, IEEE Multimedia Magazine (1995) 90–94.

[20] B. Wang, J.C. Hou, QoS-based multicast routing for distributing layered video to heterogeneous receivers in rate-based network, Proceedings of IEEE INFOCOM 2000, vol. 2, 2000, pp. 480–489.

[21] B.M. Waxman, Routing of multipoint connections, IEEE Journal on Selected Areas in Communications 6 (9) (1988) 1617–1622.

Wushao Wen received the B.S. degree from the University of Science and Technology of China in 1993, M.S. and Ph.D. degree from the University of California, Davis in 1999 and 2001 respectively, all in Electrical and Computer Engineering. He is currently a Senior Engineer in Ciena Corporation, working on optical routing and signaling system. In 2001, he worked in Summit Networks as a Member of Technical Staff. From 2000 to 2001, he was with Cisco Systems Inc. working on network management. From 1993 to 1997, he was with China Telecommunication Inc. as a senior engineer and project manager. He is the co-author of the textbook "Related Database Management System Design and Programming" published by China People's Post and Telecommunication Publishing, Inc. His research interest is on lightwave networks, network quality of service, and multimedia networks.

Biswanath Mukherjee (S'82–M'87) received the B.Tech. (Hons) degree from Indian Institute of Technology, Kharagpur (India) in 1980 and the Ph.D. degree from University of Washington, Seattle, in June 1987. At Washington, he held a GTE Teaching Fellowship and a General Electric Foundation Fellowship. In July 1987, he joined the University of California, Davis, where he has been Professor of Computer Science since July 1995, and served as Chairman of Computer Science during September 1997–June 2000. He is co-winner of paper awards presented at the 1991 and the 1994 National Computer Security Conferences. He serves or has served on the editorial boards of the IEEE/ACM Transactions on Networking, IEEE Network, ACM/Baltzer Wireless Information Networks (WINET), Journal of High-Speed Networks, Photonic Network Communications, and Optical Network Magazine. He also served as Editor-at-Large for optical networking and communications for the IEEE Communications Society. He served as the Technical Program

Chair of the IEEE INFOCOM'96 conference. He is author of the textbook "Optical Communication Networks" published by McGraw-Hill in 1997, a book which received the Association of American Publishers, Inc.'s 1997 Honorable Mention in Computer Science. He is a Member of the Board of Directors of IPLocks, Inc., a Silicon Valley startup company. He has consulted for and served on the Technical Advisory Board of a number of startup companies in optical networking. His research interests include lightwave networks, network security, and wireless networks. His e-mail address is: mukherje@cs.ucdavis.edu.

S.-H. Gary Chan (M'98) received the Ph.D. degree in Electrical Engineering with a minor in Business Administration from Stanford University, Stanford, CA, in 1999, and the B.S.E. degree (highest honor) in Electrical Engineering from Princeton University, Princeton, NJ, in 1993.

He is currently an Assistant Professor with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, and an Adjunct Researcher with the Microsoft Research Asia in Beijing. He was a Visiting Assistant Professor in networking at the Department of Computer Science, University of California, Davis, from September 1998–June 1999. During 1992–93, he was a Research Intern at the NEC Research Institute, Princeton, NJ. His research interest includes multimedia networking, peer-to-peer networks, high-speed and wireless communications networks, and Internet technologies and protocols. Dr. Chan was a William and Leila Fellow at Stanford University during 1993–94. At Princeton, he was the recipient of the Charles Ira Young Memorial Tablet and Medal, and the POEM Newport Award of Excellence in 1993. He is a member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa.

Dipak Ghosal received his B.Tech. degree in Electrical Engineering from Indian Institute of Technology, Kanpur, in 1983, his M.S. degree in computer science from Indian Institute of Science, Bangalore, in 1985, and his Ph.D. degree in Computer Science from the University of Louisiana, Lafayette, in 1988. From 1988 to 1990 he was a research associate at the Institute for Advanced Computer Studies at the University of Maryland (UMIACS) at College Park. From 1990 to 1996 he was a member of technical staff at Bellcore, Red Bank, New Jersey. Currently, he is with the faculty of the Computer Science Department at the University of California, Davis. His research interests are in the areas of IP telephony, peer-to-peer networks, mobile and ad hoc networks, and performance evaluation of computer and communication systems.