A Scalable Overlay Multicast Architecture for Large-Scale Applications

Li Lao, *Member*, *IEEE*, Jun-Hong Cui, *Member*, *IEEE*,
Mario Gerla, *Fellow*, *IEEE*, and Shigang Chen, *Member*, *IEEE*

Abstract—In this paper, we propose a Two-tier Overlay Multicast Architecture (TOMA) to provide scalable and efficient multicast support for various group communication applications. In TOMA, Multicast Service Overlay Network (MSON) is advocated as the backbone service domain, while end users in access domains form a number of small clusters, in which an application-layer multicast protocol is used for the communication between the clustered end users. TOMA is able to provide efficient resource utilization with less control overhead, especially for large-scale applications. It also alleviates the state scalability problem and simplifies multicast tree construction and maintenance when there are large numbers of groups in the network. To help MSON providers efficiently plan backbone service overlay, we suggest several provisioning algorithms to locate proxies, select overlay links, and allocate link bandwidth. Extensive simulation studies demonstrate the promising performance of TOMA.

Index Terms—Network architecture and design, multicast, network management.

1 Introduction

The Internet has overseen more and more emerging group communication applications, such as video conferencing, video on-demand, network games, and distributed interactive simulation (DIS). Over the years, tremendous efforts have been made to provide multicast support, ranging from IP multicast to recently proposed application-layer multicast. IP multicast utilizes a tree delivery structure which makes it fast, resource-efficient, and scalable for very large groups. However, IP multicast is still far from being widely deployed in the Internet due to various technical and marketing reasons [2], [14]. The most critical ones include the lack of a scalable interdomain routing protocol, the state scalability issue when there are a large number of groups, and the requirement of global deployment of multicast-capable IP routers. These issues make Internet Service Providers (ISPs) reluctant to deploy and provide multicast service.

Recently, researchers resort to the application-layer multicast approach, which implements multicast-related features at end hosts [5], [8], [12], [18], [21], [24], [25], [28], [30]. Data packets are replicated and transmitted between end hosts via unicast. These systems do not require infrastructure support and, therefore, can be easily deployed. However, application-layer multicast is generally not scalable for very large multicast groups due to its low bandwidth efficiency and heavy control overhead caused by tree maintenance at end hosts. In addition, because multicast groups are solely managed at end hosts, it is difficult for an ISP to have efficient member access control and to obtain group bandwidth usage, which makes a good pricing model impractical, if not impossible.

This paper studies the problem of providing practical solutions for large-scale multicast applications. A multicast service model involves multiple parties, such as network service providers (i.e., higher-tier ISPs), Internet Service Providers (i.e., lower-tier ISPs, or ISPs for short), and end users. Their relationship is loosely analogic to that among manufacturers, dealers, and consumers, with the raw product of bandwidth being sold to end users by ISPs through the means of multicast applications. Now, which party cares most about using multicast? End users do not as long as they get the required functionalities at a reasonable price. Neither do network service providers, as far as they can sell their connectivity/bandwidth service. Obviously, ISPs in the middle are the ones who care the most—their goal is to use limited bandwidth purchased from network service providers to support as many users as possible. Therefore, in order to stimulate the wide deployment of multicast, it is critical to develop a practical, comprehensive, and profitable multicast service model for these ISPs.

To address the above challenge, we propose a **T**wo-tier **O**verlay **M**ulticast **A**rchitecture (called **TOMA**) that provides scalable, efficient, and practical multicast support for various group communication applications. In this architecture, we advocate the notion of **M**ulticast **S**ervice **O**verlay **N**etwork (**MSON**) as the backbone service domain. An MSON consists of service nodes or proxies strategically deployed by an MSON provider (ISP). The MSON provider provisions its overlay network according to user traffic

- *L. Lao is with Google Inc., 604 Arizona Ave., Santa Monica, CA 90401. E-mail: llao@google.com.*
- *J.-H. Cui is with the Computer Science and Engineering Department, University of Connecticut, 371 Fairfield Rd., Unit 2155, Storrs, CT 06269-2155. E-mail: jcui@engr.uconn.edu.*
- *M. Gerla is with the Computer Science Department, University of California Los Angeles, 3732F Boelter Hall, Los Angeles, CA 90095-1596. E-mail: gerla@cs.ucla.edu.*
- *S. Chen is with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611. E-mail: sgchen@cise.ufl.edu.*

Manuscript received 14 Dec. 2005; revised 2 May 2006; accepted 9 May 2006; published online 9 Jan. 2007.
Recommended for acceptance by M. Ould-Khaoua.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0504-1205.
Digital Object Identifier no. 10.1109/TPDS.2007.1008.

characteristics (based on long-term measurement), purchases bandwidth from network service providers based on service level agreements (SLAs), and sells multicast services to group coordinators via service contracts. Outside MSON, end hosts subscribe to the MSON by connecting to proxies advertised by the MSON provider and form clusters around these proxies. In each cluster, application-layer multicast (instead of unicast) is used for efficient data delivery. The end users only need to pay for their regular network connection service outside MSON.

The proposed TOMA architecture not only provides scalable and efficient multicast support, it also brings many other advantages. First, unlike some other existing multicast overlays (such as [10], [12], [18], [19]) where each overlay only supports one group, an MSON provider can support a variety of group communication applications simultaneously. Second, since MSON is based on well-defined business relationships with network service providers and group coordinators, overlay service providers can put major efforts on planning and managing their overlay networks. Third, the notion of MSON significantly simplifies the management of underlying networks. Network service providers only need to provide services to limited numbers of MSON providers instead of millions or billions of individual end users. This level of traffic aggregation, in the long run, will make IntServ practical.[1]

To make TOMA a reality, we face many challenges. In particular, the efficient **MSON management** and **MSON provisioning** are the most critical issues. As an MSON is expected to accommodate a large number of multicast groups, it is crucial for an MSON provider to efficiently establish and manage numerous multicast trees. Moreover, given user traffic characteristics, the MSON provider should carefully provision the overlay network in order to reduce operation cost and improve service quality.

In this paper, we target the above issues. To address the efficient management problem, we propose a lightweight, scalable protocol called OLAMP (OverLay Aggregated Multicast Protocol). In this protocol, we adopt the *aggregated multicast* approach [17], with multiple groups sharing one delivery tree. Outside MSON, we develop efficient proxy selection mechanisms, and choose a core-based application-layer multicast routing approach for data transmission inside clusters. In addition, we suggest several effective algorithms for locating overlay proxies, identifying overlay links, and provisioning bandwidth. Furthermore, we conduct extensive simulation studies and show the promising performance of TOMA as well as the effectiveness of our provisioning algorithms.

The rest of this paper is organized as follows: In Section 2, we review background and related work. In Section 3, we present an overview of the TOMA architecture and address the critical issues of MSON management and cluster formation outside MSON. In Section 4, we describe several algorithms for overlay network provisioning. In Section 5, we evaluate the performance of TOMA and the overlay provisioning algorithms by simulations. Finally, we summarize our contribution in Section 6.

1. From this aspect, we share a similar vision with SON [16], a service overlay proposed to provide scalable end-to-end QoS support.

2 Background and Related Work

2.1 Aggregated Multicast

It is known that IP multicast is plagued from the state scalability problem, which refers to the explosion of multicast state (i.e., memory to store multicast state in the routers) and control overhead (i.e., multicast tree setup and maintenance overhead when a "soft-state" approach is employed) in the presence of a large number of coexisting multicast groups. Aggregated multicast has been proposed to improve multicast state scalability in transit (especially backbone) domains [17]. Observing that many multicast trees within a single domain are likely to have significant overlapping when there are numerous multicast groups, aggregated multicast allows multiple groups with similar members to share a single delivery tree and thus reduces the number of multicast trees in the domain. In this way, tree management overhead is significantly reduced, and multicast state information stored in the routers is dramatically decreased. In this paper, we design a protocol called OLAMP within the MSON based on aggregated multicast.

2.2 Related Work

These is a large body of work on application layer multicast [5], [8], [12], [18], [21], [24], [25], [28], [30]. In Yoid [18], each member selects its own parent to construct a multicast tree. In End System Multicast (ESM) [12], end hosts cooperatively build a mesh and establish a multicast delivery tree on top of this mesh. ALMI [24] uses a centralized entity to collect membership information and periodically calculate a minimum spanning tree. NICE [5] recursively arranges group members into a hierarchical overlay topology, which implicitly defines a source-specific delivery tree.

Recently, the notion of infrastructure-supported overlays has received increasing attention. Example seminal work includes Overcast [19] and RMX [11]. Both of them use overlay nodes to support multicast routing, and their main focus is on building reliable multicast trees. Several two-tier architectures have also been proposed for scalable multicast support (such as OMNI [6], MSN [26], and Scattercacst [10]). Moreover, in [6], [26], [27], the authors focus on single multicast tree optimization and endeavor to optimize end-to-end delay and access bandwidth usage at service nodes.

2.3 Our Contribution

Our work is different from the above work in the following aspects. The previous overlay architectures are based on different service models from ours. Most of them focus on improving multicast QoS routing performance, and they assume that the proxies are predeployed, overlay links between any two proxies are possible (i.e., a full mesh among proxies is used), and link capacities are simplified as proxy out-degree bounds. In contrast, the foundation of TOMA is well-defined business relationships between MSON providers, network service providers, and group coordinators. Accordingly, we focus on a number of new challenges faced by MSON providers, such as scalable MSON management, efficient cluster formation, and MSON provisioning. To the best of our knowledge, this paper is the first work to address the multicast state scalability issue in overlay networks. In other words, our major contribution is

a comprehensive multicast service overlay architecture, which includes a practical and profitable service model, an efficient and scalable overlay management protocol and effective overlay provisioning algorithms.

3 TOMA: A Two-Tier Overlay Multicast Architecture

3.1 TOMA Overview

In the TOMA architecture, MSON is advocated as the service backbone domain. It is an overlay network formed among a set of overlay proxies, on top of which multicast distribution trees are built for data delivery based on multicast routing protocols. Since an MSON provider always aims to have a bigger customer base and maximize its profit, the MSON management scheme should be scalable to the group size as well as the number of groups. We propose a protocol called OLAMP (Overlay Aggregated Multicast Protocol) for efficient and scalable MSON management. In OLAMP, we adopt aggregated multicast [17], which allows multiple groups to share one tree. Data packets are encapsulated when entering MSON, transmitted on aggregated trees, and decapsulated when exiting MSON. Outside MSON, end users subscribe to MSON by connecting to some proxies advertised by the MSON provider. Each proxy organizes some users into a "cluster," where an application-layer multicast tree (also denoted as peer-to-peer or P2P multicast tree) is formed for data delivery among the cluster members.

According to the functionalities of proxies, we define three kinds of proxies. The proxies that users connect to are **member proxies**. **Host proxies** are designated for managing multicast tree aggregation. The remaining **forwarding proxies** are responsible for forwarding multicast data packets.

EACH TOMA group is identified by a URL-like unique name in the form of TOMA://groupname.xxxmson.com/.[2] End users (sources and receivers) explicitly subscribe to a group by issuing a join request containing the URL-like group name. Through DNS, this request will reach the DNS server and a **group registry server** of the MSON. The group registry server enforces member access control policy and maintains group membership information. The DNS server will send back to the subscriber a list of IP addresses of the advertised member proxies, from which a member proxy will be selected.

After finding a member proxy, the end user sends its join request to the member proxy, which will subscribe to the multicast group inside MSON on behalf of this member. The member proxy will set up a P2P multicast tree in the local cluster and relay the join request to a host proxy to establish an overlay multicast tree in MSON using OLAMP.

In the backbone domain, each group is managed by a host proxy (which is similar to a core in CBT or an RP in PIM-SM). After receiving a join request for group g, this host proxy conducts multicast routing and group-tree matching to map group g to an aggregated tree. The host proxy for group g can be randomly selected by hashing the group identifier g to the host-proxy identifier space.

2. The URL-like naming approach has been adopted by many systems, such as CDN (content distribution networks), Yoid [18], Scattercast [10], and Overcast [19].

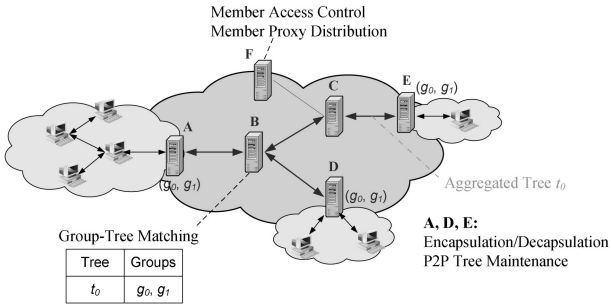


Fig. 1. A big picture of TOMA, where F is the group registry server/DNS server, B is the host proxy for groups g_0 and g_1, A, D, and E are member proxies, and groups g_0 and g_1 share the aggregated tree t_0.

In a nutshell, member proxies in TOMA manage P2P multicast trees in its cluster, host proxies conduct group-tree matching, and OLAMP connects member and host proxies, efficiently managing aggregated multicast trees in the MSON. A big picture of TOMA is illustrated in Fig. 1. In the following, we address major design issues of TOMA in detail.

3.2 OLAMP for Efficient MSON Management

If a proxy manages a large number of multicast trees, it has to maintain large forwarding tables and, thus, causes packet lookup speed to be slowed down. Furthermore, if a soft state approach is used, heavy tree management overhead due to multicast refreshing messages will be incurred. Therefore, we design OLAMP to address these issues.

OLAMP is used among proxies. It tries to use existing multicast trees to disseminate data for a new group. If there are no appropriate trees, a tree is established for this group based on multicast routing algorithms. Each member proxy maintains a simple group-tree mapping table in order to identify which trees are used for which groups. A host proxy needs to keep group and tree information (e.g., member-proxy lists of the assigned groups and tree structures) and conduct a group-tree matching algorithm. Note that the dynamic join or leave of individual end users generally do not affect the tree aggregation: Only when a cluster joins a group for the first time or completely leaves a group, the member proxy needs to join or leave the group. To facilitate our description, we denote the control messages in OLAMP as O-type messages, which include *O-JOIN*, *O-JOIN-ACK*, *O-LEAVE*, *O-LEAVE-ACK*, *O-SWITCH*, *O-GRAFT*, and *O-PRUNE*. Essentially, every proxy in MSON is capable of handling O-type messages and maintains a multicast routing table.

3.2.1 Member (Proxy) Join and Leave

When a member proxy mp decides to relay a join request for group g, it sends *O-JOIN*(g) to the host proxy hp of g. After conducting the group-to-tree matching, hp finds or computes an aggregated tree, say, t. It will send back an *O-JOIN-ACK*(g, t) message to mp. If mp has not joined the delivery tree t, it will graft to the tree by sending an *O-GRAFT*(t) message toward hp, and the proxies along this propagation path will update their routing tables accordingly.

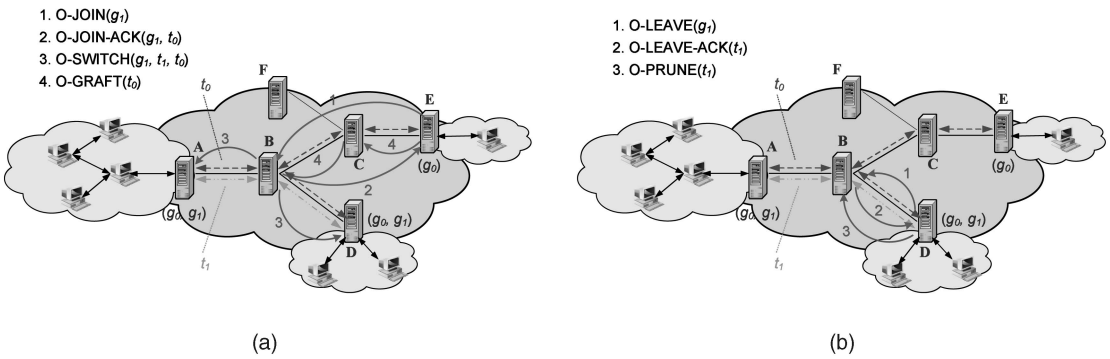


Fig. 2. OLAMP examples. (a) Member proxy E joins group g_1. (b) Member proxy D leaves roup g_1.

Similarly, when a member proxy mp discovers that no end users are connected, mp sends an *O-LEAVE*(g) message to the host proxy hp, which may trigger a group-tree matching process. If no other member proxies belong to group g, hp will remove the group-tree mapping between g and t, which may trigger removal of the tree t when no other groups are mapped onto t. In this case, hp sends an *O-LEAVE-ACK*(t) messages to the leaves of tree t, which will in turn prune from t by sending *O-PRUNE*(t) toward hp.

3.2.2 A Dynamic Group-Tree Matching Algorithm

In aggregated multicast, when an aggregated tree is bigger than a group, data will be delivered to nonmember nodes, leading to bandwidth waste. Obviously, there is a trade-off between bandwidth waste and aggregation: the more bandwidth we are willing to sacrifice, the more groups can share one tree and, thus, the better aggregation we can achieve. Hence, it is necessary to control the amount of bandwidth waste in group-tree matching. Assume that an aggregated tree t is shared by groups $g_i, 1 \le i \le n$, each of which has a native tree $t_0(g_i)$ (a native tree of a group is a "perfect" match for that group without wasting bandwidth, and it can be computed using multicast routing algorithms). Then, the **average percentage bandwidth overhead** of t can be defined as

$$\delta(t) = \frac{\sum_{i=1}^{n} B(g_i) \times (C(t) - C(t_0(g_i)))}{\sum_{i=1}^{n} B(g_i) \times C(t_0(g_i))} = \frac{C(t) \times \sum_{i=1}^{n} B(g_i)}{\sum_{i=1}^{n} B(g_i) \times C(t_0(g_i))} - 1, \quad (1)$$

where $C(t)$ is the cost of tree t (i.e., the total cost of the links on tree t), and $B(g)$ is the bandwidth requirement of group g. The cost metric can simply be the number of links or a function of bandwidth usage, congestion level, delay, or other factors.[3]

When a host proxy hp receives an *O-JOIN*(g) message from mp, it updates the corresponding entries of the multicast group table and executes the group-tree matching algorithm (Algorithm 1). The algorithm tries to minimize the number of aggregated trees without violating a given bandwidth overhead threshold b_{th}. It works as follows: If g is not new and the current tree t for group g is still appropriate (i.e., t can cover all the members of g with enough bandwidth and the bandwidth overhead $\delta(t) \le b\text{th}$), t is used for g. In all other cases, search existing trees T_e. If any existing tree is appropriate for g, it is considered as a candidate to cover g. Among all candidates, the tree with the minimum cost is selected. If no candidate is found, the native tree t_0 is used to cover g (if t_0 does not exist due to bandwidth constraint, the join request of mp has to be rejected). After mapping g to a tree (say, t'), hp sends *O-JOIN-ACK*$(g,\ t')$ back to mp. If g is not a new group and $t \neq t'$, an *O-SWITCH*$(g,\ t,\ t')$ message is sent via multicast (using tree t) to all other members of g to switch g from tree t to t'.

Algorithm 1 GTMatch(g, t, T_e, b_{th})
```
// t is the current tree for g
// T_e is the set of all existing trees
if t is not null AND t covers g AND δ(t) ≤ bth
then
  return t
else
  T_c ← ∅ // T_c is the set of candidate trees
  compute a native multicast tree t_0(g) for g
  for t' ∈ T_e do
    if t' covers g AND δ(t') ≤ bth then
      T_c ← T_c ∪ {t'}
    end if
  end for
  if T_c is not empty then
    return t' ∈ T_c with min C(t')
  else
    return t_0(g)
  end if
end if
```

We use two examples in Fig. 2 to explain OLAMP. In these two examples, g_0 and g_1 originally have their own trees t_0 and t_1, respectively. In Fig. 2a, when a user requests to join g_1 via member proxy E, E sends an *O-JOIN*(g_1) message to the host proxy B. After conducting group-tree matching, B finds out that tree t_0 can also be used to cover group g_1; therefore, it issues an *O-JOIN-ACK*$(g_1,\ t_0)$ to E and an *O-SWITCH*$(g_1,\ t_1,\ t_0)$ to the remaining proxies of g_1.

3. It is important to note that the cost metric is slightly different for IP multicast and overlay multicast. For example, if assuming each link has the same bandwidth cost to deliver a unit of data, the cost of a tree for IP multicast is proportional to the total number of links in the tree. On the other hand, for overlay multicast, the cost is proportional to the number of underlying physical links in the overlay multicast tree instead of the number of overlay links.

As E does not belong to tree t_0, it sends an *O-GRAFT*(t_0) message toward the host proxy in order to graft onto t_0. Proxies A and D only need to update the group-tree matching information for g_1. Note that since no groups belong to t_1 at this time, a prune process will also be triggered to prune member proxies A and D from t_1 (which is omitted for simplicity). In Fig. 2b, when all the users connected to proxy D leave g_1, D needs to notify host proxy B by sending an *O-LEAVE*(g_1) message. After receiving the *O-LEAVE-ACK*(t_1) from B, D then prunes itself from t_1 using an *O-PRUNE*(t_1) message.

3.3 Cluster Formation Outside MSON

3.3.1 Member Proxy Selection

On receiving a list of candidate member proxies from the MSON DNS server, an end user selects one proxy based on the criteria of low latency and low workload (in terms of processing power and bandwidth availability), since low latency is critical to real-time applications and lightly-loaded proxies tend to have more resources.

The end user measures the RTT (round-trip time) to candidate proxies by sending *ping* requests. In the reply messages, the proxies piggyback their workload information, e.g., the total number of end users they handle or the total amount of access bandwidth in use. The end user then discretizes the measured RTT values into predetermined levels. If there are multiple proxies close by, the one with the lowest workload will be selected.

3.3.2 P2P Multicast in Access Networks

Outside MSON, end users associated with the same member proxy form a cluster. In a cluster, nodes communicate in a P2P fashion via application-layer multicast trees: When an end user sends packets to the group, the packets are transmitted to all other end users in the same cluster and to the member proxy as well. The member proxy will relay these packets via the aggregated tree to other member proxies (at the edge of the MSON), which will in turn deliver the data to group members in their clusters.

Due to the existence of a member proxy node in every cluster, we adopt a core-based approach (similar to ALMI [24]) to construct P2P multicast trees. In a cluster, the member proxy acts as a core, storing the group membership information in its cluster scope. Periodically, end users monitor their peers in the same cluster regarding path quality (such as delay and available bandwidth), and report this information to their member proxies. For the scalability issue, the users can monitor a fraction of randomly selected peers and switch them after some period of time. After putting together the global picture of its clusters, the member proxy computes P2P multicast delivery trees and disseminates $(parent, children)$ entries to its members. Finally, end users connect with their children and transmit data packets via unicast. If a member leaves ungracefully, its peers will detect this from periodic probing and the multicast tree will be repaired by the member proxy.

4 Overlay Network Provisioning

In the previous section, we describe TOMA with the assumption that MSON is already constructed by the ISP. Namely, proxies have been strategically deployed and overlay link bandwidth has been purchased. In this section, we discuss the overlay network provisioning problems. Obviously, the deployment of MSON is a capital-intensive investment. Thus, it is very imperative to carefully design MSON so that the ISP can make the best revenue for its investment. In this section, we design algorithms for overlay network design by considering traffic patterns (e.g., group and member distributions), group bandwidth requirements, end-to-end delay concerns, as well as multicast routing algorithms.

4.1 Problem Formulation

We model the physical network topology as an undirected graph $G = (V, E)$, where V and E denote the sets of network nodes (or routers) and physical links, respectively.[4] The total number of routers in the network is denoted as n. Each link $e \in E$ has a bandwidth capacity $c(e)$. We also denote the set of all possible paths in G as Ω. From long-term measurements, we can obtain a set of groups $\{g_i\}$ with group member distribution and bandwidth requirements [3], [9], [13], and use this information for MSON design. We denote the number of members (from all groups) that are connected to router i as w_i.

The overlay provisioning problem can be formulated as follows: Given the set of groups $\{g_i\}$, and a physical network topology $G = (V, E)$, find a virtual topology $G' = (V', E')$ on top of G (where $V' \subset V$ and $E' \subset \Omega$), in which each $e' \in E'$ is assigned a bandwidth $b(e')$, such that G' can accommodate all the groups, and the cost of G' and the average end-to-end delay of group members are minimized. The determination of the cost of G' depends on the MSON management policy. In most cases, we can use the sum of the assigned bandwidth (purchased from underlying network service providers) to estimate bandwidth cost, and the number of overlay links (i.e., $|E'|$) to measure overlay maintenance overhead.

Clearly, to obtain G', the overlay ISP needs to make three decisions: 1) determine the locations of the proxy nodes, i.e., select V', 2) select the overlay connections between these proxies, i.e., choose E', and 3) compute the bandwidth to be reserved on each overlay link, i.e., assign a bandwidth $b(e')$ to each $e' \in E'$. Since it is difficult to achieve the above optimization goals simultaneously, we divide the whole problem into three subproblems: overlay proxy placement, overlay link selection, and bandwidth dimensioning. Solutions obtained this way may be "suboptimal," as this approach is mainly concerned with the problem manageability. In the following, we present algorithms to solve these problems.

4.2 Overlay Proxy Placement

For the proxy placement problem, we limit the total number of overlay proxies, considering the deployment cost of proxies. Since end users tend to receive data packets from the closest proxies, the locations of overlay proxies directly affect the data transmission latency. Intuitively, if a router is connected to a lot of users, a proxy should be placed near

4. We assume the MSON provider can obtain the knowledge about underlying network topology from the network service provider.

this router to reduce the average delay. Hence, we can minimize the total delay between users and their proxies by intelligently placing proxies.

The *Overlay Proxy Placement* can be formulated as follows: Given the number of group members w_i $(1 \leq i \leq n)$ for all n routers and the shortest distance d_{ij} between any two routers i and j $(1 \leq i, j \leq n)$, find no more than K $(1 \leq K \leq n)$ routers as proxies, such that the weighted sum of distance from each router to its nearest proxy is minimized.

Lemma 1. *The Overlay Proxy Placement problem is NP-complete.*

Proof. If we fix the number of chosen proxies in the Overlay Proxy Placement problem, we obtain the p-Median problem, i.e., in graph G, determining p "median" nodes such that the sum of the distance between each remaining node and its nearest median node is minimized. p-Median problem has been proven to be NP-complete [20]; thus, the Overlay Proxy Placement problem is also NP-complete. □

To solve this problem, we present a greedy approximation algorithm.[5] In each step, a router r is selected as a proxy if its selection can reduce the weighted sum of distance (i.e., the objective function) by the largest amount. This procedure repeats until the maximum number of overlay proxies K is reached. In each iteration, computing the weighted sum of distance for each router requires $O(n)$ time, and the whole iteration takes $O(n^2)$ time. Therefore, the time complexity of the algorithm is $O(Kn^2)$.

This algorithm tries to minimize the objective function at each step, but it may be stuck in local minima. In Section 5, we show that this algorithm can achieve competitive performance in comparison with the optimal solution obtained by ILP (Integer Linear Programming). Due to space limitation, we include the ILP formulation in an extended version of this paper [22].

4.3 Overlay Link Selection

Once the proxy locations have been determined, the next step is to connect these proxies into a mesh, on top of which overlay multicast trees will be constructed. There are two potential optimization goals, namely, minimizing the end-to-end latency and minimizing the overlay maintenance overhead. The end-to-end latency can be measured by the average number of physical hops, and the overlay maintenance overhead can be approximately represented by the number of overlay links (i.e., $|E'|$).

Existing solutions focus on either one of the goals, but not both. For example, Complete Graph and Adjacent connection [23] have been proposed to optimize the end-to-end delay. The former approach establishes an overlay link between every proxy pair, whereas the latter approach selects an overlay link between two proxies only if the network-layer path does not go through other intermediate proxies, thus removing "redundant" overlay links. Unlike these approaches, the k-Minimum Spanning Tree (MST) approach [29] aims to reduce the cost of maintaining the overlay by using k least-overlapping MSTs to connect proxies, though at the expense of higher delay.

In this paper, we try to reconcile the trade-offs of these approaches. We adopt a delay threshold to bound the delay penalty caused by removing overlay links, and then minimize the overlay maintenance overhead. We formulate the following *Overlay Link Selection* problem: Given a delay threshold D, minimize the total number of selected overlay links such that for every pair of overlay proxies i and j, the percentage of increased shortest path delay in the overlay $G'(V', E')$ versus the original network $G(V, E)$ is no larger than D: $\frac{d_{ij}(G')-d_{ij}(G)}{d_{ij}(G)} \leq D$ $(i, j \in [1, n'], i \neq j)$, where $d_{ij}(G)$ denotes the delay of the shortest path between i and j on graph G, and n' is the total number of proxies. In fact, when $D = \infty$, this problem minimizes the number of overlay links irrespective of the delay and it can be solved by 1-MST. When $D = 0$, it minimizes the number of overlay links while maintaining the smallest end-to-end delay, and Adjacent Connection gives the solution to this problem. Therefore, our problem formulation and solutions provide higher flexibility for overlay providers to tune the trade-offs between overlay maintenance overhead and end-to-end delay.

Lemma 2. *The Overlay Link Selection problem is NP-complete.*

Proof. To show that this problem is in NP, we can restate this optimization problem as a decision problem: We want to determine if we can choose K overlay links such that the overlay network is connected, and the increase of overlay path delay between any proxy pair is bounded by D. Suppose we are given K overlay links, we can validate if these links satisfy the above requirements in polynomial time.

We prove the NP-hardness of the Overlay Link Selection problem by restricting it to a d-Spanner problem with unit cost and arbitrary (polybounded) length [15]. We first limit the number of proxies n' to be equal to the number of routers n. We then create a complete graph G'' of G, and denote the maximum value among $d_{ij}(G)(1 + D)$ $(i, j \in [1, n'], i \neq j)$ as D''. Any solution E' for this restricted Overlay Link Selection problem is a solution for the D''-Spanner problem. Conversely, any solution E'' for the D''-Spanner problem is a solution for the restricted Overlay Link Selection problem. In the equivalent D''-Spanner problem, all the edges have unit cost (i.e., all the overlay links have the same maintenance cost), and the edge length varies (the delay of each overlay link depends on the number of underlying physical links) and is clearly polybounded. In [15], it has been proven that a d-Spanner problem with unit cost and arbitrary length is NP-hard, so we can conclude that the restricted Overlay Link Selection problem is NP-hard. Therefore, the original Overlay Link Selection problem is NP-complete. □

Though there are approximation algorithms (with approximation ratio $O(n \log d)$) for the general d-spanner problem with unit cost and arbitrary length [15], it is unclear if these algorithms can be applied to the Overlay Link Selection problem. In this paper, we propose solutions

5. It should be noted that there exist many approximation algorithms for the p-Median problem, which may potentially be used to solve the Overlay Proxy Placement problem. Since the main purpose of this paper is to formulate the problem and examine how the overlay dimensioning affects the performance of TOMA, we defer the comparison of existing algorithms with ours to future work.

for this problem. Again, we would point out that it is worth investigating other approaches and examining their performance, which is nevertheless out of the scope of this paper.

Before presenting our solutions, we define some notations. An **overlay link** is the shortest IP-layer path connecting two proxies, or a **proxy pair**. An **overlay path** is composed of overlay links and it connects two proxies. For an overlay network with n' overlay proxies, there are a total of p proxy pairs and p overlay links, where $p = \frac{n'(n'-1)}{2}$. We rank the proxy pairs and overlay links from 1 to p, respectively. For every proxy pair i $(1 \leq i \leq p)$, we denote each candidate overlay path between them that satisfies the delay constraint as $\langle i, j \rangle$ $(1 \leq j \leq np_i)$, where np_i is the number of candidate overlay paths for i.

In our solutions, we first enumerate for each proxy pair the set of candidate overlay paths satisfying the delay threshold. For each overlay path, we find out the overlay links which the path is composed of. Then, we can select a set of the overlay links to minimize its cardinality. In the following, we propose an ILP formulation and a greedy algorithm.

4.3.1 ILP Formulation

We define the following variables:

$$x_i = \begin{cases} 1, & \text{if the } i\text{th overlay link is selected} \\ 0, & \text{otherwise} \end{cases}$$
$$\forall i \in [1, p],$$

and

$$y_{ij} = \begin{cases} 1, & \text{if the } j\text{th overlay path for the } i\text{th} \\ & \text{proxy pair is selected} \\ 0, & \text{otherwise} \end{cases}$$
$$\forall i \in [1, p], \forall j \in [1, np_i].$$

The objective is to minimize the total number of selected overlay links, i.e., $\sum_{i=1}^{p} x_i$, subject to the following constraints:

1. Every proxy pair i is covered by at least one overlay path:
$$\sum_{j=1}^{np_i} y_{ij} \geq 1 \quad \forall i \in [1, p].$$
2. If an overlay path is selected, each overlay link on the path must be selected:
$$y_{ij} \leq x_k \quad \forall i \in [1, p], \forall j \in [1, np_i], \forall k \in [1, nl_{ij}],$$
where nl_{ij} denotes the number of overlay links on an overlay path $\langle i, j \rangle$.
3. The delay constraint:
$$\frac{d_{ij}(G') - d_{ij}(G)}{d_{ij}(G)} \leq D \quad \forall i, j \in [1, n'], i \neq j.$$

4.3.2 Greedy Algorithm

To reduce the complexity of solving the above ILP, we propose a greedy algorithm. For each overlay link l, we define a utility:

$$u(l) = \sum_{\langle i,j \rangle \in op_l} \frac{1}{np_i} \times \frac{1}{nl'_{ij}},$$

where op_l is the set of overlay paths that contain the overlay link l and connect two unconnected proxies, nl'_{ij} is the number of unselected overlay links on an overlay path $\langle i, j \rangle$. Intuitively, an overlay link should have higher priority to be selected, if it is contained in more candidate overlay paths or it connects a proxy pair with fewer and shorter candidate overlay paths. Hence, the link utility is determined by the number of unselected overlay links on a candidate overly path and the number of candidate overlay paths between a proxy pair.

In the Greedy algorithm, we first compute the utility function for each overlay link. Then, we repeat the following steps until all the proxies are connected. In each iteration, we select an overlay link with the highest utility. If all the overlay links for an overlay path are selected, the corresponding proxy pair is considered to be connected. Then, the utilities for the remaining overlay links are updated.

The Greedy algorithm is not optimal in that it heuristically selects overlay links according to a utility function. Nevertheless, we will show that it can yield near-optimal solutions in Section 5.

4.4 Bandwidth Dimensioning

Having determined the overlay topology (overlay proxies and links), we then need to decide the bandwidth required on overlay links to accommodate the groups obtained from long-term measurements.

Bandwidth dimensioning is complex since it is tightly related to multicast routing algorithms. Thus, we suggest a simulation-based approach to take multicast routing algorithms into account. The basic idea is: Compute multicast trees based on the given routing algorithm for all the groups, and then by summing up the traffic volume on each tree, the amount of bandwidth to be leased on every overlay link can be determined.

Based on long-term measurement results, the computed bandwidth is sufficient to satisfy the requirement of these groups on average. However, the traffic peak rate is likely to exceed the average rate. Hence, the MSON provider should consider overdimensioning the network by reserving extra bandwidth. If the bandwidth is still insufficient during peak hours, the MSON can either reject some groups or lease bandwidth from higher-tier ISPs for short-term usage (possibly at a higher price).

5 Performance Evaluation

In this section, we first compare the performance of TOMA with two representative application-layer multicast protocols NICE [5] and NARADA [12], and an IP multicast protocol (Core-Based Tree [4]) using NS-2. Then, we evaluate the effectiveness of our overlay provisioning algorithms.

5.1 Simulation Settings

We use two types of network topologies and group membership models. The first type is synthetic Transit-Stub (TS) topologies [7] with 50 transit domain routers and

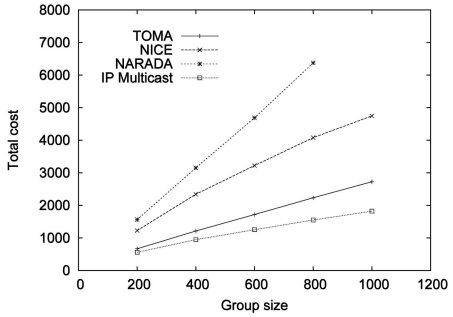


Fig. 3. Tree cost versus group size in Transit-Stub.

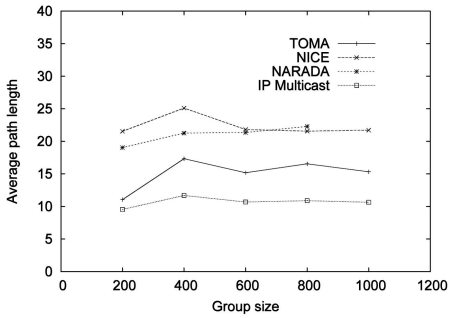


Fig. 4. Average path length versus group size in Transit-Stub.

500-2,000 stub domain routers. End hosts are attached to stub routers uniformly at random. The second type of network topology is abstracted from a real network topology, AT&T IP backbone [1], and it consists of 54 nodes. To each router i, we assign a weight w_i and randomly attach end hosts to the router with a probability proportional to w_i.

We use simple overlay provisioning heuristics to evaluate the performance of TOMA with unoptimized backbone overlay networks (the overlay provisioning algorithms are evaluated separately). We randomly select 80 percent of the transit nodes (i.e., 40 nodes) in TS topologies and nine gateway routers in the AT&T topology as proxy nodes, because transit nodes and gateway routers usually have high degrees and are located in the core of the network. The overlay links are constructed using the Adjacent Connection described in Section 4.3.

5.2 Multicast Tree Performance

To test the scalability of different schemes, we focus on large group sizes of 200 to 1,000 members. End hosts join the multicast group during an interval of 400 seconds and the session ends at 1,000 seconds. We collect the metrics after the multicast tree has stabilized. We only present the results for TS topologies since AT&T yields similar results [22].

5.2.1 Multicast Tree Cost

Multicast tree cost measures the number of links in a multicast tree. It quantifies the bandwidth efficiency of multicast routing schemes. In Fig. 3, we plot the average tree cost as group size increases. Clearly, the performance of TOMA is comparable to IP multicast. In addition, TOMA outperforms NICE and NARADA in all cases, and their difference magnifies as group size is increased. This efficiency gain of TOMA versus NICE and NARADA is due to two reasons. First, TOMA takes advantage of the carefully provisioned overlay network which resembles the underlying network topology, whereas NICE and NARADA rely on path probing techniques and construct overlay topologies with degraded quality. Second, by using proxies as intermediate nodes, TOMA allows packets to be replicated at proxies and, hence, decreases the redundant packets sent over the physical links.

5.2.2 Average Path Length

Path Length is the number of links on the path from the source to a member. The results for average path length are shown in Fig. 4. Because of the high computation overhead of NARADA for large groups in the simulations, we were unable to complete the simulations for NARADA when the group size is 1,000. As expected, IP multicast have the shortest end-to-end paths. The path lengths of TOMA trees are much shorter than those of NICE and NARADA trees on average. For instance, at a group size of 800, the average path lengths of TOMA, NICE, and NARADA trees are 16.5, 21.6, and 22.3, respectively. Again, the performance improvement of TOMA over NICE and NARADA is gained through efficient overlays: In TOMA, overlay links are constructed based on the shortest paths in network layer; as a result, data packets can avoid going through unnecessarily long paths.

We also found out that TOMA achieves higher performance in terms of link stress (defined as the number of identical data packets delivered over each link), which is again due to the efficient overlay construction [22].

5.3 Control Overhead

We plot the total number of control messages generated by TOMA, NICE, and NARADA during a group's life time in Fig. 5. Among the three protocols, NARADA has the highest control overhead. In addition, even though NICE is more scalable than NARADA as it uses a hierarchical structure, TOMA significantly outperforms NICE. At group size of 1,000, TOMA generates only about one third as many control messages as NICE. The reason is simple: In TOMA, the local clusters remain rather static and a member stays in the same cluster in spite of the behaviors of other members. However, in NICE, as members join and leave, the clusters change very frequently to enforce the bounds on cluster size, which induces numerous control messages. Furthermore, a NICE node needs to periodically send "heartbeat" messages to all other cluster members (or even to multiple clusters in the case of cluster leaders), whereas a TOMA node only needs to refresh the connection with its parent.

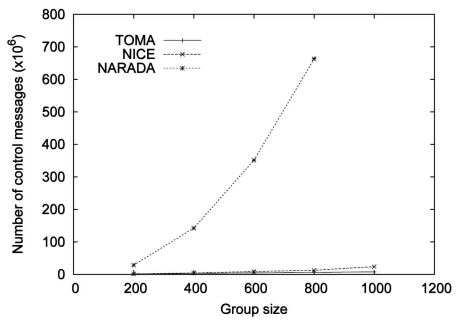


Fig. 5. Control Overhead for a single group in Transit-Stub.

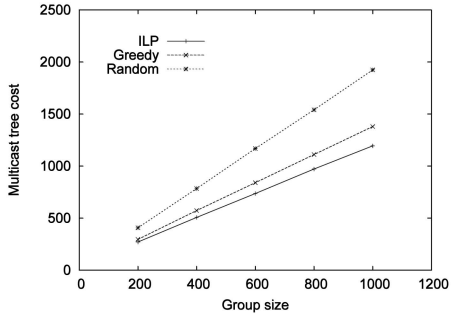


Fig. 6. Tree cost for different proxy placement.

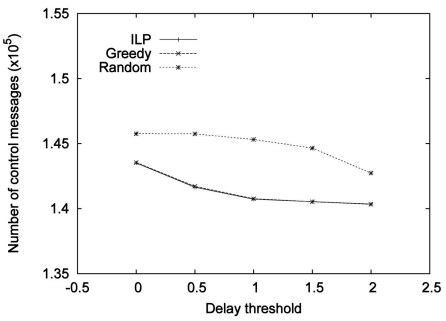


Fig. 8. Control overhead for different link selection.

5.4 Overlay Provisioning

In this section, we evaluate the overlay provisioning algorithms on the AT&T topology. We assume the properties of multicast groups can be obtained through long-term measurements, and the overlay network is constructed based on the measurement results. Unless otherwise specified, proxy nodes are placed in the locations determined by the greedy algorithm and connected using the Adjacent Connection approach.

5.4.1 Overlay Proxy Placement

We compute proxy locations for $K = 9$ based on the weights assigned to routers using the ILP and greedy algorithms, and compare the performance of multicast groups with different group sizes on the resulting overlay networks. As a reference, we also include the results when proxies are randomly selected.

The multicast tree cost as group size varies is plotted in Fig. 6. The ILP solution yields the lowest tree cost, since it optimally places proxies as close to exchange routers (which have a bulk of members) as possible and, thus, prevents the packets from traversing the links between proxies and access routers multiple times. In contrast, the random approach chooses proxies randomly from all the routers in the network. As a result, end users may use some fairly long paths to connect to the proxies, and some physical links may be used multiple times by different users. As for the greedy algorithm, it selects some gateway routers and exchange routers, so its performance lies in between.

The average end-to-end path length is depicted in Fig. 7. It is clear that both the ILP and greedy solutions reduce the average path length greatly. For example, the average path length is approximately 10.4 to 10.7 when proxies are randomly placed, and it is reduced to below 7.0 by the ILP or greedy approach. This remarkable improvement is again due to the fact that the two latter approaches place proxies closer to end users to avoid using unnecessarily long paths for data delivery.

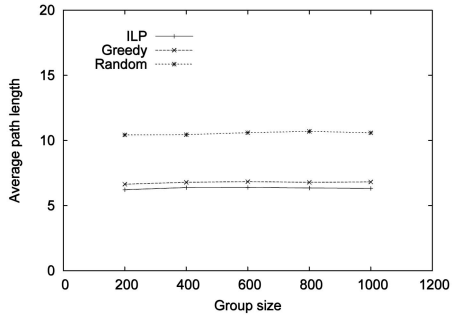


Fig. 7. Average path length for different proxy placement.

5.4.2 Overlay Link Selection

For overlay link selection, we vary the delay threshold and plot the control overhead of TOMA on the overlay networks generated by the ILP and Greedy algorithms in Fig. 8. We also include the results of a Random algorithm, which randomly selects overlay links until all overlay proxies are connected without exceeding the delay threshold. For each delay threshold, we conduct 10 simulations for the Random algorithm and compute the average results. The average multicast group size is fixed at 200.

As shown in Fig. 8, when the delay threshold increases, all of the three algorithms has lower control overhead, as larger delay threshold allows longer overlay paths which have more overlaps and use fewer overlay links overall. In addition, Greedy gives the optimal solutions on the AT&T topology. By contrast, the random algorithm does not work as well, as it does not differentiate overlay links that can be used in many candidate overlay paths from those that cannot.

5.4.3 Bandwidth Dimensioning

To determine the bandwidth for each overlay link, we model the traffic pattern as follows: 1) for every group, a proxy has attached end users participating in the group with a given probability as described in Section 5.1, 2) groups arrive according to a Poisson process, and the average life time is exponentially distributed, and 3) each group requires the same bandwidth throughout its lifetime, and the bandwidth requirements for different groups follow a certain distribution. In the simulations, we define three types of multicast groups: 50 percent of the groups are low bandwidth (10K), 30 percent are medium bandwidth (100K), and 20 percent are high bandwidth (1M).

We generate sample traces of multiple groups. For each trace, we compute aggregated trees for the groups using the group-tree matching algorithm and determine the bandwidth required on each link averaged for different sampling points. Then, we overdimension the bandwidth by a certain amount and validate the dimensioning results by measuring *group join request rejection ratio* for another trace with the same traffic pattern. A join request of a multicast group is rejected if the residual bandwidth of an overlay link is not enough to accommodate the multicast tree computed for that group.

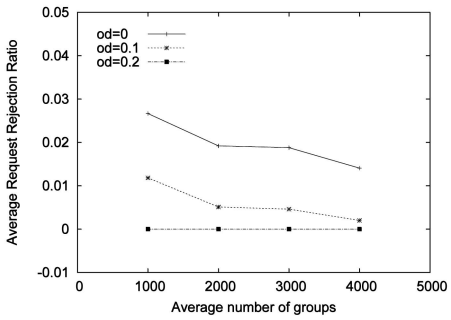


Fig. 9. Request rejection ratio for bandwidth dimensioning.

Fig. 9 shows the average request rejection ratio when the percentage of overdimensioning (denoted as *od*) and the number of groups are varied. The rejection ratio remains very low for different numbers of coexisting groups, even without overdimensioning. It is not 0 because the bandwidth is reserved based on the average traffic rate instead of the peak rate. As *od* increases, the rejection ratio decreases, because more bandwidth is reserved. The rejection ratio decreases when there are more groups in the network, because the a larger number of groups can have better bandwidth multiplexing. These results indicate that our bandwidth dimensioning scheme is indeed very effective.

5.5 Summary

Our observations through simulation experiments can be summarized as follows: TOMA creates multicast distribution trees with quality almost comparable to IP multicast trees; the control overhead of TOMA is significantly less than NICE and NARADA for large groups; the performance of TOMA can be improved significantly by the proposed overlay provisioning algorithms. Our additional results in [22] demonstrate that TOMA is more robust than NICE and NARADA when there are ungraceful user leaves and TOMA is scalable to large numbers of groups in terms of control overhead and multicast state.

If we recall the architecture difference between TOMA and application layer multicast, it is not difficult to understand why TOMA outperforms NICE and NARADA by a large margin. In TOMA, the strategic MSON provisioning takes physical network topologies into account and, thus, provides efficient multicast data delivery. In addition, a large amount of probing overhead can be saved due to the simple clustering technique. Finally, the aggregated multicast approach further makes TOMA scalable to large numbers of groups. In contrast, in application layer multicast, the control messages for maintaining groups and trees will increase rapidly with group size. Due to limited bandwidth at end systems, application layer multicast trees tend to have larger depth and longer data latency. Nevertheless, we want to point out that TOMA requires infrastructure support to achieve the above benefits, which is a trade-off of TOMA versus NICE and NARADA.

6 Conclusions

In this paper, we propose and develop a two-tier overlay multicast architecture (TOMA) to support group communication applications in an efficient and scalable way. Our contributions can be summarized as follows:

1. We advocate the notion of infrastructure-supported overlays to facilitate the deployment of multicast service.
2. We provide a viable architecture design of TOMA, which adopts MSON as the backbone service domain and P2P multicast in the access domains to achieve efficient resource utilization with reduced control overhead.
3. We develop OLAMP for MSON management. The control overhead for establishing and maintaining multicast trees are significantly reduced, and far less forwarding state needs to be maintained at proxy nodes.
4. To efficiently plan the backbone service overlay, we suggest several provisioning algorithms to locate proxies, select overlay links, and allocate link bandwidth.
5. By extensive simulation studies, we show the promising performance of TOMA and demonstrate the effectiveness of our provisioning algorithms. We believe that the invention of our practical, comprehensive, and scalable multicast service model would significantly facilitate the multicast wide deployment.

Acknowledgments

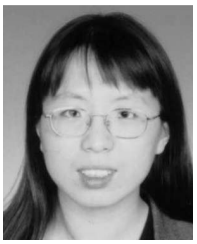
This material is based upon work supported by the US National Science Foundation under Grant No. 0435515 and Grant No. 0435230.

References

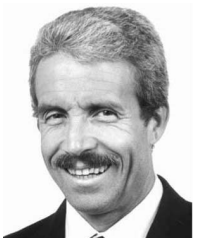
[1] AT&T IP Backbone, http://www.ipservices.att.com/backbone/, 2001.
[2] K. Almeroth, "The Evolution of Multicast: From the MBone to Inter-Domain Multicast to Internet2 Deployment," *IEEE Network,* vol. 14, no. 1, pp. 10-20, Jan./Feb. 2000.
[3] K. Almeroth and M. Ammar, "Multicast Group Behavior in the Internet's Multicast Backbone (MBone)," *IEEE Comm.,* vol. 35, no. 6, pp. 124-129, June 1997.
[4] A. Ballardie, "Core Based Trees (CBT Version 2) Multicast Routing: Protocol Specification," IETF RFC 2189, Sept. 1997.
[5] S. Banerjee, C. Kommareddy, and B. Bhattacharjee, "Scalable Application Layer Multicast," *Proc. ACM SIGCOMM,* pp. 205-217, Aug. 2002.
[6] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an Efficient Overlay Multicast Infrastructure for Real-Time Applications," *Proc. IEEE INFOCOM,* vol. 2, pp. 1521-1531, Apr. 2003.
[7] K. Calvert, E. Zegura, and S. Bhattacharjee, "How to Model an Internetwork," *Proc. IEEE INFOCOM,* vol. 2, pp. 594-602, Mar. 1996.
[8] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE J. Selected Areas in Comm.,* vol. 20, no. 8, pp. 1489-1499, Oct. 2002.
[9] R. Chalmers and K. Almeroth, "Modeling the Branching Characteristics and Efficiency Gains of Global Multicast Trees," *Proc. IEEE INFOCOM,* vol. 1, pp. 449-458, Apr. 2001.
[10] Y. Chawathe, S. McCanne, and E.A. Brewer, *An Architecture for Internet Content Distributions as an Infrastructure Service,* 2000, unpublished, http://www.cs.berkeley.edu/yatin/papers/.
[11] Y. Chawathe, S. McCanne, and E.A. Brewer, "RMX: Reliable Multicast for Heterogeneous Networks," *Proc. IEEE INFOCOM,* vol. 2, pp. 795-804, Mar. 2000.
[12] Y.-H. Chu, S.G. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM Sigmetrics,* pp. 1-12, June 2000.
[13] J.-H. Cui, M. Faloutsos, D. Maggiorini, M. Gerla, and K. Boussetta, "Measuring and Modelling the Group Membership in the Internet," *Proc. ACM SIGCOMM/USENIX Internet Measurement Conf. (IMC '03),* pp. 65-77, Oct. 2003.

[14] C. Diot, B. Levine, J. Lyles, H. Kassem, and D. Balensiefen, "Deployment Issues for the IP Multicast Service and Architecture," *IEEE Network,* vol. 14, no. 1, pp. 78-88, Jan./Feb. 2000.
[15] Y. Dodis and S. Khanna, "Design Networks with Bounded Pairwise Distance," *Proc. 31st Ann. ACM Symp. Theory of Computing (STOC),* pp. 750-759, 1999.
[16] Z. Duan, Z.-L. Zhang, and Y.T. Hou, "Service Overlay Networks: SLAs, QoS, and Bandwidth Provisioning," *IEEE/ACM Trans. Networking,* vol. 11, no. 6, pp. 870-883, Dec. 2003.
[17] A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos, "Aggregated Multicast: An Approach to Reduce Multicast State," *Proc. Sixth Global Internet Symp. (GI '01),* vol. 3, pp. 1595-1599, Nov. 2001.
[18] P. Francis, "Yoid: Extending the Multicast Internet Architecture," white paper, http://www.aciri.org/yoid/, 2007.
[19] J. Jannotti, D.K. Gifford, K.L. Johnson, M.F. Kaashoek, and J.W. O'Toole Jr., "Overcast: Reliable Multicasting with an Overlay Network," *Proc. USENIX Symp. Operating Systems Design and Implementation,* pp. 197-212, Oct. 2000.
[20] O. Kariv and L. Hakimi, "An Algorithmic Approach to Network Location Problems. ii: The p-Medians," *SIAM J. Applied Math.,* vol. 37, no. 3, pp. 539-560, Dec. 1979.
[21] M. Kwon and S. Fahmy, "Topology Aware Overlay Networks for Group Communication," *Proc. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV '02),* pp. 127-136, May 2002.
[22] L. Lao, J.-H. Cui, and M. Gerla, "A Scalable Overlay Multicast Architecture for Large-Scale Applications," Technical Report CSD TR040008, Univ. of California Los Angeles, http://www.cs.ucla.edu/NRL/hpi/AggMC/index.html, Feb. 2004.
[23] Z. Li and P. Mohapatra, "The Impact of Topology on Overlay Routing Serive," *Proc. IEEE INFOCOM,* vol. 1, pp. 408-418, Mar. 2004.
[24] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure," *Proc. Third USENIX Symp. Internet Technologies and Systems,* pp. 49-60, Mar. 2001.
[25] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," *Proc. Conf. Networked Group Comm. (NGC),* pp. 14-29, Nov. 2001.
[26] S. Shi and J.S. Turner, "Routing in Overlay Multicast Networks," *Proc. IEEE INFOCOM,* vol. 3, pp. 1200-1208, June 2002.
[27] S. Shi, J.S. Turner, and M. Waldvogel, "Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks," *Proc. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV '01),* pp. 83-92, June 2001.
[28] A. Sobeih, W. Yurcik, and J.C. Hou, "VRing: A Case for Building Application-Layer Multicast Rings (Rather Than Trees)," *Proc. IEEE Computer Soc. 12th Ann. Int'l Symp. Modeling, Analysis, and Simulation of Computer and Telecomm. Systems (MASCOTS '04),* Oct. 2004.
[29] A. Young, C. Jiang, M. Zheng, A. Krishnamurthy, L. Peterson, and R.Y. Wang, "Overlay Mesh Construction Using Interleaved Spanning Trees," *Proc. IEEE INFOCOM,* vol. 1, pp. 396-407, Mar. 2004.
[30] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J.D. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination," *Proc. Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV '01),* pp. 11-20, June 2001.

Li Lao received the BS degree from Fudan University, China, in 1998. She received the MS and PhD degrees in computer science from the University of California Los Angeles in 2002 and 2006, respectively. She joined Google Inc. in April 2006. Her research focuses on multicasting, overlay network management, multicast modeling, and performance evaluation. She is a member of the IEEE.



Jun-Hong Cui received the BS degree in computer science from Jilin University, China, in 1995, the MS degree in computer engineering from the Chinese Academy of Sciences in 1998, and the PhD degree in computer science from the University of California Los Angeles in 2003. Currently, she is an assistant professor in the Computer Science and Engineering Department at the University of Connecticut. Her research interests are in computer networks and data communications. They cover the protocol design, network modeling, and performance evaluation of the Internet, wireless networks, sensor networks, peer-to-peer networks, and overlay networks. Currently, her research mainly focuses on multicasting and QoS support in wired and wireless networks, algorithm and protocol design in large-scale mobile wireless sensor networks, and mobility modeling and management in mobile ad hoc networks. Please see http://www.cse.uconn.edu/~jcui/ for her recent projects and publications. She is a member of the IEEE.



Mario Gerla received the graduate degree in engineering from the Politecnico di Milano in 1966, and the MS and PhD degrees in engineering from the University of California Los Angeles in 1970 and 1973, respectively. After working for the Network Analysis Corporation from 1973 to 1976, he joined the faculty of the Computer Science Department at the University of California Los Angeles, where he is now a professor. His research interests cover the performance evaluation, design, and control of distributed computer communication systems, high-speed computer networks, wireless LANs, and ad hoc wireless networks. He has worked on the design, implementation, and testing of various wireless ad hoc network protocols (channel access, clustering, routing, and transport) within the DARPA WAMIS, GloMo projects, and, most recently, the ONR MINUTEMAN project. He is also conducting research on QoS routing, multicasting protocols, and TCP transport for the Next Generation Internet (see www.cs.ucla.edu/NRL for recent publications). He is a fellow of the IEEE.

Shigang Chen received the BS degree in computer science from the University of Science and Technology of China in 1993. He received the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. After graduation, he worked for Cisco Systems for three years before joining the University of Florida as an assistant professor in the Department of Computer and Information Science and Engineering. His research interests include network security, quality of service, and sensor networks. He received the IEEE Communications Society Best Tutorial Paper Award in 1999. He was a guest editor for *ACM/Baltzer Journal of Wireless Networks* (*WINET*) and *IEEE Transactions on Vehicle Technologies*. He is serving as a Technical Program Committee cochair for the Computer and Network Security Symposium of IEEE IWCCC 2006. He served as a vice Technical Program Committee chair for IEEE MASS 2005, a vice general chair for QShine 2005, a Technical Program Committee cochair for QShine 2004, and a Technical Program Committee member for many conferences including IEEE ICNP, IEEE INFOCOM, IEEE SANS, IEEE ISCC, IEEE Globecom, etc. He is a member of the IEEE.