A Scalable Hybrid Overlay Multicast Architecture for Large-Scale Applications

Li Lao[1], Jun-Hong Cui[2], Mario Gerla[1]
llao@cs.ucla.edu, jcui@cse.uconn.edu, gerla@cs.ucla.edu
[1] Computer Science Department, University of California, Los Angeles, CA 90095
[2] Computer Science & Engineering Department, University of Connecticut, Storrs, CT 06029

Technical Report TR040008
02-09-2004
Computer Science Department
UCLA

Abstract— We propose a two-tier overlay multicast architecture to efficiently support large-scale multicast applications. In this architecture, overlay multicast scheme is adopted in the backbone domain, while application-layer multicast protocol is responsible for the communication between the limited number of end users in access domains. Our two-tier architecture is able to provide efficient resource utilization with less control overhead, especially for large-scale real-time applications. It also alleviates the forwarding state scalability problem of overlay multicast and simplifies multicast tree construction and maintenance when there are large numbers of groups ongoing in the networks. Based on this architecture, we also suggest a cost-based pricing model for the overlay ISPs to charge multicast groups. We believe this pricing model provides incentives for both service providers and clients to adopt our multicast service scheme. Simulation studies indicate that this architecture performs well in several common scenarios, and it is scalable to group size as well as to the number of co-existing groups.

I. Introduction

In this paper, our goal is to design a scalable, efficient, and practical multicast architecture. Multicast is defined as the communication mechanism among more than one machines. Starting from the inception of the concept, there are mainly three architectural tracks which targets efficient and practical multicast service support: IP multicast, application-layer multicast [1], and overlay multicast.

IP multicast treats multicast delivery as a primitive operation at the network level. It utilizes a tree delivery structure, on which data are only replicated at branching routers and are forwarded only once over each link. And routers are responsible for maintaining group routing tables. This approach makes IP multicast fast, resource efficient and scale well to support very large multicast groups. However, even after approximately 20 years of multicast research and engineering efforts (since Stephen Deering established the IP multicast model [18] in 1988), IP multicast is still far from being widely deployed on the Internet. As a matter of fact, the Mbone (a multicast testbed in the Internet) is the only global multicast infrastructure available. This is due to many technical reasons as well as marketing reasons. The most critical ones 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 and transport services, the requirement of global deployment of multicast-capable IP routers and the lack of appropriate pricing models, as make Internet Service Providers (ISPs) reluctant to deploy and provide multicast service. To alleviate some of the difficulties, solutions in alternative architectural tracks have been proposed to provide multicast service: application-layer multicast and overlay multicast.

Application-layer multicast [12] [6] [28] [26] [22] [30] [37] [9] implements multicast-related features exclusively at end systems, and does not require infrastructure support from intermediate nodes (such as routers or proxies). Data packets are transmitted between end hosts via unicast, and are only replicated at end hosts. Though highly flexible, it leads to relatively low bandwidth efficiency, since a data packet may traverse a physical link multiple times. For large multicast groups, due to limited bandwidth at end systems, the depth of the multicast tree has to be increased, as leads to long latency for data delivery. In addition, the control overhead of exchanging information between peers and establishing multicast trees will increase as the group size increases. Thus, application-layer multicast is only suitable for applications with small multicast groups, such as video conference, multi-player games and distance education, or for groups with low bandwidth data, such as news and sports ticker services, and stock quotes and updates. Furthermore, application layer multicast has the same billing problem as IP multicast: since group members are distributed all over the Internet and they join or leave the group at will, neither ISP nor group coordinator is able to measure the bandwidth usage associated with a group. Without appropriate billing methods, ISPs or group coordinators will not be motivated to provide such services.

In overlay multicast networks [25] [10] [1] [32] [31] [7], a set of special overlay service nodes deployed by the overlay ISPs self-organize into an overlay network and deliver data on multicast distribution trees built on top of the overlay network. End hosts subscribe to appropriate overlay nodes and receive data via unicast or local IP multicast. The overlay ISPs are able to manage the overlay topology to optimize the network performance and enable efficient resource usage that is comparable to IP multicast. The communication overhead to maintain con-

[1] We use application-layer multicast to refer those multicast protocols which only involve end hosts, without the support from intermediate proxies

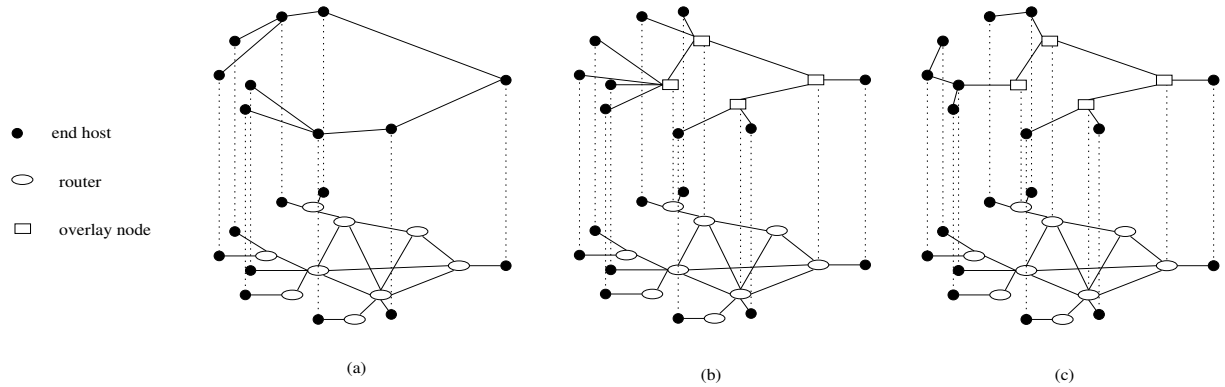

Fig. 1. A comparison of (a) application layer multicast, (b) overlay multicast and (c) hybrid overlay multicast (SHOMA).

trol and data delivery path is also reduced, since the communication is now limited inside the overlay network and inside each overlay node's local scope. So for large-scale applications or applications with high bandwidth requirement, for example, content distribution applications including Internet TV, web caching, file distribution, and video streaming, overlay multicast approach has more advantages compared with application-layer multicast. By observing these differences between application layer multicast and overlay multicast schemes, we believe that without explicit support from the infrastructure, it is not possible to build practically deployable multi-point distribution systems that can scale well beyond a few hundred to a few thousand client.

Overall, it seems that overlay multicast is the recent trend for scalable multicast service support. However, unlike application-layer multicast, which completely eliminates multicast routing functionalities from intermediate nodes (routers or proxies), overlay multicast does involve the overlay service nodes (i.e., proxies) to assist multicast routing. That is, proxy nodes need to establish and maintain multicast tree routing information for each multicast group. When there are a large number of multicast groups, the limitations of proxy processing power and network interface bandwidth will be challenged. In other words, overlay multicast inherits the state scalability problem from IP multicast. Further more, in overlay service networks, it is even more important to provide appropriate pricing models in order to stimulate ISPs' interests in multicast service deployment.

Motivated by the above critical issues, in this paper, we propose a two-tier multicast architecture, called Scalable Hybrid Overlay Multicast Architecture (SHOMA), to provide scalable, efficient, and practical multicast support for large-scale applications. In this architecture, we advocate the notion of multicast service overlay network (MSON) as the backbone service domain. MSON consists service nodes or proxies which are strategically deployed by the MSON provider. In MSON, the traffic intensity is very high and efficient network resource management becomes a very critical issue. To solve the state scalability issue, we adopt aggregated multicast approach [21], with multicast data transmitted on aggregated trees. For end hosts (group members), they subscribe to MSON, by transparently connecting to some special proxies (called member proxy) 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. A high level comparison of application-layer multicast, overlay multicast, and SHOMA is illustrated in Fig. 1.

Our two-tier architecture is able to provide efficient resource utilization with less control overhead, especially for large-scale real-time applications. It also alleviates the forwarding state scalability problem of overlay multicast and simplifies multicast tree construction and maintenance when there are large numbers of groups ongoing in the networks: overlay proxies in the backbone service domain employ "aggregated multicast" and maintain multicast state for aggregated delivery trees instead of individual groups. Therefore, our architecture scales to multicast group size as well as to number of coexisting multicast groups. To our best knowledge, this paper is the first one to address the state scalability problem in overlay multicast.

Based on this architecture, we also suggest a cost-based pricing model for the overlay ISPs to charge multicast groups. We believe this pricing model provides incentives for both service providers and clients to adopt our multicast service scheme.

The rest of this paper is organized as follows. Section II illustrates our strong motivations for advocating MSON, and gives a brief overview of aggregated multicast approach. In Section III, we present our hybrid overlay multicast architecture in detail. In Section IV, we propose a pricing model to solve billing issues of multicast service. Then Section V evaluates the performance of hybrid overlay multicast architecture through simulations. Finally, we offer an overall summary of our contributions in Section VII.

II. Background

A. Multicast State Scalability and Aggregated Multicast

Multicast state scalability is among the technical difficulties that delay the deployment of IP multicast. Conventional IP multicast protocols establish and maintain a multicast tree per group or per group/source, and require each router to maintain separate states for individual groups or group/sources. Hence, large number of co-existing groups mean large amount of state to be maintained at routers and high multicast tree setup and maintenance control overhead. This state scalability problem is even exacerbated in backbone networks, because there are potentially enormous multicast groups crossing backbone domain.

The state scalability has prompted some research in forwarding state reduction. Most schemes attempt to reduce forwarding state by "intra-group" or "inter-group" state aggregation. [35] [33] [14] propose intra-group aggregation approaches, which try to reduce forwarding state at non-branched routers, but they mainly target networks with a larger number of sparse groups. Some inter-group aggregation schemes try to achieve state reduction by aggregating forwarding state at routers [29] [34]. Thaler and Handley analyze the aggregatability of forwarding state in [34] using an input/output filter model. Radoslavov et. al. propose algorithms to aggregate forwarding state and study the bandwidth-memory tradeoff with simulations in [29]. However, the state aggregatability of this type of schemes heavily depends on multicast address allocation. It should be noted that all the schemes mentioned above are dedicated to solve the state explosion issue without examining the control overhead explosion one.

To solve the state scalability problem in IP multicast, a scheme called aggregated multicast has been proposed in [21]. The key idea is to force multicast groups to share a single distribution tree. This enforcement takes place at the edge routers of the network. Data packets from different groups are multiplexed on the same distribution tree, call *aggregated tree*. Each data packet of these groups is encapsulated and travels on the aggregated tree. This way, routers in the middle of the network, namely core routers, need to keep state only per aggregated tree, which are much less in number than the groups they are servicing. Of course, edge routers of the network need to maintain sufficient information to multiplex and demultiplex groups in and from aggregated trees. Aggregated multicast can reduce the required multicast state at core routers. It can also reduce the management overhead for the distribution trees, since there are fewer trees to be established and maintained. Fig. 2 illustrates the basic idea of aggregated multicast.

In aggregated multicast, we need to match groups to aggregated trees. The group-tree matching problem hides several subtleties. The set of the group members and the tree leaves are not always identical. A match is *perfect* for a group, if all the tree leaves have group members. A match could also be a *leaky match*, if there are leaves of the tree that do not have group members. In other words, we send data to part of the tree that has no receivers. A disadvantage of the leaky match is that some bandwidth is wasted to deliver data to nodes that are not member of the group. Namely, we trade off bandwidth for state scalability.

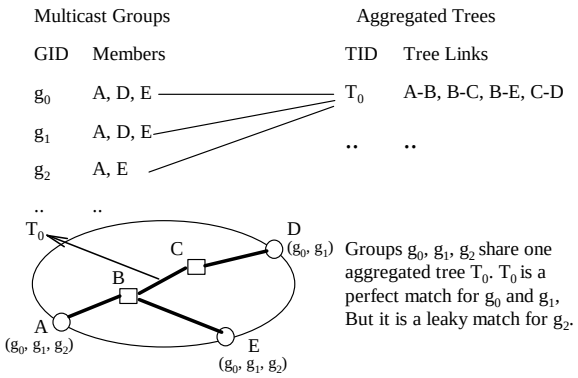

Fig. 2. An illustration of aggregated multicast

As we mentioned earlier, the state scalability problem also exists in overlay multicast, since each overlay proxy node acts similarly to a router when establishing multicast trees and forwarding multicast packets. In fact, due to the limitations of the proxy processing power and network interface bandwidth, the state scalability problem is even worse. This is because the performance of proxies will be degraded significantly when the routing tables are huge, and the bandwidth becomes even more scarce when there are a large amount of tree maintenance messages. For the MSON providers, however, they want to gain more revenue forever, which means that they more customers forever. Thus, we employ aggregated multicast scheme in MSON, by which the number of multicast distribution trees can be significantly reduced. As a result, the required memory and processing power at each overlay proxy node are also reduced, which makes packet forwarding faster. In addition, there is less communication overhead associated with tree setup and maintenance. Therefore, aggregated multicast becomes a powerful tool to improve the scalability and efficiency of MSON. Finally, aggregated multicast can also facilitate fast failure recovery with less control overhead, and assist to support end user access control and establishing practical pricing model as will be addressed in the following sections.

III. Scalable Hybrid Overlay Multicast Architecture

We design Scalable Hybrid Overlay Multicast Architecture (SHOMA) to provide scalable and efficient multicast services to end users. In this section, we describe our proposed architecture in more details.

A. An Overview

SHOMA is a two-tier multicast architecture. In this architecture, multicast service overlay network (MSON) is advocated as the service backbone domain. In MSON, overlay

proxies are strategically placed to form an overlay network, on top of which aggregated multicast distribution trees are built for data delivery. By using aggregated multicast, multiple groups are forced to share a single aggregated tree. Each aggregated tree is assigned a tree address, which is unique in MSON, while transparent to the end users. 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 some special proxies (called member proxy) advertised by the MSON provider. Each member proxy organizes some end 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. Therefore, SHOMA is built above network and transport layers.

Each (SHOMA) group is identified by a URL-like unique name. Group names have the form of *shoma://groupname.xxxmson.com/* [2]. End users (sources and receivers) explicitly subscribe to the group, by sending out a join request containing the URL-like group name. Through DNS, this request will reach the DNS server of the MSON, in which there resides a **group registry server**. It is the responsibility of the group initiator to register the group in the group registry server. In other words, the group registry server maintains the member information of each group, thus it is easy for the MSON to implement its member access control policy. After receive SHOMA join request, the DNS server will send a list of IP addresses of the advertised member proxies back to the subscriber. How to choose an appropriate member proxy will be described later in Section III-B.

Inside MSON, there are three types of proxy nodes:

1. Member proxies: when an end user joins a group, its join request is re-directed by the MSON DNS server to an approximate proxy at the edge of overlay networks, which will then subscribe to multicast groups on behalf of end host members. These proxies are referred as "member proxies" since they are members of the multicast groups inside the overlay service backbone domain (MSON). In this way, multiple end hosts of the same group might be attached to one member proxy, and we call this set of end users and their member proxy as a "cluster". Member proxies participate in multicast data forwarding of both backbone domain and user access domains (outside MSON). After receiving join request for a group g, the member proxies will set up a peer-to-peer multicast tree in the local cluster and relay join request to a predetermined proxy (i.e., "host proxy" for group g) to establish the overlay multicast tree in backbone domain.

2. Host proxies: in backbone domain, each group is managed by a host proxy. After receiving a join request for g relayed by a member proxy, this host proxy conducts multicast routing and group-tree matching mechanisms to map group g to an aggregated tree. To improve the state scalability in a further step, SHOMA adopts bi-directional aggregated trees in backbone domain. An aggregated tree is identified by a pair of the host proxy IP address and a tree ID allocated by the host proxy. Note that a host proxy is only present in the control plane and may not be involved in data delivery. It can even be dynamically changed during the lifetime of a session, when it no longer has appropriate trees for that group due to group membership dynamics, or when it fails. The host proxy for a group g can be randomly selected or by applying a hash function (called *group-to-host* hashing function).

3. Forwarding proxies: those proxies which are neither member proxies, nor host proxies, are referred as "forwarding proxies". They are mainly responsible for forwarding multicast data inside the backbone domain.

These three types of proxies can have different processing power and bandwidth capacity commensurate with their functionalities. For example, host proxies should be very powerful in processing capability in order to perform computationally-intensive tasks such as group-tree matching; forwarding proxies require higher bandwidth to accomplish successful data delivery; and member proxies require both. Note that, it is possible that some member proxies are "forwarding" proxies for some groups as well.

In a nutshell, in SHOMA, member proxies control member access policy and manage peer-to-peer multicast trees in its cluster, and host proxies conduct group-tree matching and manage aggregated multicast trees in the backbone domain. A big picture of SHOMA is illustrated in Fig. 3

We have discussed the main components of SHOMA (forwarding, member and host proxies) and their functionalities. In the following, we describe this architecture in more details. To facilitate our description, we divide the control messages into two types: P-type and O-type. The messages that facilitates end users to join or leave groups and peer-to-peer tree construction are P-type, such as *P-JOIN*, *P-LEAVE* and *P-TREE*. The messages used to help matching groups to trees inside MSON are O-type, which includes *O-JOIN*, *O-JOIN-*

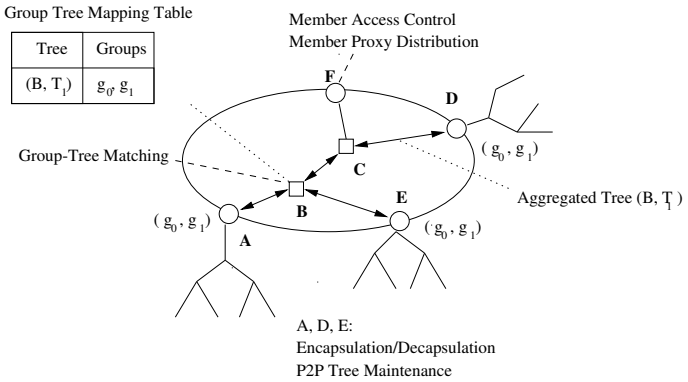

Fig. 3. A big picture of SHOMA, where F is group registry server/DNS server, B is host proxy, A, D and E are member proxies, groups g_0 and g_1 shares the bi-directional aggregated tree (B, T_1).

[2]The URL-like naming approach has been adopted by many systems, such as CDN (content dilivery networks), Yoid [22], Scattercast [1], Overcast [25], etc.

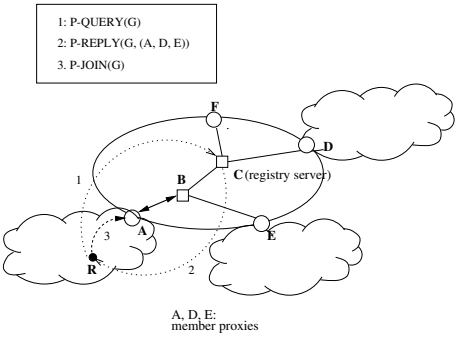

Fig. 4. Member R of group g queries host proxy B for member proxy list.

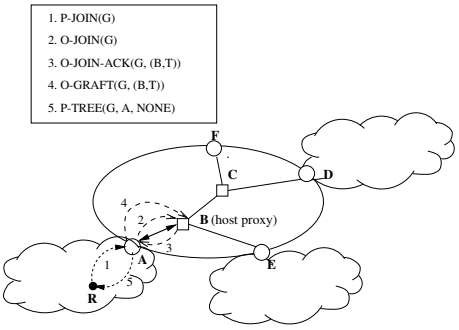

Fig. 5. Member R joins group g with host proxy B.

ACK O-LEAVE, *O-GRAFT* and *O-PRUNE*.

B. Member Proxy Selection

When an end host joins group g, it first queries the MSON DNS server using a URL-like group name to obtain a list of available member proxies, identifies an appropriate proxy m, and sends a *P-JOIN(g)* request to m (as shown in Fig. 4). Although the choice of member proxies should depend on application-specific requirements, we provide some general guidelines for selecting a good member proxy.

The first criteria is low distance or latency. It can be obtained from round-trip-time (RTT), which measures the communication delay between an end user and a member proxy. After obtaining member proxy list, the end user measures the RTT values to all eligible proxies and discretizes the RTT values into per-determined levels. The RTT discretization improves the stability of RTT values and allows other metrics to be considered when the proxies have similar RTTs.

The second criteria is low workload. Workload can be determined by the total number of end users a proxy currently handles or total amount of access bandwidth in use at the proxy. This metric can reflect the processing delay or available bandwidth at the proxy node. The workload is measured by the proxy itself and can be obtained by the end users in centralized or distributed fashion. In the centralized method, the MSON DNS server maintains the workload of all member proxies and distributes this information to a new member in the query response message. In the distributed approach, each member proxy piggybacks its workload when end user measures RTT. After collecting RTTs and workload for all member proxies, the end user selects the one with lowest RTT. If multiple proxies fall into the same RTT level, the workload is used to break the tie.

This member proxy selection method allows a client to choose a good proxy; however, it incurs a latency of at least an RTT between the client and the proxy. In order to reduce the latency of the join process, the new member can subscribe to a proxy randomly selected from the member proxy list and receives data temporarily from this proxy. If the performance with this temporary proxy node is not satisfactory, the user can probe eligible member proxies. To avoid an end user oscillating between proxies with similar distance, the end user is allowed to switch proxy only if the RTT to the new proxy can upgrade to a higher level. If this is the case, the end host unsubscribes from the old proxy and subscribes to the new proxy.

C. Member Join and Leave

Outside MSON, an end host joins group g by subscribing to an appropriate member proxy m. After receiving *P-JOIN(g)* request, member proxy m checks if it already belongs to group g. If yes, it has the group-tree matching information and suppresses this request to save communication overhead. On the other hand, if it has not joined the group, it will use the group-to-host hashing function to get the host proxy h, and then send *O-JOIN(g)* request h. After conducting the group-to-tree matching, the host proxy h establishes or finds an appropriate aggregated tree, say, T. It will send back a *O-JOIN-ACK(g, T)* message to m. If m has not joined the delivery tree T, it will graft to the tree by sending *O-GRAFT(T)* message towards h.

Now that group g has been mapped to T inside backbone domain, m constructs or updates the peer-to-peer multicast tree t in its local cluster and replies the new member a *P-TREE(g, parent, children)* message, where*parent* and *children* represents the information for establishing P2P multicast tree t (We will discuss P2P multicast tree construction protocol in Section III-F). From now on, this member will start to receive packets for g relayed by m. An example of member join is illustrated in Fig. 5.

When an end host leaves group g, it sends a *P-LEAVE(g)* message to its member proxy m. The proxy may adjust the local multicast delivery tree and distribute the tree change to the rest of the cluster with *P-TREE(g, parent, children)* messages. If no more receivers are attached to m, m propagates a *O-LEAVE(g)* message to the host proxy h, 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 matching between g and T. This group-tree mapping may trigger removal of the tree T when there are no other groups mapped onto T. In this case, h sends *O-LEAVE-ACK(T)* messages to the tree leaves of T, which will in turn prune from the tree by sending *O-PRUNE(T)*. This procedure is shown in Fig. 6. In case the end host leaves the group ungracefully, its member proxy will be able to detect this from periodic "heart-

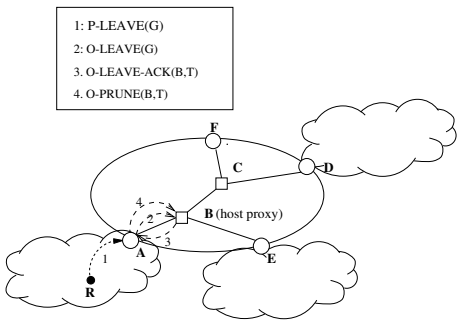

Fig. 6. Member R leaves group g with host proxy B.

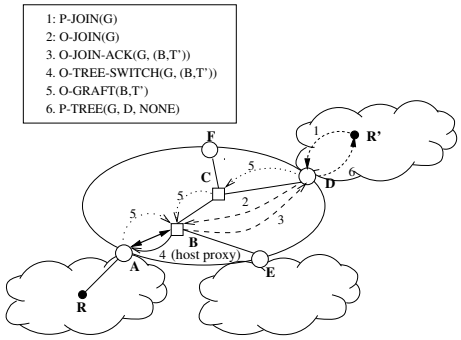

Fig. 7. Group g starts with member R. A new member R' joins group and group switches from (B, T) to (B, T').

beat" message exchange.

In the backbone domain, when the member proxies of a group g are changed due to end host join-or-leave dynamics, its original aggregated tree T may not be able to cover the group again. In this case, the tree switch procedure is triggered, as shown in Fig. 7. The host proxy first finds or establishes an appropriate tree T' for g and removes g from T. It notifies the member proxies to join the new tree T' and leave T by using message *O-TREE-SWITCH(g, T')*. Note that this tree switch process may trigger the establishment of a new tree T' and/or the removal of the old tree T.

The details of multicast routing, group-tree matching and peer-to-peer tree construction algorithms will be explained in the following sections.

D. Multicast Routing in MSON

When a host proxy cannot find an appropriate group-tree mapping for a multicast group, it needs to run multicast routing algorithm to establish a new multicast delivery tree. In SHOMA, the group-tree matching algorithm is compatible with existing multicast routing protocols such as CBT [5], PIM [17], and SSM [24]. For single-source applications, source based trees result in higher performance (i.e., less latency) than core based tree. Conversely, for applications with multiple sources, core based tree reduce multicast tree setup and maintenance overhead. Considering that bi-directional trees can achieve better tree aggregation regardless of how many sources there are, we design our multicast trees to be bi-directional. Thus, when a host proxy needs to construct a new multicast tree (if current aggregated trees are inappropriate for a group), it uses CBT to compute a bi-directional multicast tree with the host proxy itself as the core.

E. Group-Tree Matching in MSON

To map a multicast group to an aggregated tree, a host proxy needs to conduct group-tree matching algorithm. In this section, we present a simple algorithm while with a small amount of additional control overhead.

First, we define bandwidth waste function. As mentioned earlier in Section II, leaky match can aggregate more groups into a multicast tree, but it introduces extra bandwidth waste. Hence, it is necessary to control the amount of bandwidth waste for leaky match. 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). With the assumption that all multicast groups have same bandwidth request, we define the **average bandwidth waste** for T:

$$\begin{aligned}\delta(T) &= \frac{n \times C(T) - \sum_{i=1}^{n} C(T_0(g_i))}{\sum_{i=1}^{n} C(T_0(g_i))} \\ &= \frac{n \times C(T)}{\sum_{i=1}^{n} C(T_0(g_i))} - 1,\end{aligned} \tag{1}$$

where $C(T)$ is the cost of tree T, i.e., the total cost of tree T's links. It should be noted that a host proxy needs to know the tree topology information to compute average bandwidth waste. To overcome this complexity, we propose an **estimated bandwidth waste** function that takes into account only the group member and tree leaf information:

$$\begin{aligned}\delta_{est}(T) &= \frac{n \times |leaf(T)|}{\sum_{i=1}^{n} |leaf(T_0(g_i))|} - 1 \\ &= \frac{n \times |leaf(T)|}{\sum_{i=1}^{n} |g_i|} - 1,\end{aligned} \tag{2}$$

A group-tree matching algorithm similar to the one proposed in Bi-dirEctional Aggregated Multicast Protocol (BEAM) [16] can be used to map a group to an aggregated tree, and set up or remove trees accordingly. The basic idea is as follows: the host proxy attempts to map the group to existing aggregated trees if the tree can cover all group members and the estimated bandwidth waste δ_{est} is less than a pre-defined bandwidth waste threshold b_{th}. If this fails, it will use multicast routing protocol to compute the native multicast tree for this group as the aggregated tree (as illustrated in Section III-D and set up this new tree. If this group was originally mapped to a different tree, the old mapping is removed and the new mapping is installed. This group-tree matching algorithm enables group-tree mapping to adapt to membership dynamics and link metric changes. In addition, it allows the host proxy of a group to be changed when the current proxy does not have

an appropriate tree while another host proxy does. This approach increases group aggregatibility at the cost of delay for tree construction and message communication overhead.

Furthermore, real-time multicast applications such as video streaming and video-conferencing call for multicast services with Quality of Service (QoS) support. To provide QoS support in our architecture, the host proxy should collect link state information and group membership information, enforce admission control for new multicast group request, and perform QoS-aware group-tree matching algorithm. QoS support also requires traffic conditioning at member proxies to control source rate. Details on aggregated QoS multicast provisioning can be found in [15].

F. P2P Multicast outside MSON

We use application layer multicast outside MSON, considering its self-organization capability and higher efficiency than unicast. By subscribing appropriate member proxies, end users form into different clusters. In each cluster, nodes communicate in a P2P fashion. 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.

In our architecture, every cluster has one special node, that is, the member proxy. This proxy is relatively more powerful and stable than end users and it maintains membership information. This characteristics fits a centralized method naturally. Therefore, we use an approach similar to ALMI [28] in P2P multicast tree construction. For each member, the member proxy randomly selects a subset of users to be monitored and sends this user list to the member. Each member sends probing packets to the subset of users and its parent periodically, and reports the information (such as delays) to the member proxy. After the proxy collects all information, it calculates appropriate P2P multicast delivery trees and distributes them to the end users in the form of $(parent, children)$. Finally, end users will connect with their children and deliver data packets to them through unicast connections. During the tree transition period, the multicast packets are sent on old routes as well as new routes for a short time to avoid data loss.

The constructed multicast distribution trees can be either bi-directional or unidirectional. Bi-directional trees accommodate both single- and multi-source group communications, whereas unidirectional trees provide services with higher quality. Depending on which performance metric is the most important, different multicast trees (for example, Minimum Spanning Tree, Compact Tree [32], Degree-Constrained Shortest Path Tree [12], or Shortest Widest Path Tree [11] [36]) can be established.

Since the member proxy nodes periodically computing new multicast trees, the clusters are able to maintain high-quality multicast tree in presence of group dynamics and transient failure of links or nodes. If a member leaves ungracefully or if a path between two neighbors becomes broken, these events can be detected from periodic probing and the multicast tree will be repaired by member proxies. Due to the limited size of each cluster, the control message overhead is expected to be very small.

G. Discussions

Member and Source Access Control One critical problem in traditional IP multicast is the lack of effective access control mechanisms. In SHOMA, access control can be managed by the group registry server. Since every end user needs to obtain the list of available member proxies, it has to contact the MSON DNS server and the group registry server in the first place. Consequently, the group registry server will rule out the ineligible end hosts.

Besides member access control, the sources of a group may also need to be explicitly controlled for security reasons. In this case, bi-directional trees may not be able to prevent unauthorized members from injecting packets into the network. There are two solutions to this problem. If there are limited sources in a local cluster (such as Internet TV or video streaming), one source-based tree is constructed for each source, and packets sent from unauthorized source will be dropped. When there are a large number of sources in a local cluster, multiple source-based trees will result in enormous overhead. Then a source-based tree rooted at the proxy can be constructed. The sources will send data packets to the proxy directly, which filters out unauthorized packets and relays the remaining packets to other nodes in the local cluster. There is a tradeoff of higher tree maintenance overhead versus longer latency between these two methods, and the choice should be made depending on the average number of sources in a cluster.

Fault Recovery of Multicast Trees Since the MSON provider buys bandwidth from higher tier ISPs and the overlay proxy nodes are specially designed servers, it is expected that proxy and link failures do not occur very often and it is quite unlikely for network partition to take place in the backbone domain. However, our architecture still takes account of these situations, and uses a pre-planned restoration approach for fast recovery.

The employment of aggregated multicast greatly facilitates the fault tolerance of SHOMA. Aggregated multicast reduces the number of trees to be maintained by the overlay network, so the backup tree computation and maintenance cost is significantly reduced. In addition, since the recovery is done for aggregated multicast trees rather than individual groups, there is less communication overhead when a failure occurs. When a new aggregated tree is established, the host proxy computes a backup tree by using some redundant tree fault-tolerance schemes, such as the algorithm described in [27] and [20]. When a proxy or link failure occurs, it can be detected by the lack of "heartbeat" message exchange caused by the failed proxy or failed link and propagates this information to all other proxy nodes. The host proxies determine the affected aggregated trees, retrieve their backup trees, and switches the

related multicast groups to the backup trees.

Resilience of Member and Host Proxies Member proxies and host proxies have special functionalities that needs to be separately handled when they fail. If a member proxy dies, the end hosts are able to detect the failure and will contact the MSON DNS server and the group registry server to re-join the group. In case of a host proxy failure, a new host proxy is chosen from the backup proxy list, and then member proxies will switch to new host proxies, which will conduct group-tree matching and assign new multicast trees.

Load Balancing When there is a large number of bandwidth-demanding multicast groups in the overlay network, the overlay links may be congested and the multicast trees traversing these links will yield poor performance for end users. To detect congested links, the overlay proxies need to monitor the current congestion conditions on its adjacent links and report overloaded link back to host proxies. For reserved or assured services, the admission control module in the host proxy will take account of the QoS requirements and available resources, and decides whether a multicast group can be admitted [15]. For best-effort traffics, the host proxies will try to bypass those congested links when executing the group-tree matching procedure.

The working load at member proxies can also be unbalanced if a member proxy is particularly popular. When the MSON provider provisions the network resource usage, it can hide the overloaded proxies from the multicast sessions by not providing these proxies as available member proxies to end users. In addition, when an end user tries to find a member proxy node before it joins a group, it uses working load as one criteria. In this way, the end user tends to find a lightly-loaded proxy node.

Heterogeneity Handling For high-bandwidth applications such as video-streaming, the inherent heterogeneity of current Internet has made multicast a challenging problem, since there is no single rate that can fit the demand of receivers with different bandwidth and processing capabilities. A promising solution to this problem is to divide the group members into a number of homogeneous sub-groups, which has been adopted in existing overlay multicast architectures [1] [10]. Even though this approach solves user heterogeneity problem, it exacerbates the state scalability problem, because multiple multicast trees are now needed for each multicast group. SHOMA, on the other hand, can be seamlessly integrated with this approach by significantly decreasing the number of aggregated trees to manage.

IV. Pricing Model

Ideally, a MSON architecture should provide MSON ISPs with practical means of accounting the cost of a multicast group and billing the group initiator or group members accordingly. Its pricing scheme should bring revenue to ISPs to motivate them to deploy the services, and promote customers to purchase the services by saving money for them. In this section, we suggest such a pricing model for ISPs who deploy SHOMA architecture.

ISPs are not willing to deploy new services unless they can maximize profit and minimize cost. Multicast, irrespective of in which layer it is implemented, achieves bandwidth savings over unicast by minimizing the transmission of duplicate packets over every link. Hence, it seems that ISPs should prefer existing multicast architectures over unicast for higher profit. However, this is not the case. Clearly, it is not feasible for an ISP to account the resource usage of multicast communication in IP multicast or application-layer multicast, in that the group membership information is distributed in routers (which may belong to different ISPs) or non-cooperating end hosts. In contrast, in SHOMA, host proxies are responsible for maintaining group information inside MSON and member proxies control member hosts in local clusters. With this information at hand, ISPs are able to estimate resource usage such as bandwidth consumption in MSON. Note that we do not consider the bandwidth usage outside MSON, since we expect the end users to bear the responsibility and have the freedom to select appropriate Internet connections to maximize application performance within their budget limit.

Generally, the major cost for ISPs to deploy SHOMA services can be categorized into bandwidth cost and equipment cost. Bandwidth cost is the amount of bandwidth leased from tier-one ISPs, and it depends on factors such as number of groups, group size, membership distribution, and group duration. Equipment cost includes the deployment and maintenance cost of storage, memory, CPU, etc. Therefore, in our scheme, the ISPs charge every group a usage-based price for bandwidth cost and a flat-rate price for equipment cost.

Bandwidth Price Our proposed bandwidth price is based on Chuang-Sirbu Law, which states that for IP multicast the cost of a multicast tree varies at the 0.8 power of the multicast group size [13]:

$$\frac{L_m}{\overline{L_u}} = N_m^k \tag{3}$$

where L_m is the total number of links in the multicast distribution tree, $\overline{L_u}$ the average number of links in a unicast path, N_m the multicast group size, and k a scaling factor. Recent study shows that a similar power law relationship with a power value of approximately 0.9 exists in application-layer multicast trees [19]. Based on these studies, we propose that when a multicast group is leaky matched to an aggregated SHOMA tree with a bandwidth waste threshold of b_{th}, the relative bandwidth cost of this multicast tree is bounded by:

$$\frac{L_m}{\overline{L_u}} = b_{th} \times N_p^{k'} \tag{4}$$

where N_p is the number of participating member proxies of the multicast group, and k' is a scaling factor empirically determined by ISPs. In other words, the price for the multicast group P_m is calculated based on the price of unicast service for the same group P_u:

$$\begin{aligned} P_m &= b_{th} \times N_p^{k'} \times \frac{P_u}{N_p} \\ &= b_{th} \times N_p^{k'-1} \times P_u \end{aligned} \tag{5}$$

In fact, ISPs can set a price in the range of $[P_m, P_u]$:

$$P_1 = P_m + \alpha \times (P_u - P_m) \qquad (\alpha \in [0,1]) \qquad (6)$$

By setting appropriate α, overlay ISPs has the flexibility to take account into marketing factors such as variations of demand and supply, promotion discounts, etc. In addition, this price is able to compensate for the cost of overlay ISPs to purchase bandwidth from tier-one ISPs, and meanwhile it is still lower than what the customers need to pay for unicast services. Consequently, both overlay ISPs and customers are willing to deploy and use this kind of services.

Equipment Price Because the equipment price is not affected significantly by group size, a flat-rate price should suffice. For each multicast group, its equipment price is:

$$P_2 = \beta \times \frac{C_{eq}}{N_g} \qquad (\beta \geq 1) \qquad (7)$$

where C_{eq} is the total amount of equipment cost, and N_g is the total number of multicast groups. Similar to α in P_1, β is used to control the net profit for P_2.

Finally, the total price of a multicast group is the sum of bandwidth cost P_1 and equipment cost P_2:

$$P_{total} = P_1 + P_2 \qquad (8)$$

After the price is charged by overlay ISP, it is up to the group coordinator or initiator to decide how to share the cost among the group members. It can use existing approaches, such as Equal Tree Split (ETS), Equal Link Split among Downstream members (ELSD), or Equal Next-Hop Split (ENHS), to decide the receivers' share of the charge [23]. Depending on which approach the group coordinator prefers, the host proxy of this group may need to collect relevant information from member proxies and deliver the summarized data to the group coordinator.

V. Simulation Studies

In this section, we evaluate the performance of our architecture by using NS-2 simulations [3]. We compare SHOMA with a scalable application layer multicast protocol NICE [6], an IP multicast protocol (Core-Based Tree [5]), and non-scalable unicast protocol in a wide variety of network topologies. We find that SHOMA can achieve very competitive performance for large groups with hundreds (or even thousands) of members.

A. Network Topologies and Group Membership Model

To comprehensively evaluate our architecture, we use two types of network topologies. The first type of network topologies is generated using the Transit-Stub Model developed by Institute of Georgia Technology [8]. These topologies have 50 transit domain routers and 500-2000 stub domain routers. Each end host is attached to a stub router uniformly at random. To test the scalability of different schemes, we focus on large group sizes and vary the number of members in each group from 200 to 1000.

The second type of network topology is abstracted from a real network topology, AT&T IP backbone [4], which has a total of 123 nodes: 9 gateway routers, 9 backbone routers, 9 remote GSR (Gigabit Switch Routers) access routers, and 96 remote access routers. The abstract topology is constructed as follows: the attached remote access routers of a gateway or backbone router is "contracted" into one **contracted node**. In addition, we create a neighbor node called **exchange node** for each gateway router in the backbone, since gateway routers represents connectivity to other peering networks and/or Internet public exchange points. This abstraction procedure results in a simplified network of 54 nodes. Each end host are randomly assigned to a contracted node or exchange node according to *Random Node Weight Model*. In this model, each router is assigned a weight, which represents the probability that this router has attached end host(s) participating in a multicast group. Thus, for a group with fixed group size, the number of group members attached to a router is proportional to this router's weight. For the different routers in this abstracted network, gateway nodes and backbone nodes are assumed to be core routers only and are assigned weight 0. Each access router is assigned a weight of 0.01, and a contracted node's weight is the summation of the weights of all access routers from which it is contracted. Exchange nodes are assigned a weight of 0.9 since they usually connects to peering networks and tend to have large number of group members.

B. Implementation Issues

We have implemented a simplified version of SHOMA architecture in NS-2. In this implementation, the host proxy constructs Core-Based Trees rooted at itself when conducting group-tree matching. Inside the local cluster, member proxy distributes to end hosts a subset of group members inside the cluster, collects delay information from the end hosts, and then calculates a minimum spanning tree based on this delay information.

It is clear that the overlay network topology (eg, the location of overlay nodes and overlay links, and the selection of host proxies and member proxies) will affect the performance of SHOMA significantly. However, the construction of optimal overlay network topology is another problem and is out of the scope of this paper. Therefore, in this study, we choose to construct overlay network in a heuristic way. In the simulations using Transit-Stub synthetic topologies, we randomly select 80% of the transit nodes (i.e., 40 nodes) as member and host proxies. For each experiment, we repeat the same simulation with different sets of proxy nodes and take the average value for each metric. For the AT&T backbone topology, we select gateway routers (9 nodes) as member and host proxies. After the overlay nodes are determined, the overlay links are constructed based on the shortest paths between every pair of overlay nodes: if a shortest path goes through intermediate overlay node(s), then it is not eligible as overlay link. The

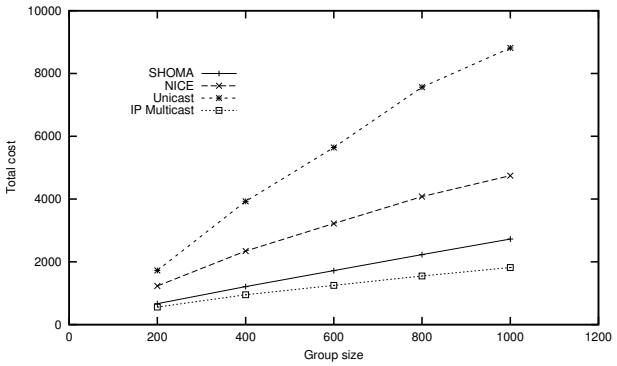

Fig. 8. Tree Cost vs. group size in Transit-Stub topology.

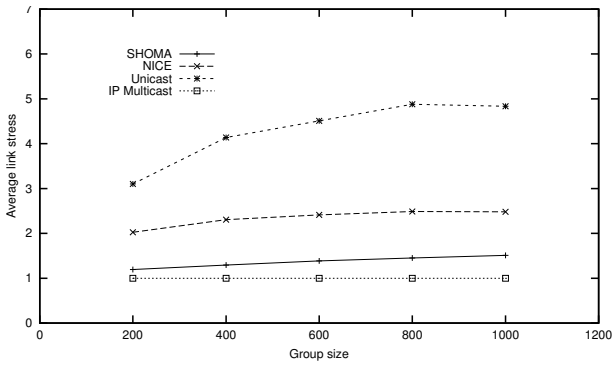

Fig. 9. Average stress vs. group size in Transit-Stub topology.

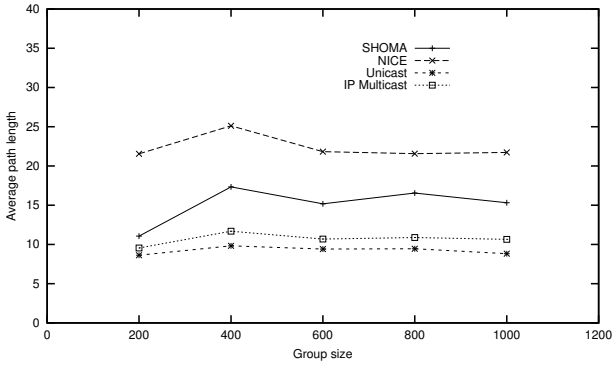

Fig. 10. Average path length vs. group size in Transit-Stub topology.

links that do not violate this constraint are preserved as overlay links.

C. Multicast Tree Performance

We use the following metrics to compare the multicast tree performance of these different schemes. *Multicast tree cost* is measured by the number of links in a multicast distribution tree. It quantifies the efficiency of the multicast routing schemes. Application level multicast trees and unicast paths may traverse an underlying link more than once, and thus they usually have a higher cost than IP multicast trees. To measure the quality of data paths, we randomly select a member as source and measure the link stress and path length when data are transmitted from source to all members on the multicast trees. *Link Stress* is defined as the number of identical data packets delivered over each link. IP multicast trees has the least link stress since only a single copy of a data packet is sent over each link. *Path Length* is the number of links from the source to a member. Unicast and shortest-path multicast schemes are usually optimized on this metric and thus have smallest path lengths.

In simulation experiments, a set of end hosts join the multicast group during an interval of 400 seconds. Then we collect the metrics after the multicast tree has stabilized. We found that SHOMA is able to converge within 10 seconds of simulation time after the join process is completed. In contrast, NICE tree needs hundreds of seconds of simulation time to stabilize, when the parameter is set as in [6].

Multicast tree cost in Transit-Stub topology We first use the Transit-Stub topology with 50 transit routers and 1000 stub routers to evaluate the performance of SHOMA. In Fig. 8, we plot the tree cost of SHOMA, NICE, and CBT as group size increases from 200 to 1000. As a reference, we also include the total link cost of the unicast paths from source to each member. Compared with the cost of unicast paths, NICE trees reduce the cost by 30-46%, SHOMA is able to reduce the cost by approximately 61-70%, and CBT trees save the cost by 68-80%. Clearly, the performance of SHOMA trees is close to that of the CBT, which represents the best performance that can be achieved by any application layer multicast or overlay multicast schemes. By using overlay proxies inside MSON, SHOMA outperforms NICE in all cases with respect to multicast tree cost, and the cost difference increases with group size.

Average link stress in Transit-Stub topology Fig. 9 shows the average link stress for these four schemes as the group size varies. IP multicast maintains a unit stress since no duplicate packets are transmitted on the same link. SHOMA trees exhibit average link stress between 1.19 and 1.51, whereas the average link stress of NICE trees is always higher than 2.00. In both SHOMA and NICE, the link stress do not vary greatly with different group size. However, unicast is not as scalable as SHOMA and NICE, since its link stress keeps increasing when group size grows.

Average path length in Transit-Stub topology As illustrated in Fig. 10, SHOMA trees show average path length that is closer to the two best schemes unicast and CBT than NICE. For instance, at group size 1000, SHOMA adds an additional latency of 4.67 hops to the path length of CBT (10.65 hops) on average, whereas NICE trees increase the latency by an average value of 11.07 hops.

Link stress distribution in AT&T topology We carry out the same set of experiments on AT&T backbone topology, and observe a similar trend as the Transit-Stub topology. As a further step, we plot the cumulative distribution of stress and path length in Fig. 11 and 12 when group size is 200. As shown in Fig. 11 it is evident that SHOMA and NICE outperform unicast in terms of link utilization efficiency, since the maximum

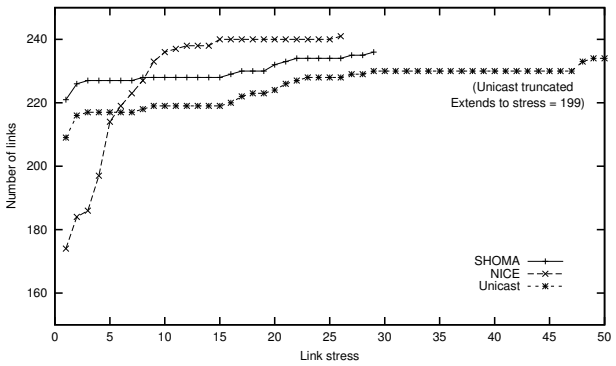

Fig. 11. Stress distribution for 200 members in AT&T topology.

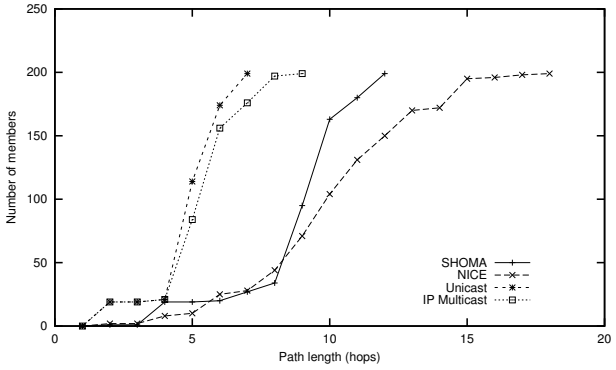

Fig. 12. Path length distribution for 200 members in AT&T topology.

link stress of these two schemes (i.e., 29 for SHOMA and 26 for NICE) is significantly smaller than that of unicast, which is 199. In addition, SHOMA uses fewer number of links than NICE, while its maximum link stress is comparable to that of NICE. In SHOMA, 231 links out of 236 links in total has unit stress, and only 9 links have stress higher than 5. In NICE, 174 links out of a total number of 241 links has unit stress, and 27 links have stress higher than 5.

Path length distribution in AT&T topology As for path length distribution, Fig. 12 again confirms our earlier observation that the path lengths from a random source to each member achieved by SHOMA is closer to shortest path length in unicast scheme: 81.9% members have path lengths within 10 hops. On the other hand, in NICE, approximately half group members use paths between 10 to 18 hops. We want to point out that when the group size increases, the metric difference between NICE and SHOMA is even more clearly observed, and thus more benefits can be achieved by using SHOMA for large groups.

D. Control Overhead

Recall that overlay multicast schemes (including SHOMA) generally induce less control overhead than application layer multicast schemes, since the control message exchange is restricted within limited number of nodes. NICE, a representative application layer multicast scheme, uses a hierarchical approach to organize group members into clusters, and the

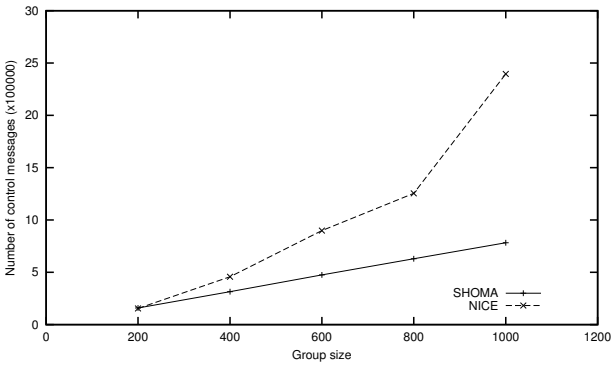

Fig. 13. Control Overhead for single group in Transit-Stub topology.

control overhead of every member is bounded by $O(klogN)$, where k is a parameter to determine cluster size, and N is the total number of group members [6]. Hence, we conduct experiments on the control overhead of SHOMA and NICE for a single group when group size varies.

Furthermore, by applying aggregated multicast and forcing multiple groups to share one aggregated tree inside MSON, we expect SHOMA to reduce multicast tree setup and maintenance overhead significantly when there are a large number of co-existing groups. To examine the amount of benefits achieved by using aggregated multicast, we compare the control overhead incurred by SHOMA with or without aggregated multicast in presence of large number of groups.

Control overhead for single group in Transit-Stub topology In this set of experiments, a set of end hosts join the multicast group during an interval of 400 seconds, and the multicast session ends at 1000 seconds. Then we collect the total number of control messages transmitted during 1000 simulation seconds. Since control overhead is closely related to values of user-defined parameters in both schemes, we set the parameters in NICE as their default values in the released code [2], and set the parameters in SHOMA correspondingly whenever possible. For example, the heartbeat period in NICE is set to 10 seconds; in SHOMA, the refresh period inside MSON and local clusters is also set to 10 seconds. We found out that in NICE, the total number of control messages increases very rapidly with group size, while in SHOMA, the increase is much more steady. At group size 1000, SHOMA induces only about one third the amount of the control messages produced by NICE.

Control overhead for multiple groups in AT&T topology To examine the effectiveness of aggregated multicast in reducing control overhead in presence of large number of simultaneously active groups, we compare two versions of SHOMA (aggregation-enabled and aggregation-disabled) with respect to the following two metrics: multicast tree setup overhead and maintenance overhead. In SHOMA, the establishment and tear-down of multicast trees are accomplished through the relay of *O-GRAFT* and *O-PRUNE* messages and multicast state update at intermediate proxies, so we quantify the multicast tree setup overhead by using the total number of *O-GRAFT*

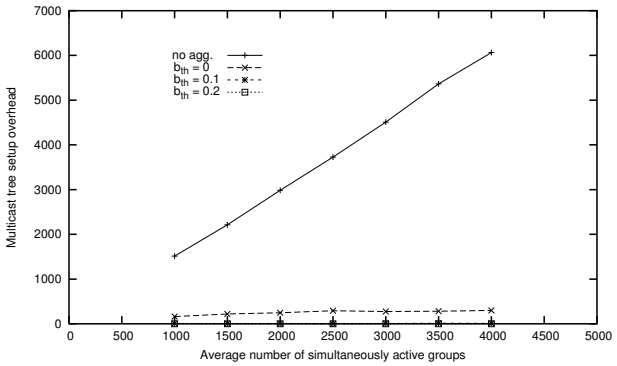

Fig. 14. Multicast tree setup overhead for multiple groups in AT&T topology.

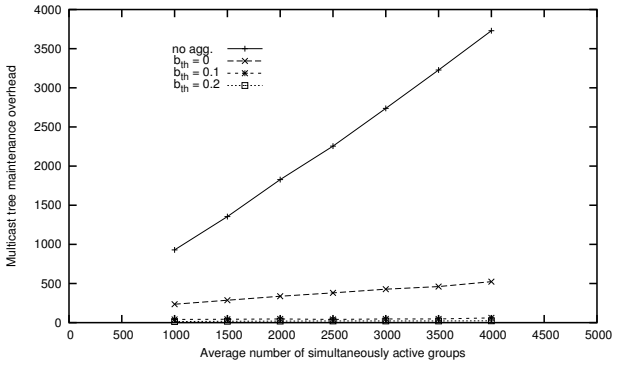

Fig. 15. Multicast tree maintenance overhead for multiple groups in AT&T topology.

and *O-PRUNE* messages. On the other hand, in each proxy, the multicast state is maintained as "soft state" and needs to be explicitly refreshed periodically. We collect the total number of refresh messages transmitted as the amount of tree maintenance overhead.

In the simulation experiments, we assume multicast group requests arrive as a Poisson process with arrival rate λ, and groups's lifetime has an exponential distribution with mean μ^{-1}. At steady state, the average number of groups is $\bar{N} = \lambda/\mu$. We fix average group life time as 100 seconds, and change the group arrival rate in order to get different number of groups. We run the simulation for 1000 seconds, and collect data after steady state is reached (after 400 seconds in our scenario). Note that the group-tree matching algorithm presented in Section III-E controls the bandwidth waste in leaky match with a threshold b_{th}, which we vary from 0 to 0.2 in the simulations.

Fig. 14 and 15 plot the results for multicast tree setup and maintenance overhead, respectively. In these figures, when aggregated multicast is disabled, the overhead increases very rapidly with the number of groups. On the contrary, the curves for aggregated multicast trees are relatively flat, showing that the number of multicast trees stabilize after the number of groups reaches a certain value. We also observe that the more multicast groups become active, the more communication cost is reduced in both figures. When b_{th} is raised from 0 to 0.2,

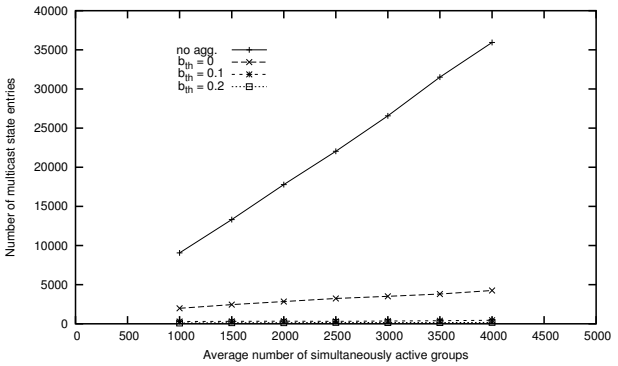

Fig. 16. Multicast state for multiple groups in AT&T topology.

the control overhead is further reduced, though at the cost of higher bandwidth waste. This provides overlay ISPs with flexibility of controlling the trade-off of bandwidth cost versus control overhead by setting appropriate b_{th}.

E. Multicast State Scalability

To investigate the multicast state scalability of SHOMA, we perform the same set of experiments for multiple groups in AT&T topology as explained in previous section, and plot the results for number of multicast forwarding state entries vs. the average number of simultaneously active groups in Fig. 16. This figure exhibit a similar trend as Fig. 14 and 15: more reduction in multicast state when there are more concurrent groups and more bandwidth is wasted. This figure demonstrates that SHOMA is scalable to the number of co-existing groups.

In conclusion, our observations through simulation experiments can be summarized as follows: SHOMA creates multicast distribution trees with tree cost and average link stress almost comparable to that of CBT; the data paths on SHOMA trees have lower latency than those on NICE trees; the control overhead of SHOMA is significantly less than NICE for large groups; SHOMA keeps multicast tree setup and maintenance overhead low in presence of large number of simultaneously active groups.

VI. Related Work

A. Application-layer Multicast

Recently application layer multicast has emerged as a new architecture to apply multicast paradigm in the Internet. The proposed schemes can be classified into two categories: structured [30] [37] [9], and unstructured [12] [6] [28] [26] [22]. Structured application layer multicast schemes leverage Distributed Hash Table (DHT)-based overlay networks and build multicast forwarding trees on top of this structured overlay. Here we concentrate on unstructured application layer multicast schemes because they are more related to our work.

End System Multicast [12] [11] and Application Level Multicast Infrastructure (ALMI) [28] are targeted at applications with small and sparse groups, such as audio and video con-

ferences. In End System Multicast, end hosts periodically exchange group membership information and routing information, build a mesh based on end-to-end measurements, and run a distributed distance vector protocol to construct a multicast delivery tree. The authors also demonstrate the importance of optimizing and adapting the overlay to application-specific requirements such as latency and bandwidth. ALMI uses a centralized entity to collect membership information, periodically calculate a minimum spanning tree based on the measurement updates received from all members, and distribute this information to group members.

NICE [6], on the other hand, is designed to support applications with very large receiver sets and relatively low bandwidth requirements. It recursively arrange group members into a hierarchical overlay topology, which implicitly defines a source-specific tree for data delivery. It has been shown that NICE scales to large groups in terms of control overhead and logical hops.

Other application layer multicast protocols include TAG [26] and Yoid [22]. TAG exploits the underlying network topology when constructing application-layer multicast trees. In Yoid, each member is responsible for discovering and selecting a parent, and thus multicast trees are constructed without underlying mesh.

B. Overlay Multicast

Overlay multicast networks provide multicast services through a collection of strategically placed network proxies. A number of architectures on this topic have been proposed. Overcast [25] provides reliable single-source multicast by using a distributed protocol to build data distribution trees rooted at a central source. This root is responsible for redirecting a client's HTTP requests to a Overcast node, from which the client receives data using TCP connection. RMX [10] provides reliable data delivery to heterogeneous end users by using a set of RMX proxies that are organized into a spanning tree. The end users are split into a number of locally-scoped multicast data groups of homogeneous members, each of which contains a RMX proxy. Each RMX proxy communicates with peer proxies using TCP and uses simple multicast congestion control within its data group.

[32] [31] and [7] focus on optimizing the end-to-end delay and access bandwidth usage at the Multicast Service Nodes. Shi et al proposes a set of heuristic algorithms to solve minimum-diameter degree-limited spanning tree problem and bounded-diameter residual-balanced spanning tree problem [32] [31]. The authors of [7] formulate the problem as minimum average-latency degree-bounded spanning tree problem and proposed an iterative distributed algorithm.

Our work is different from above application-layer and overlay multicast schemes. We have proposed a complete solution for providing efficient multicast service, including a two-tier architecture that scales to both the number of groups and group size, and a feasible pricing scheme that provides incentives for both overlay ISPs and end users to adopt this service model.

VII. Conclusions and Future Work

We propose and develop an overlay multicast architecture to support large scale multicast applications in an efficient and scalable way. Our contribution could be summarized as follows:

- We adopt overlay multicast scheme in the MSON and application layer multicast outside MSON to provide efficient resource utilization and reduced control overhead
- By applying aggregated multicast inside overlay network, the control overhead for establishing and maintaining multicast trees can be further reduced, and significantly less forwarding state needs to be maintained at proxy nodes.
- We suggest a pricing scheme that is simple to use, and is able to stimulate the ISPs and customers to deploy and purchase related services.
- We show that our approach is very promising in a series of simulation experiments. We can achieve scalability with regard to group size and number of co-existing groups.

References

[1] In *Unpublished, available at http://www.cs.berkeley.edu/yatin/papers/*.

[2] *myns (P2P) simulator alpha release version 0.1*. http://www.cs.umd.edu/ suman/research/myns/index.html.

[3] *The Network Simulator - NS-2*. http://www.isi.edu/nsnam/ns/.

[4] *AT&T IP Backbone*, 2001. http://www.ipservices.att.com/backbone/.

[5] A. Ballardie. Core Based Trees (CBT version 2) multicast routing: protocol specification. *IETF RFC 2189*, September 1997.

[6] S. Banerjee, C. Kommareddy, and B. Bhattacharjee. Scalable application layer multicast. In *Proceedings of ACM SIGCOMM*, August 2002.

[7] 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*, April 2003.

[8] K. Calvert, E. Zegura, and S. Bhattacharjee. How to model and internetwork. In *Proceedings of IEEE INFOCOM*, March 1996.

[9] 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, October 2002.

[10] Y. Chawathe, S. McCanne, and E. A. Brewer. RMX: Reliable multicast for heterogeneous networks. In *Proceedings of IEEE INFOCOM*, March 2000.

[11] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *Proceedings of ACM SIGCOMM*, August 2001.

[12] Y.-H. Chu, S. G. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM Sigmetrics*, June 2000.

[13] J. Chuang and M. Sibru. Pricing multicast communications: A cost-based approach. *Proceedings of the Internet Society INET'98 Conference*, July 1998.

[14] L. H. M. Costa, S. Fdida, and O. C.M. Duarte. Hop-by-hop multicast routing protocol. In *Proceedings of ACM SIGCOMM*, August 2001.

[15] J.-H. Cui, J. Kim, A. Fei, M. Faloutsos, and M. Gerla. Scalable QoS multicast provisioning in Diff-Serv-supported MPLS networks. In *Proceedings of IEEE GLOBECOM*, November 2002.

[16] J.-H. Cui, L. Lao, D. Maggiorini, and M. Gerla. BEAM: A distributed aggregated multicast protocol using bi-directional trees. In *Proceedings of IEEE ICC*, May 2003.

[17] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. The PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, April 1996.

[18] Stephen Deering. Multicast routing in a datagram internetwork. *Ph.D thesis*, December 1991.

[19] S. Fahmy and M. Kwon. Characterizing overlay multicast networks. In *Proceedings of IEEE ICNP*, November 2003.

[20] A. Fei, J.-H. Cui, M. Gerla, and D. Cavendish. A "dual-tree" scheme for fault-tolerant multicast. In *Proceedings of IEEE ICC*, June 2001.

[21] Aiguo Fei, Jun-Hong Cui, Mario Gerla, and Michalis Faloutsos. Aggregated Multicast: an approach to reduce multicast state. *Proceedings of Sixth Global Internet Symposium(GI2001)*, November 2001.

[22] P. Francis. *Yoid: Extending the Multicast Internet Architecture*. White papar, http://www.aciri.org/yoid/.

[23] S. Herzog, S. Shenker, and D. Estrin. Sharing the "cost" of multicast trees: An axiomatic analysis. In *Proceedings of ACM SIGCOMM*, August 1995.

[24] Hugh Holbrook and Brad Cain. Source-Specific Multicast for IP. *Internet draft: draft-holbrook-ssm-arch-03.txt*, November 2001.

[25] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*, October 2000.

[26] M. Kwon and S. Fahmy. Topology aware overlay networks for group communication. In *Proceedings of NOSSDAV'02*, May 2002.

[27] M. Medard, S. Finn, R. Barry, and R. Gallager. Redundant trees for preplanned recovery in arbitrary vertex-redundant or edge-redundant graphs. *IEEE/ACM Transactions on Networking*, 7(5):641–652, October 1999.

[28] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of the 3rd USNIX Symposium on Internet Technologies and Systems*, March 2001.

[29] P. I. Radoslavov, D. Estrin, and R. Govindan. Exploiting the bandwidth-memory tradeoff in multicast state aggregation. Technical report, USC Dept. of CS Technical Report 99-697 (Second Revision), July 1999.

[30] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proceedings of NGC*, November 2001.

[31] S. Shi and J. S. Turner. Routing in overlay multicast networks. In *Proceedings of IEEE INFOCOM*, June 2002.

[32] S. Shi, J. S. Turner, and M. Waldvogel. Dimensioing server access bandwidth and multicast routing in overlay networks. In *Proceedings of NOSSDAV'01*, June 2001.

[33] I. Stoica, T.S. Ng, and H. Zhang. REUNITE: A recursive unicast approach to multicast. In *Proceedings of IEEE INFOCOM'00*, Tel Aviv, Israel, March 2000.

[34] D. Thaler and M. Handley. On the aggregatability of multicast forwarding state. *Proceedings of IEEE INFOCOM*, March 2000.

[35] J. Tian and G. Neufeld. Forwarding state reduction for sparse mode multicast communications. *Proceedings of IEEE INFOCOM*, March 1998.

[36] Z. Wang and J. Crowcroft. Bandwidth-delay based routing algorithms. In *Proceedings of IEEE GLOBECOM*, November 1995.

[37] 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.