TOMA: A Viable Solution for Large-Scale Multicast Service Support *

Li Lao[1], Jun-Hong Cui[2], and Mario Gerla[1]

[1] Computer Science Dept., University of California, Los Angeles, CA 90095
[2] Computer Science & Engineering Dept., University of Connecticut, CT 06029

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 the 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 networks. Simulation studies are conducted and the results demonstrate the promising performance of TOMA.

1 Introduction

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 scale well to support very large groups. However, IP multicast is still far from being widely deployed due to many technical reasons as well as marketing reasons. The most critical reasons include: the lack of a scalable inter-domain routing protocol, the state scalability issue with a large number of groups, the lack of support in access control, the requirement of global deployment of multicast-capable IP routers and the lack of appropriate pricing models [10, 1]. These issues make Internet Service Providers (ISPs) reluctant to deploy and provide multicast service.

In recent years, researchers resort to application-layer multicast approach: multicast-related features are implemented at end hosts [12, 9, 16, 3, 14, 19, 17, 20, 6]. Data packets are transmitted between end hosts via unicast, and are replicated at end hosts. These systems do not require infrastructure support from intermediate nodes (such as routers), and thus can be easily deployed. However, application-layer multicast is generally not scalable to support large multicast groups due to its relatively low bandwidth efficiency and heavy control

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

overhead caused by tree maintenance at end hosts. In addition, from the point of view of an ISP, this approach is hard to have an effective service model to make profit: group membership and multicast trees are solely managed at end hosts, thus it is difficult for the ISP to have efficient member access control and to obtain the knowledge of the group bandwidth usage, and makes an appropriate pricing model impractical, if not impossible.

Then the question is: what is the viable or practical solution for large-scale multicast support? In a multicast service, multiple parties are involved: network service providers (or higher-tier ISPs), Internet Service Providers (or lower-tier ISPs, which we refer to as ISPs for short in this paper), and end users. However, which party really cares about multicast? End users do not care as long as they can get their service at a reasonable price. This is why many network games are implemented using unicast. Network service providers do not care as long as they can sell their connectivity service with a good price. Obviously, ISPs in the middle are the ones who really care about multicast: their goal is to use limited bandwidth purchased from network service providers to support as many users as possible, i.e., to make a biggest profit. Therefore, to develop a practical, comprehensive, and profitable multicast service model for ISPs is the critical path to multicast wide deployment.

Strongly motivated, in this paper, we propose a **T**wo-tier **O**verlay **M**ulticast **A**rchitecture (called **TOMA**) to provide 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. MSON consists of service nodes or proxies which are strategically deployed by an MSON provider (ISP). The design of MSON relies on well-defined business relationships between the MSON provider, network service providers (i.e., the underlying network domains), and group coordinators (or initiators): the MSON provider dimensions its overlay network according to user requests (provided by long-term measurement), purchases bandwidth from the network service providers based on their service level agreements (SLAs), and sells its multicast services to group coordinators via service contracts. Outside MSON, end hosts (group members) subscribe to MSON by transparently connecting to some special proxies (called *member proxies*) advertised by the MSON provider. Instead of communicating with its member proxy using unicast, an end host could form a cluster with other end hosts close by. In the cluster, application-layer multicast is used for efficient data delivery between the limited number of end users. The end users participating in the groups only need to pay for their regular network connection service outside MSON.

To make TOMA a reality, we face many challenges: 1. **Efficient management of MSON**: An MSON is expected to accommodate a large number of multicast groups. How does an MSON provider efficiently establish and manage numerous multicast trees? 2. **Cluster formation outside MSON**: Multicast group members may disperse around the world. How should they select and subscribe to appropriate member proxies? And how are efficient clusters formed among end users? 3. **MSON dimensioning**: Given the TOMA architecture,

how should the MSON provider dimension the overlay network? In other words, where should the overlay proxies be placed, which links are used to transmit data, and how much bandwidth should be reserved on each link? 4. **Pricing**: The lack of feasible pricing schemes is one of the main barriers that delay the deployment of IP multicast. Therefore, how to charge the users of MSON is an important factor to decide whether MSON is a practical solution.

In this paper, we focus our efforts on investigating the first two issues (Note that we have also developed some effective mechanisms for overlay dimensioning and pricing [15], which are not presented here due to space limit). To address the management of MSON, we propose an efficient and scalable protocol, called OLAMP (OverLay Aggregated Multicast Protocol), in which we adopt *aggregated multicast* approach [11], with multiple groups sharing one delivery tree. Outside MSON, we develop efficient member proxy selection mechanisms, and choose a core-based application-layer multicast routing approach for data transmission inside clusters. We conduct extensive simulation studies and show the promising performance of TOMA: it provides efficient multicast service comparable to IP multicast; it outperforms NICE, a representative application layer multicast protocol, in the simulated scenarios; and it is scalable to group size as well as to the number of co-existing groups.

The rest of this paper is organized as follows. We review some background and related work in Sect. 2. In Sect. 3, we present an overview of TOMA architecture and address the critical issues of MSON management and cluster formation outside MSON. In Sect. 4, we evaluate the performance of TOMA by extensive simulations. Finally, we summarize our contribution in Sect. 5.

2 Background

2.1 Aggregated Multicast

It is known that IP multicast is plagued from the state scalability issue. Multicast state scalability problem 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) for a large number of co-existing multicast groups. Aggregated multicast is proposed to improve multicast state scalability in transit (especially backbone) domain [11]. Observing that many multicast trees within a single domain are likely to have significant overlapping when there are numerous multicast groups, aggregated multicast forces multiple groups with similar shapes to share a single tree and reduces the number of multicast trees in the transit domain. Thus, the tree management overhead is significantly reduced, and the multicast state information stored in the routers is dramatically reduced too. In this paper, we design a protocol called OLAMP within MSON based on aggregated multicast.

2.2 Related Work

There is a large body of work on application-layer multicast [12, 9, 16, 3, 14, 19, 17, 20, 6]. In Yoid [12], multicast trees are constructed by each member individ-

ually selecting a parent. In End System Multicast [9], end hosts cooperatively construct a multicast delivery tree on top of a mesh. ALMI [16] uses a centralized entity to collect membership information and to periodically calculate a minimum spanning tree. TAG [14] exploits the underlying network topology to optimize multicast trees.

Recently, the notion of infrastructure-supported overlays has received increasing attentions. Example seminal works are Overcast [13] and RMX [8]. Both of them use overlay nodes to support multicast routing, and their main focus is on building reliable multicast trees. In the literature, several two-tier architectures have also been proposed for scalable multicast support (such as OMNI [4], MSN [18], and Scattercacst [7]). However, these architectures are based on different service models from ours. 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 dimensioning. Moreover, in [18, 4], the authors focus on single multicast tree optimization and endeavor to optimize end-to-end delay and access bandwidth usage at service nodes. In our work, we put our efforts on the scalability issues for multiple co-existing groups. To our best knowledge, this paper is the first work to address the multicast state scalability issue in overlay networks.

3 TOMA: A Two-Tier Overlay Multicast Architecture

3.1 TOMA Overview

In the TOMA architecture, MSON is advocated as the service backbone domain. In MSON, overlay proxies are strategically placed to form an overlay network, on top of which multicast distribution trees are built for data delivery. Since an MSON provider always aims to have a bigger customer base and make a bigger profit, the MSON management scheme should be scalable to the group size (i.e., the number of members) as well as the number of groups. In this paper, we propose a protocol called OLAMP for efficient and scalable MSON management. In OLAMP, we adopt aggregated multicast [11]. Data packets are encapsulated at incoming proxies, transmitted on aggregated trees, and decapsulated at outgoing proxies. Outside MSON, end users subscribe to MSON by transparently connecting to **member proxies** advertised by the MSON provider. Each member proxy organizes some users into a "cluster", where an application-layer multicast tree (also denoted as peer-to-peer or P2P multicast tree in this paper) is formed for data delivery among the cluster members.

Each TOMA group is identified by a URL-like name *TOMA://groupname.xyzmson.com/*. End users (sources and receivers) explicitly send out a subscription request containing the group name, which will reach the DNS server and **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.

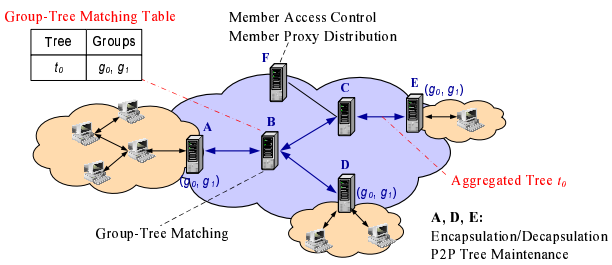

Fig. 1. A big picture of TOMA, where F is the group registry server/DNS server, B is the host proxy for groups g_0 and g_1, A, D and E are member proxies, and groups g_0 and g_1 share the aggregated tree t_0.

After finding a member proxy, the end user sends its join message to the member proxy, which will then subscribe to the multicast group inside the 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 predetermined proxy (referred to as **host proxy** for group g) to establish the overlay multicast tree in the backbone domain using OLAMP.

In the backbone domain, each group is managed by a host proxy. After receiving a join request for group g, this host proxy conducts multicast routing and group-tree matching (i.e., the procedure of assigning groups to aggregated trees) mechanisms to map group g to an aggregated tree. Considering that bi-directional trees can be used to cover groups irrespective of the location of sources, we use bi-directional aggregated trees to further reduce the number of aggregated trees. The host proxy for group g can be randomly selected or by applying a hash function (called *group-to-host* hashing function). Note that group member dynamics (i.e., 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 send join or leave request to the corresponding host proxy.

In a nutshell, in TOMA, member proxies manage P2P multicast trees in their clusters, host proxies conduct group-tree matching, and OLAMP connects member proxies and host proxies, efficiently managing aggregated multicast trees in the backbone domain. It is also possible to have some OLAMP-aware **"pure" forwarding proxies** whose main responsibility is to forward multicast data inside the backbone domain. A big picture of TOMA is illustrated in Fig. 1. In the following, we address some major design issues of TOMA in more details.

3.2 OLAMP for Efficient MSON Management

In MSON, the state scalability issue is even worse than IP multicast since the capability of proxies is usually far below that of IP routers. If a proxy manages large numbers of multicast trees, it has to maintain large forwarding tables and thus causes lookup speed to be slowed down. Furthermore, if a soft state approach

is used, heavy tree management overhead due to multicast refresh messages will be incurred. Therefore, we design OLAMP to provide scalable and efficient MSON management.

OLAMP is manipulated among member proxies, host proxies, as well as "pure" forwarding proxies. 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, in MSON, every proxy is capable of handling O-type messages and maintains a multicast routing table.

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, the host proxy hp finds or computes an appropriate 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 towards hp, and the proxies along this propagation path will update their routing tables accordingly.

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, the host proxy will remove the group-tree mapping between g and t. This group-tree matching may trigger removal of the tree t when there are no other groups mapped onto t. In this case, hp sends an *O-LEAVE-ACK(t)* messages to the tree leaves of t, which will in turn prune from the tree by sending *O-PRUNE(t)* towards hp.

A Dynamic Group-Tree Matching Algorithm When an aggregated tree is bigger than a group, data will be delivered to non-member 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 $g_i, 1 \leq i \leq n$, each of which has a native tree $t_0(g_i)$ (a native tree of a group is a "perfect" match for that group and it can be computed using multicast routing algorithms). Then the **average percentage bandwidth overhead** for 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.

When a host proxy hp receives an *O-JOIN(g)* message, it updates the corresponding entries of multicast group table and executes the QoS-Aware group-tree matching algorithm (Algorithm 1), in which we denote the given bandwidth overhead threshold as 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 members of g with enough bandwidth and the bandwidth overhead is within the threshold b_{th}), t

Algorithm 1 GTMatch(g, t, T_e, b_{th})

```
//t is 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) ≤ b_th then
  return t
else
  T_c ← ∅ //T_c is the set of candidate trees
  compute a native multicast tree t_0 for g
  for t ∈ T_e do
    if t covers g AND δ(t) ≤ b_th 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
  end if
end if
```

is used for g. In all other cases, check if any existing tree is appropriate for g. If so, the one with the minimum cost is selected. Otherwise, the native tree t_0 is used to cover g (if t_0 does not exist due to bandwidth constraint, g has to be rejected). After mapping g to a tree 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'.

3.3 Cluster Formation Outside MSON

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 message to a *ping* request, the proxy piggybacks its workload information, e.g., the total number of end users it currently handles or the total amount of access bandwidth in use. If there are multiple proxies close by, the one with the lowest workload is selected by the user.

P2P Multicast in Access Networks Outside MSON, the 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 to other member proxies (at the edge of the MSON), which will in turn deliver the data to the 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 [16]) to construct P2P multicast trees. In the cluster, the member proxy acts as a core, storing the group membership information in its cluster scope. Periodically, end users monitor its peers in the same cluster regarding the path quality (such as delay and available bandwidth), and report this information to its member proxy. After putting together the global picture of its clusters, the member proxy computes P2P multicast delivery trees and disseminates the $(parent, children)$ entries to the 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 Performance Evaluation

In this section, we conduct simulation experiments in NS-2 to evaluate the performance of TOMA. We compare TOMA with a scalable application-layer multicast protocol NICE [3], an IP multicast protocol (Core-Based Tree [2]), and unicast protocol in various network topologies. NICE [3] scales to large groups in terms of control overhead and logical hops by recursively arranging group members into clusters to form a hierarchical overlay topology, which implicitly defines a source-specific tree for data delivery. In comparison with NICE, we find that TOMA can achieve very competitive performance for large groups.

4.1 Simulation Settings

To comprehensively evaluate TOMA, we use two types of network topologies and group membership models. The first type is synthetic Transit-Stub (TS) topologies [5] with 50 transit domain routers and $500-2000$ stub domain routers. End hosts are attached to stub routers uniformly at random. The second type of topology is abstracted from a real network topology, AT&T backbone, and it consists of 54 routers. 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 [15].

For overlay dimensioning, we use some simple heuristics to determine proxy locations and overlay links. We randomly select 80% of the transit nodes (i.e., 40 nodes) in TS topologies and 8 gateway routers in AT&T topologies as proxy nodes. Note that proxy nodes can be co-located with either transit (i.e., gateway) or stub (i.e., access) nodes. We choose proxy nodes from transit nodes and gateway routers because they usually have high degree and are located in the core of the network. Between every pair of proxies, an overlay link is established if the shortest IP-layer path between them does not go through any other proxies. As a result, the constructed overlay networks tend to resemble the underlying network topologies and thus have high routing efficiency. It is worth noting that our simulation results in [15] demonstrate that overlay networks dimensioned with more systematic approaches can achieve even better performance.

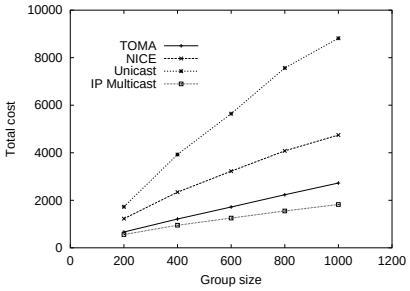

Fig. 2. Tree cost vs. group size.

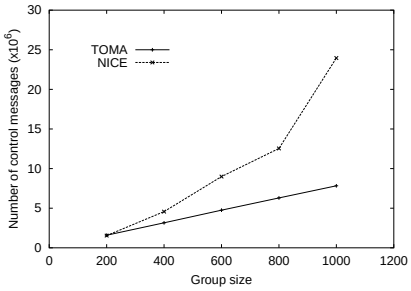

Fig. 3. Control Overhead vs. group size.

4.2 Multicast Tree Performance

In simulation experiments, we focus on large group sizes of 200 to 1000 members to test the scalability of TOMA. End hosts join the multicast group during an interval of 400 seconds, and the session ends at 1000 seconds. We collect the metrics after the multicast tree has stabilized. Due to space limitation, we only present the results for TS topologies since AT&T yields similar results [15].

In Fig. 2, we plot the average tree cost (defined as the total number of links in a multicast tree) of these protocols as group size increases. As a reference, we also include the total link cost for unicast. Clearly, the performance of TOMA is comparable to IP multicast. In addition, TOMA outperforms NICE in all cases, and their difference magnifies as group size is increased. This efficiency gain of TOMA vs. NICE is due to two reasons. First, TOMA takes advantage of the carefully dimensioned overlay network which resembles the underlying network topology, whereas NICE relies on path probing techniques and constructs overlay topologies with degraded quality. Second, by using proxies as intermediate nodes, TOMA allows the packets to be replicated at proxies and hence decreases the redundant packets sent over the physical links. We also found out that TOMA achieves higher performance than NICE in several other metrics (e.g., link stress and path length), which is again due to the efficient overlay construction [15].

4.3 Control Overhead

In this subsection, we compare the total number of control messages generated by TOMA and NICE during a group's life time, i.e., 1000 seconds. For the sake of a fair comparison, we set the parameters in NICE and TOMA to the same values whenever possible. We found out that in NICE, the total number of control messages increases very rapidly with group size, while in TOMA, the increase is much more steady (Fig. 3). At group size of 1000, 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 split, merge, or change cluster leaders

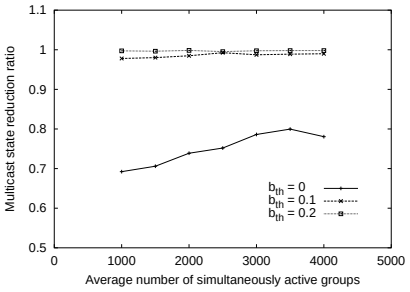

Fig. 4. State reduction.

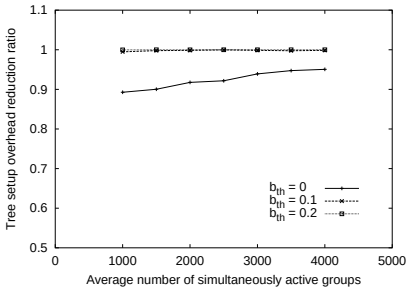

Fig. 5. Tree setup overhead reduction.

very frequently to enforce the bounds on cluster size, inducing 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) to maintain the clusters, whereas a TOMA node only needs to refresh with its parent to maintain the connection.

4.4 Effectiveness of OLAMP

Recall that OLAMP not only reduces multicast forwarding entries, but also suppresses a lot of control messages transmitted for tree setup and maintenance. Therefore, we first measure *reducible multicast forwarding entries*, that is, the multicast state entries in non-member proxies, since member proxies always need to maintain group state for encapsulation and decapsulation purpose. We then measure *multicast tree setup overhead*, defined as the total number of tree setup messages relayed by proxies when multicast trees are established inside MSON.

We perform simulations for multiple groups in AT&T topology. We assume multicast groups arrive as a Poisson process, and their lifetime has an exponential distribution. We vary the bandwidth overhead threshold b_{th} in the group-tree matching algorithm presented in Sect. 3.2 from 0 to 0.2 in the simulation.

We define reduction ratio of multicast state entries as the percentage of reducible multicast state entries reduced by OLAMP. Then we plot this metric vs. number of active groups in Fig. 4. Clearly, more than 70% of the reducible multicast state entries can be reduced even when b_{th} is 0. We also observe that as more multicast groups become active, the reduction ratio improves. Furthermore, when b_{th} is raised from 0 to 0.2, we are able to eliminate more than 99% of the reducible state, albeit at the expense of higher bandwidth overhead. This figure demonstrates that TOMA is scalable to the number of co-existing groups.

We plot the reduction ratio of multicast tree setup overhead in Fig. 5. This figure exhibits a similar trend as Fig. 4: more reduction in tree setup overhead is accomplished when there are more concurrent groups and more bandwidth is wasted to force higher aggregation. We obtained similar results for tree maintenance overhead [15]. These results indicate that the MSON ISP can control the trade-off of bandwidth cost versus control overhead by setting b_{th} properly.

4.5 Summary

Our simulation results can be summarized as follows: TOMA creates multicast trees with performance almost comparable to IP multicast; the control overhead of TOMA is significantly less than NICE for large groups; 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 NICE, it is not difficult to draw the above conclusions. First of all, the strategic MSON dimensioning tends to take physical network topologies into consideration, and thus provides efficient multicast data delivery. In addition, the control overhead of TOMA is greatly reduced, since the communication is limited inside each local cluster and inside the overlay network. Finally, aggregated multicast approach further reduces the tree maintenance overhead and makes TOMA scalable to large numbers of groups. In contrast, in application layer multicast, the control overhead of exchanging information between peers and establishing multicast trees will increase rapidly with group size. Due to limited bandwidth at end systems, as group size increases, the depth of the multicast trees has to be increased, leading to long latency for data delivery. Nevertheless, we want to point out that TOMA requires infrastructure support to achieve the above benefits, which is a trade-off of TOMA vs. NICE.

5 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) By extensive simulation studies, we show that the performance of TOMA is very promising. We believe that the invention of our practical, comprehensive, and scalable multicast service model would significantly facilitate the multicast wide deployment.

Future Work We plan to continue our work in two directions. First, more research efforts are needed to investigate heuristic and distributed algorithms for overlay dimensioning. Second, we would like to conduct more comprehensive performance evaluation of TOMA, by comparing with other application-layer multicast protocols, and applying other multicast routing algorithms (eg., those reported in [18] and [4]) to OLAMP.

References

1. K. Almeroth. The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment. *IEEE Network*, Jan./Feb. 2000.

2. A. Ballardie. Core Based Trees (CBT version 2) multicast routing: protocol specification. *IETF RFC 2189*, Sept. 1997.
3. S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable application layer multicast. In *Proceedings of ACM SIGCOMM*, Aug. 2002.
4. S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of IEEE INFOCOM*, Apr. 2003.
5. K. Calvert, E. Zegura, and S. Bhattacharjee. How to model and internetwork. In *Proceedings of IEEE INFOCOM*, Mar. 1996.
6. M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489 – 1499, Oct. 2002.
7. 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/.
8. Y. Chawathe, S. McCanne, and E. A. Brewer. RMX: Reliable multicast for heterogeneous networks. In *Proceedings of IEEE INFOCOM*, Mar. 2000.
9. Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, June 2000.
10. C. Diot, B. Levine, J. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, Jan. 2000.
11. A. Fei, J.-H. Cui, M. Gerla, and M. Faloutsos. Aggregated Multicast: an approach to reduce multicast state. *Proceedings of Sixth Global Internet Symposium(GI2001)*, Nov. 2001.
12. P. Francis. *Yoid: Extending the Multicast Internet Architecture*. White papar, http://www.aciri.org/yoid/.
13. J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O. Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*, Oct. 2000.
14. M. Kwon and S. Fahmy. Topology aware overlay networks for group communication. In *Proceedings of NOSSDAV'02*, May 2002.
15. L. Lao, J.-H. Cui, and M. Gerla. A scalable overlay multicast architecture for large-scale applications. Technical report, UCLA CSD TR040008. http://www.cs.ucla.edu/NRL/hpi/papers.html, Feb. 2004.
16. D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.
17. S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proceedings of NGC*, Nov. 2001.
18. S. Shi and J. S. Turner. Routing in overlay multicast networks. In *Proceedings of IEEE INFOCOM*, June 2002.
19. A. Sobeih, W. Yurcik, and J. C. Hou. Vring: a case for building application-layer multicast rings (rather than trees). In *Proceedings of the IEEE Computer Society's 12th Annual International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, Oct. 2004.
20. 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. In *Proceedings of NOSSDAV'01*, June 2001.