Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Networks 50 (2006) 781–806
Computer Networks
www.elsevier.com/locate/comnet

Universal IP multicast delivery

Beichuan Zhang [a,*], Wenjie Wang [b], Sugih Jamin [b], Daniel Massey [c], Lixia Zhang [d]

[a] Computer Science Department, University of Arizona, Tucson, AZ 85721-0077, USA
[b] EECS Department, University of Michigan, Ann Arbor, MI 48109-2122, USA
[c] Computer Science Department, Colorado State University, Fort Collins, CO 80523-1873, USA
[d] Computer Science Department, UCLA, Los Angeles, CA 90095-1596, USA

Available online 2 September 2005

Abstract

A ubiquitous and efficient multicast data delivery service is essential to the success of large-scale group communication applications. The original IP multicast design is to enhance network routers with multicast capability [S. Deering, D. Cheriton, Multicast routing in datagram internetworks and extended LANs, ACM Transactions on Computer Systems 8(2) (1990) 85–110]. This approach can achieve great transmission efficiency and performance but also poses a critical dependency on universal deployment. A different approach, overlay multicast, moves multicast functionality to end hosts, thereby removing the dependency on router deployment, albeit at the cost of noticeable performance penalty compared to IP multicast. In this paper we present the Universal Multicast (UM) framework, along with a set of mechanisms and protocols, to provide ubiquitous multicast delivery service on the Internet. Our design can fully utilize native IP multicast wherever it is available, and automatically build unicast tunnels to connect IP Multicast "islands" to form an overall multicast overlay. The UM design consists of three major components: an overlay multicast protocol (HMTP) for inter-island routing, an intra-island multicast management protocol (HGMP) to glue overlay multicast and native IP multicast together, and a daemon program to implement the functionality at hosts. In addition to performance evaluation through simulations, we have also implemented parts of the UM framework. Our prototype implementation has been used to broadcast several workshops and the ACM SIGCOMM 2004 conference live on the Internet. We present some statistics collected during the live broadcast and describe mechanisms we adopted to support end hosts behind Network Address Translation (NAT) gateways and firewalls.

Keywords: IP multicast; End-host multicast; Overlay multicast

* Corresponding author.
E-mail addresses: bzhang@cs.arizona.edu (B. Zhang), wenjiew@eecs.umich.edu (W. Wang), jamin@eecs.umich.edu (S. Jamin), massey@cs.colostate.edu (D. Massey), lixia@cs.ucla.edu (L. Zhang).

1389-1286/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.comnet.2005.07.016

1. Introduction

IP multicast [16] is designed to provide efficient and high performance data delivery to potentially large numbers of receivers. However, its deployment not only requires *router support* from all Internet service providers, but also raises new issues and challenges in network control and management. As a result, the full deployment of IP multicast has been long in coming. Although alternate approaches (e.g., [36,29,40,14,19]) have been proposed to simplify IP multicast implementation and alleviate the management issues, they do not remove the router dependency. Today's Internet only has spotted IP multicast deployment *within* local area networks, at individual campuses, and by a handful of service providers. Lack of ubiquitous IP multicast support in today's Internet hinders the development of multicast applications, which in turn reduces incentives for IP multicast deployment.

In response to the slow deployment of IP multicast, a number of application-layer multicast mechanisms have been developed. We can sort these mechanisms into two categories: end-host multicast and multicast server overlay. In end-host multicast, group members (usually end-hosts) are organized to replicate and forward packets to each other, without any reliance on router support. However this approach incurs performance penalties because, depending on the locations of individual hosts, packets are likely to travel along sub-optimal paths, and packets may traverse the same links multiple times. Furthermore, since end-hosts are owned by individual users, they are less stable and less trustworthy than routers. In multicast server overlay, dedicated servers need to be placed over the Internet by application service providers. These servers receive packets from the data source and forward them to individual receivers via unicast delivery. Multicast server overlay can provide better performance and stability, at the expense of deployability. None of these approaches makes use of native IP multicast delivery that is available in multicast-enabled islands which widely exist, including enterprise networks, campus networks, or last hop Ethernet connectivity.

In this paper we propose the Universal Multicast (UM) framework and its associated set of mechanisms and protocols. Our goal is to provide ubiquitous multicast delivery service and to utilize IP multicast wherever it is available. UM offers scalable and efficient end-host multicast support in places where native IP multicast does not exist, and incorporates various infrastructure multicast support automatically. This approach enables ubiquitous multicast delivery service immediately, thus breaking the deadlock between application development and network deployment. At the same time, it utilizes existing multicast infrastructure support to improve both end-user performance and bandwidth usage efficiency, thus encouraging further infrastructure support deployment where such support provides benefit. Instead of debating whether the Internet infrastructure would, or should, deploy IP multicast support, UM simply takes advantages from existing support and leaves the final decision to the need of applications and service providers as the Internet evolves. In addition, although this paper presents a specific end-host multicast protocol, the UM framework does not rely on any specific end-host multicast protocol or IP multicast routing protocol. As both end-host multicast and network multicast are in constant development and deployment, protocol independence gives UM great flexibility to accommodate different network environments and new protocols in the future.

Our partial prototype implementation of the UM framework has been used to broadcast several workshops and the ACM SIGCOMM 2004 conference live on the Internet. Quite unexpectedly, a major stumbling block we had to overcome in rolling out a *universal* multicast framework turned out to be supporting hosts behind firewalls and Network Address Translation (NAT) gateways [17]. We call such hosts *guarded hosts*. We also did not fully appreciate the large and increasing number of hosts whose access links have asymmetric bandwidths. Supporting both guarded hosts and hosts with asymmetric bandwidths led us to augment our multicast tree building protocol with various network measurement and topology constraint detection mechanisms. One lesson learned is that overlay multicast protocols must take into account whether a host is a guarded host or one

with asymmetric bandwidth when building multicast trees or meshes.

The rest of the paper is structured as follows. Section 2 describes the overall architecture and system components, Section 3 presents a simple and scalable end-host multicast protocol, HMTP, and Section 4 presents a management protocol that glues end-host multicast and native IP multicast together. Section 5 describes our prototype implementation and live broadcasting experiments on the Internet. We discuss related work in Section 6, and conclude in Section 7.

2. Overview

Like IP multicast, Universal Multicast is not made of a single protocol but a set of components at various levels. Fig. 1 shows a sample UM group illustrating the architecture. In UM's view, the Internet is composed of multiple IP-multicast-enabled ''islands'', which are separated by unicast-only routers. Every group member runs a daemon program (Agent) in user space to provide the UM functionality. Of each multicast group, one or more group members in each island are elected as designated members (DM) which build dynamic unicast tunnels connecting to other islands. Application data are transmitted via native IP multicast inside islands and encapsulated in unicast packets to flow through tunnels from one island to another. The current UM design focuses on providing the best-effort multicast data delivery service on the Internet. As demonstrated in our real-world experiments, it is also important to consider the quality of service for applications. Adding QoS support will be a major future work in the UM architecture.

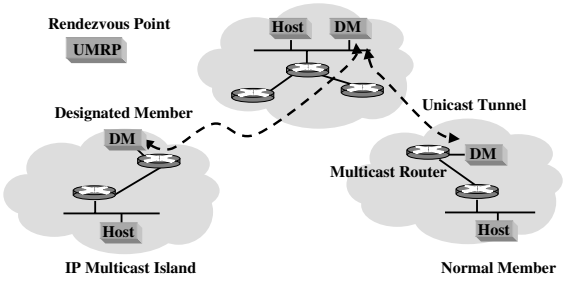

Fig. 1. Universal multicast architecture.

2.1. Island

An island is a network of any size that supports IP multicast. It can be as large as a network with multiple domains (e.g., the MBone), or as small as a single host. If a host is totally isolated from IP multicast connectivity, it is an island by itself. An island's boundary does not need explicit configuration; it is simply the furthest extent that an IP multicast packet can reach in the network. If administrators enable more IP multicast routers, the island boundary will automatically expand accordingly as IP multicast packets reach a wider area.

2.2. Designated member

Of each multicast group, one or more members are automatically elected as designated members (DM) in each participating island. A DM's main tasks are running an end-host multicast routing protocol to build inter-island tunnels to reach DMs in other islands, and forwarding packets in and out of the island on behalf of the group. When there are multiple DMs in the same island, coordination is needed among them.

2.3. Rendezvous point (UMRP)

UM uses a ''rendezvous point'' to achieve the goal of dynamic group membership support. Each group has a rendezvous point, called universal multicast rendezvous point (UMRP), whose main purpose is to help a new DM join the inter-island overlay. UMRP keeps information of which end-host multicast protocol the group is using, and corresponding protocol-specific information. A new DM queries the UMRP first to learn the routing protocol and its parameters. It then runs the protocol to join the inter-island overlay. UMRP is only used to bootstrap new DMs; it is not involved in subsequent overlay construction or data packet forwarding, thus its location has no impact on the performance of data delivery. One UMRP can serve multiple multicast groups. It can be a

normal end host, a dedicated server, or a cluster of servers. Failure of UMRP prevents new DMs from joining the inter-island overlay, but does not affect data dissemination among existing islands. For important groups, UMRP's reliability can be improved by adding some redundancy.

2.4. Group identifier (GID)

Since the current Internet lacks a global address allocation scheme for multicast, we cannot simply use traditional class D IP addresses to identify groups. Instead, UM uses the combination of UMRP's unicast IP address and a 32-bit *group number* as the group identifier, called *GID*. The group number is assigned by UMRP upon the creation of the group and is unique among all groups served by the same UMRP. Therefore, GID can uniquely identify a group globally. If a user-friendly *group name* is used in place of the group number, UMRP will be responsible for resolving group name to group number. The GID is obtained off-line by users or applications, and passed to the multicast application to join the group. For example, to join a UM group named *forest.cs.ucla.edu/mytalk*, applications will use DNS to resolve the UMRP *forest.cs.ucla.edu* to *131.179.96.162*, and then contact the UMRP to resolve the group name *mytalk* to a group number *1234*. Thus the GID is *131.179.96.162/1234*. It is then used in both data packet and routing packet header to identify the group.

2.5. Local multicast groups

Though GID uniquely identifies a UM group, IP multicast enabled applications, operating systems and routers only understand class D IP addresses. Within each island we use native IP multicast groups for multicast delivery. For each UM group, every island with active group members will have three local IP multicast groups: *DATA_GROUP* for transmitting data packets, *ASSERTION_GROUP* for electing DMs, and *DM_GROUP* for coordinating multiple DMs. These local groups are created by the DM in each island using any address allocation scheme available in the island. Local multicast groups of one island are totally independent from those of other islands. The mapping between a GID and the corresponding three local groups is done by the *Group Directory*. Similar to the multicast session directory [26], the Group Directory is a well-known IP multicast address plus a well-known port number. In each island, the DM multicasts the mapping information to this special group, and other members can tune in to learn the mapping information.

The key difference between the MBone and UM is that the former is manually configured and managed by network administrators, while the latter is fully self-organized by end hosts. We designed two protocols to accomplish this: host multicast tree protocol (HMTP) (Section 3) for dynamically establishing/removing tunnels, and host group management protocol (HGMP) (Section 4) for automatically electing and coordinating DMs.

3. Host multicast tree protocol (HMTP)

Existing end-host multicast protocols can be categorized into tree-based and mesh-based protocols by the type of overlay they build. A tree is an overlay where there is a single path between any node pair, while a mesh may support more than one path between any node pair. BTP [28], TBCP [33], and Yoid [21] are examples of tree-based protocols; Narada [12] and Hypercast [31] are examples of mesh-based protocols. A mesh makes use of more overlay links; thus it can provide shorter end-to-end delay. However, it incurs more routing overhead and requires that each node maintain more routing states. HMTP is a tree-based end-host multicast protocol, which builds an efficient and scalable group-shared tree for multicast delivery. If there is a need for shorter end-to-end delay, as [43] has shown, a mesh can be "grown" on top of HMTP tree by adding some additional overlay links.

To reduce routing inefficiency, an overlay should be congruent to the underlying network topology to the furthest extent possible. In our design, however, we assume that members have no knowledge of the underlying physical network. While tools such as `traceroute` are available

for discovering such information, they are usually not very dependable because intermediate routers could block their probes. Furthermore, running such tools prior to data delivery may take longer than the data delivery time itself. Hence without knowing the underlying physical network topology, end-hosts use end-to-end measurements of some network properties to serve as distance metric. Currently we use member-to-member round-trip time (rtt) as the only distance metric in tree building. End-hosts can obtain rtt estimates by actively probing other hosts, querying distance services like IDMaps [23], or calculating from node coordinates [35,15], etc. In the future we may add bottleneck bandwidth as a second metric.

3.1. Protocol design

3.1.1. Join

So as to reduce the total cost of the tree and to maintain a maximum degree constraint at every host, HMTP tries to cluster nearby members together. The simplest way to achieve such clustering would be to let each new member[1] choose as its parent an existing member closest to it. This simple scheme, however, requires someone to maintain a list of all group members. Such a list is not necessary if each new member chooses as its parent only the closest from a random subset of existing members. A tree so generated, however, will have a very random (and most likely grossly sub-optimal) structure. In HMTP we define some rules to generate a partial list of existing members. The rules ensure that when a new member joins the tree by choosing the closest member from this list as its parent, the resulting tree will not be totally random.

In Fig. 2, node H is a newcomer joining a group. It knows of the group's UMRP from the group's GID. By querying the UMRP, it learns that node A is the root of the tree. H sets A as its potential parent and asks A for a list of A's children. From the list of A and A's children, H picks the closest one, in this case D, as a new potential parent. H repeats this process and finds the next potential parent, F. In the next iteration, H finds that the new potential parent remains F. So H attempts to make F its parent by sending a join request to F. It is the potential parent who decides whether to accept a join request, based on its policy, bandwidth, traffic load, etc. If F rejects H's request, H marks F as invalid and goes back up one level and resumes the search for a parent. It eventually finds G. The detail algorithm is described in Fig. 3. A node normally is configured with a maximum degree constraint to prevent it from accepting too many children. It is not a protocol design parameter, but a tunable factor based on each individual node's bandwidth. Larger maximum node degree makes the tree structure more compact, thus shortening the delay, but it requires more bandwidth for the application.

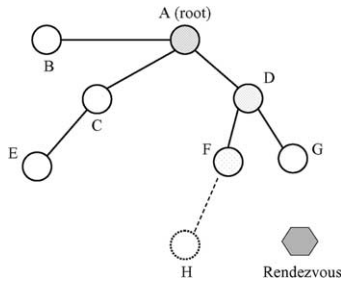

Fig. 2. H joins a tree consisting of A–G.

To summarize, a newcomer tries to find a good parent by searching a small part of the tree. It stops when it reaches a leaf node, or a node that is closer than all its neighbors. The algorithm scales to the number of group members. It does not guarantee that the parent chosen is the nearest one among all members; however, since all members follow the same rule, the distance information is encoded into the tree structure and should help every newcomer. For example, two nearby hosts will have similar delay to other hosts, so they are likely to make the same decision at each step. When one of them joins the tree, it may not choose the nearest existing member. But when the second one joins, it probably will choose the first one as its parent. Therefore nearby members are more likely to be clustered together.

Clustering nearby members makes the tree structure congruent to the network topology to

[1] In UM, only designated members participate in tree construction and maintenance. In this section, we use "member" to mean "designated member" for ease of exposition.

1) All members are regarded as valid potential parents by default. A stack S is allocated for storing past potential parents.
2) Find the root by querying UMRP. Set the root as potential parent.
3) Query the potential parent to discover all its children. Measure rtts to the potential parent and its children.
4) Find the nearest member among the potential parent and all its children, except those marked as invalid. If all of them are invalid, pop the top element in S and make it the new potential parent, return to step 3.
5) If the nearest member is not the current potential parent, push current potential parent onto the stack, set the nearest member as the new potential parent, return to step 3.
6) Otherwise, send the potential parent a join request. If refused, mark the potential parent invalid and return to step 4. Otherwise parent found, establish a unicast tunnel to it.

Fig. 3. Join algorithm.

the first order. Links with large latency will be traversed fewer times. Members behind such links (e.g., members on dial-up lines) are likely to be pushed to the edges of the tree. Given the same set of members but different join sequences, the protocol will produce different trees. However, with periodic tree improvement (Section 3.1.4) run by every member, these trees will eventually converge to some stable structures that have similar gross qualities. For example, if a group has a dial-up host with large last-hop delay, and some other well-connected hosts with relatively short delay, the well-connected hosts should form the core of the tree and the dial-up host will connect to the core by a single link after the tree stabilizes, regardless of the members' join order. Even if the dial-up host is the first member to join the tree and becomes the root, the shape of the resulting tree should be similar with similar gross quality. This is confirmed by our simulations with random join sequences.

3.1.2. Maintenance

States in HMTP are refreshed by periodic message exchanges between neighbors. Every child sends REFRESH messages to its parent. The parent replies by sending back PATH messages. From the root of the tree to each member, there is only one, loop-free path along the tree. The member list of this path is called the *root path*. The PATH message sent by the parent contains the root path of the parent. By appending itself to its parent's root path, a member constructs its own root path. Every member must maintain the freshness of its list of children and its root path. The root sends REFRESH messages to UMRP, so that UMRP always knows who is the current root.

3.1.3. Leave

When a member leaves a group, it notifies its parent and children. Its parent simply deletes the leaving member from its children list. It is the leaving member's children's responsibility to find new parents. A child looks for a new parent by running the join algorithm in reverse order (Fig. 4). If the root is leaving, its children contact UMRP after a random delay. The first member to contact UMRP will be assigned as the new root.

3.1.4. Improvement

As network conditions and group membership change over time, members may need to restructure the tree, by switching to new parents, to improve performance. Parent switching in HMTP is done by periodically re-running the join procedure (Fig. 3). To reduce not only the workload of members near the root, but also the time needed to complete tree restructuring, members do not start the re-joining procedure from the root, but from a randomly picked node in their root paths. Furthermore, in step 4 of the re-join procedure, a member other than the closest one could be picked as the next potential parent. This will allow members to explore other branches of the tree for a

1) Push root path into the stack S. Delete self and current parent (which is leaving or has crashed) from the top of the stack.
2) Pop grandparent as potential parent from the stack, wait for a random delay, then start running the join algorithm from step 3, until a new parent is found. The random delay is used to prevent all orphans from contacting the grandparent at the same time.

Fig. 4. Repair algorithm.

better parent. After finding a new parent, a member may switch to it if the new parent is closer than the current one.

Periodically running tree improvement raises a stability concern. The fundamental trade-off is between keeping the tree structure stable and adapting it to the changing network environment. In HMTP, tree improvement runs every several minutes. The actual frequency varies for each member. A member that already has a close parent can run tree improvement less often. We also enforce a threshold on delay gain in triggering a parent switch. This can avoid oscillation caused by transient network changes. After the parent switch, a member should push a PATH message down to its sub-tree to update its descendants' root paths quickly, instead of waiting for regularly scheduled PATH messages. These tune-ups make the HMTP tree both stable and adaptive.

3.1.5. Partition recovery

When a non-leaf member crashes, the tree is partitioned. Surviving members must be able to detect the failure and repair the tree. Repairing the tree upon a member crash is similar to handling a member leave. For example, when the root node fails, its children will contact UMRP, and one of them will be assigned as the new root. The difference in this case is that surviving members usually do not receive prior notification of the crash. Hence node failure is detected by noticing repeatedly missing REFRESH or PATH messages. When a node failure is detected, the parent of the failed node simply updates its children list; the children of the failed node must run the repair algorithm (Fig. 4). As long as a node on its root path or the UMRP is available, a partitioned member can always re-join the tree.

In the rare cases when all hosts in a member's root path *and* the UMRP fail at the same time, the best thing this member can do is to try connecting to other group members in its cache. In the process of joining the tree or during tree improvement, a member learns of members in other branches of the delivery tree. These members can be cached and used for partition recovery. Members with short root path may want to cache more such members. Using cached members does not, however, guarantee that the tree can be reconnected. In the worst case, the group could be partitioned into pieces that cannot find each other. Existing end-host based overlay networks usually require every member to maintain an accurate and complete member list to guarantee partition recovery. Such a requirement limits the scalability of these protocols. HMTP achieves partition recovery as long as there is a surviving host in a member's root path or the UMRP is not down.

3.1.6. Loop detection and resolution

If a member switches to one of its descendants, a routing loop is formed. Members in a loop will see itself in its root path. To prevent loop formation, a member checks that it is not in its potential parent's root path before settling on a new parent. With this simple algorithm, loops can still be formed if there are concurrent topology changes, e.g., in Fig. 2, if C joins F and D joins E at the same time, a loop will form. When there is a loop, a member in the loop will detect it after seeing itself in the root path. In a tree structure the existence of a loop also means a tree partition. Hence the member detecting a loop immediately breaks the loop by leaving its current parent, and re-joins the tree starting from the root node. If multiple members detect the loop at the same time, the loop will be broken into multiple pieces, each piece is loop-free and will re-join the tree independently. The loop detection and resolution will take some time. For detection, it depends on how frequently the root path is updated (by the PATH messages); for resolution, it is the same as that of joining the tree. With HMTP, the formation of a loop requires multiple conflicting topology changes happen at (relatively) the same time. Thus loops, if they occur at all, should be relatively rare, and the overall overhead of loop detection and resolution should be small.

3.1.7. Join delay and foster care

One problem of the basic join algorithm is long join latency, especially for large group size and sparse groups. To reduce join latency, we allow new members to temporarily attach to a random member. Existing members must accept a limited number of temporary (foster) children for a short

period of time. After that time, the parent can remove the foster children. Data packets are forwarded to all of a node's children, including its foster children. However, a node does not list its foster children in reply to a join query (step 3 of Fig. 3).

3.1.8. U-turn and triangle optimization

Suppose there are three hosts A, B, and C, attached to three routers X, Y, and Z respectively (Fig. 5), and the delays between A, B, and C are $D_{AC} > D_{AB} > D_{BC}$. When constructing a spanning tree among these three hosts, we prefer to discard the longest overlay link AC. However, since end hosts do not have knowledge about the router topology (i.e., there is no physical link between X and Z), they may make wrong decisions; for example, suppose A initiates a new multicast group. It is the first member of the group and becomes the tree's root. If C joins before B, the tree becomes A–C–B. Packets from A go to C first, then are forwarded back to B. We call this the *U-turn* problem. Since B is currently a child of C, C will never use B as a potential parent. And since the B–C distance is smaller than the B–A distance, B will always pick C as parent. So tree improvement cannot solve this U-turn problem.

Our solution is to give B more information to make the correct decision when it joins the tree. When A passes its children list to B, it also includes the delay from A to all its children (in this case, D_{AC}). B measures the delay D_{AB} and D_{BC} following the join algorithm. Now B knows all three link delays. The *triangle optimization* states that, B will choose A as its parent unless D_{AB} is the longest one among the three virtual links. After B joins A, C will discover B during tree improvement and since D_{BC} is shorter than D_{AC}, C will switch to B, thus the U-turn problem is avoided. In general, a node's latencies to all of its children are always available because of the periodic exchanges of REFRESH and PATH messages. The triangle optimization is applied at each step of the join and tree improvement processes. In the case that the potential parent has more than one existing children, the newcomer shall apply triangle optimization between the potential parent and the child who is the nearest to the newcomer.

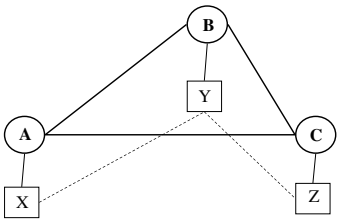

Fig. 5. X, Y and Z are routers; A, B and C are end hosts; $D_{AC} > D_{AB} > D_{BC}$.

3.1.9. Server backbone

Using end users' computers to forward packets makes the design deployable. For applications where performance and stability are critical, however, there is a need of using dedicated servers to provide the multicast delivery service. UM does not require the use of dedicated servers, but it can easily incorporate them if there are servers available. One approach is to associate each host with a *rank* when it joins the HMTP tree. A node will only join a parent whose rank is equal to or higher than its own. A parent can ask a child to leave if there is a need to make room for another child with higher rank than the old child's. Normal end hosts have default the lowest rank, while servers are configured with higher rank. Security measures may be implemented to prevent false claims of one's rank. But, it is up to applications and is orthogonal to UM design. After joining the HMTP tree, dedicated servers will occupy the top of the tree, while regular hosts are pushed down to the edge. Thus we will automatically have a stable server backbone as the core to serve the group.

3.2. Performance evaluation

We conducted simulations to evaluate the performance of HMTP against naive unicast and IP multicast. For IP multicast, we use both shortest path source tree (SPST, as used by DVMRP) and shortest path group tree (SPGT, as used by CBT).

3.2.1. Metrics

The quality of a tree is judged by the following metrics: tree cost, delay penalty, and link load.

Tree cost is the sum of all the tree link latencies. It is a convenient, though somewhat simplified, metric to capture total network resource consumption of a tree. The ratio of a tree's cost to that of a corresponding SPST is the tree's *cost ratio*.

Delay penalty measures how much the overlay stretches end-to-end latency. *Relative delay penalty* (*RDP*) is the ratio of the latency between a node pair on the overlay to the latency between them on the physical network. *ARDP* is then the average RDP over all node pairs: The smaller the delay penalty, the closer node-pair latencies on the overlay are to latencies on the physical network. *Group diameter* is the maximum delay between any node pair. It represents the time after which packets are assured to reach every member. The ratio of group diameter to the diameter of an SPST is the *group diameter inflation*.

Link load of a physical link is the number of duplicates the link has to carry when a packet is multicast to the group. IP multicast has load of 1 on all links, while overlay multicast has load greater than 1 on some links.

Results presented in this section are based on simulations on a network topology consisting of 1000 routers, and 3300 links. Some additional nodes as end hosts are randomly attached to routers. The maximum node degree constraint when running HMTP is set to eight. Except for some results from a single run (Figs. 6, 8, and 11), data points represent averages over 100 runs with 95% confidence interval.

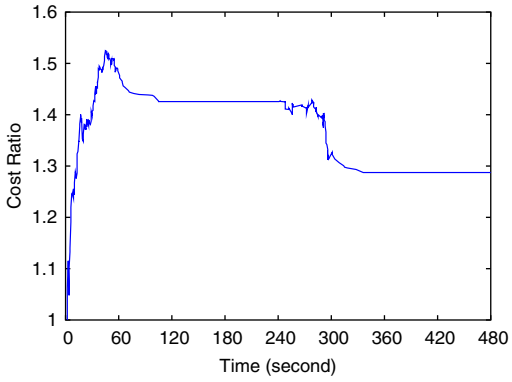

Fig. 6. HMTP tree cost ratio vs. time, with members join and leave.

3.2.2. Simulations

Fig. 6 shows the result from a typical run to construct an HMTP tree with 100 members. Members join the group one by one within the first minute of simulated time, causing the sharp initial increase in tree cost ratio (y-axis). In the simulations, members run the tree improvement algorithm once every 30 s. By the second minute, tree cost ratio has decreased to a stable point, meaning the tree structure has converged. In most of our simulations, the tree stabilizes and there is no more changes to the tree structure after four to five runs of tree improvement algorithm by each member. Note that even without running the tree improvement algorithm, the tree cost ratio is relatively low at peak (about 1.5 times that of SPST). We attribute this to how the join algorithm itself helps newcomers find a close-by parent. In the simulated scenario, 50 members leave the tree in random order during the fifth minute of simulated time. After the 50 departures, it takes the remaining members less than a minute to settle upon another stable tree. We conclude that a HMTP tree can converge quickly in the face of drastic group membership changes.

We next experimented with various multicast delivery trees connecting 20–500 members to compare their tree costs. For each delivery tree, we randomly pick a member to serve as the source (for SPST and naive unicast) or rendezvous host (for SPGT) or root (for HMTP), and calculate the tree cost of the resulting tree. As expected, SPST and SPGT have a cost ratio of 1, while naive unicast have large cost ratio which also increases significantly with larger group size (Fig. 7). HMTP's cost ratio is slightly greater than 1 and increases very slowly with larger group size. Therefore, in terms of tree cost, HMTP's efficiency is close to that of IP multicast. More analysis of cost ratio results can be found in [45].

Group-shared trees such as SPGT and HMTP incur penalty on end-to-end delay. Fig. 8 shows the probability density function (pdf) of RDP among all pairs of members in an HMTP tree. Most pairs have RDP less than 5, but the worst one reaches 12. Note however, high delay ratio does not necessarily mean large absolute delay. Large absolute delay is reflected in large group

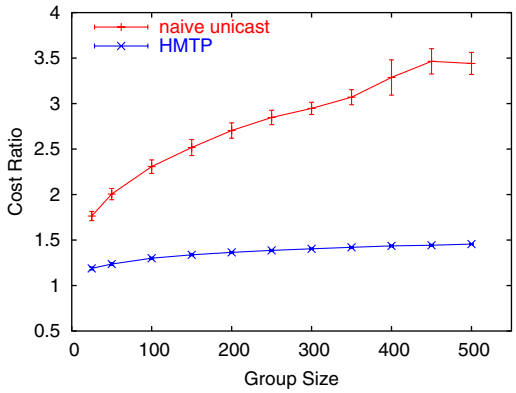

Fig. 7. Cost ratio of HMTP and unicast star.

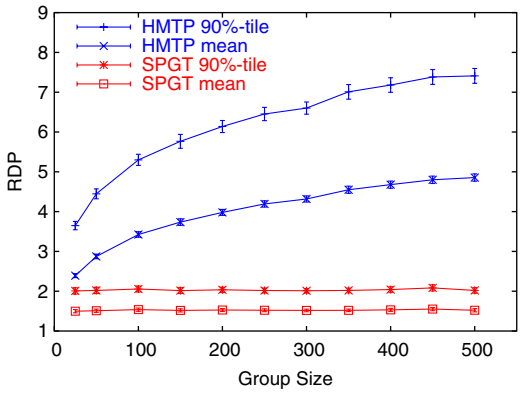

Fig. 9. Tree delay of HMTP and SPGT tree.

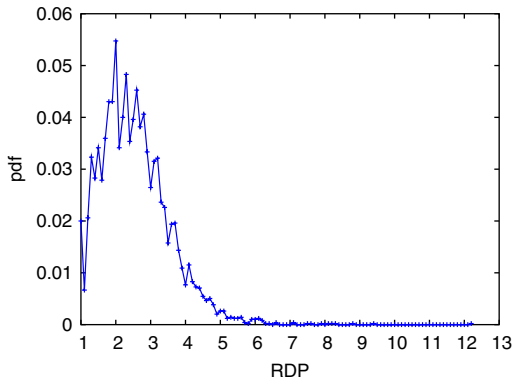

Fig. 8. Distribution of RDP in a HMTP tree of 100 members.

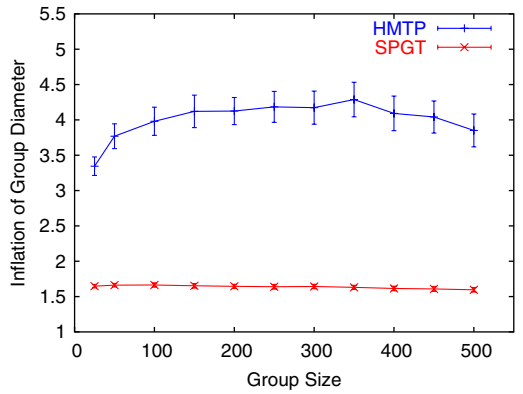

Fig. 10. Inflation of group diameter for HMTP and SPGT tree.

diameter inflation. Fig. 9 shows the average and 90%-tile RDP from our simulations as we increase the group size. The corresponding group diameter inflation is shown in Fig. 10. As delay ratio increases in the former graph, the group diameter inflation stays largely flat in the latter graph. More analysis of RDP results can be found in [45].

Fig. 11 compares the link load of HMTP and naive unicast in a multicast tree connecting a group with 100 members. Though most of the links have a load of 1, naive unicast has a very high load on a few links. The worst case happens at the last hop to the source, where the link load is almost the same as the group size, as expected. If one link is congested because of high link load, all the downstream members see performance deg-

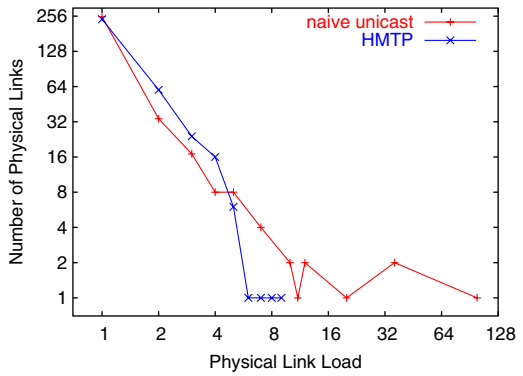

Fig. 11. Link load of a HMTP and naive unicast tree, with 100 members.

radation. HMTP avoids this problem by reducing worst case link load significantly. Under HMTP, an end host can control the load on its last hop link by adjusting the number of neighbors it connects to. We use a maximum degree constraint of eight throughout our simulations. In reality, the degree constraint can be adjusted according to available bandwidth. Smaller degree constraint results in a deeper tree, which may have a higher cost and higher delay, but with lower load on physical links.

In addition to simulations on random topologies, we also run simulations on a snapshot of real Internet topology. To construct the Internet topology, we downloaded one day's (March 1st, 2001) worth of traceroute data from the NLANR site [34]. The traceroutes were conducted among every pair of more than 100 US sites. After removing incomplete measurements, we obtained a topology consisting of 978 routers and 96 hosts. We then ran 1000 simulations on this topology using the same parameters as the simulations we ran on the randomly generated topologies. Comparing the trees constructed by HMTP against that constructed by IP multicast[2] we computed values for our various metrics, averaged over the 1000 runs. Our results are:

- tree cost ratio: 0.99,
- average RDP: 1.76,
- 90%-tile RDP: 2.50,
- group diameter inflation: 2.40, and
- worst link load: 8.

3.3. Improving end-to-end latency

Fig. 12 compares HMTP with two other protocols, Yoid and Narada. It is clear that HMTP has shorter end-to-end latency than Yoid, but longer than Narada. Compared with Yoid, HMTP uses several heuristics to avoid local minima and try to cluster nearby nodes together in tree construction. Therefore HMTP is able to achieve shorter end-to-end latency than Yoid. Narada is a mesh-based protocol. It uses many more overlay links, which can serve as "shortcuts" between nodes, thus it has shorter latency than HMTP. In another work [43], we show that adding some additional overlay links to HMTP trees can result in a mesh structure (TMesh) that has shorter latency than Narada. Generally TMesh uses less overlay links than Narada, but as the group size increases, TMesh's advantage on end-to-end latency becomes more pronounced (Fig. 12). Applying the same TMesh algorithm on a randomly generated tree does not have similar results (Fig. 13). This demonstrates that HMTP can be a good substrate for more sophisticated protocols if there is such a need.

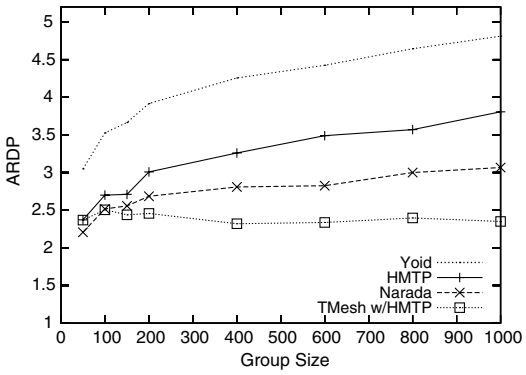

Fig. 12. ARDP performance of various overlays.

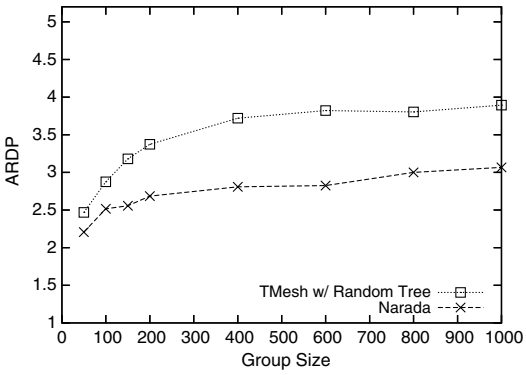

Fig. 13. ARDP of TMesh overlay built on random tree.

[2] IP multicast does not create a minimum spanning tree, which explains the cost ratio below 1 in our result.

4. Host group management protocol (HGMP)

In HMTP or any other end-host multicast protocol, the basic assumption is that every member host is isolated in terms of network multicast connectivity. Considering IP Multicast is available in every Ethernet, many campus networks, enterprise networks, and a few ISPs, the assumption of all hosts are isolated is certainly too pessimistic in the current Internet, not to mention in the future. Taking advantage of the installed base of native IP multicast will not only improve the service performance and scalability, but also encourage further deployment of native IP multicast.

HGMP expands every single node in end-host multicast into a multicast island. On one hand, HGMP must retain the deployability of end-host multicast. It cannot require any administrative configuration in DM election, DM coordination and island management, which would be much easier otherwise. On the other hand, HGMP must also allow the evolution of multicast service, being able to automatically take advantage of any infrastructure support, such as network multicast, dedicated servers, etc., if available.

One important feature we achieved in the design of HGMP is its *protocol independence*: HGMP assumes the existence of an intra-island IP multicast protocol (e.g., DVMRP) and an inter-island multicast protocol (e.g., HMTP), but it does not make any assumption on which particular protocol is in use as long as they provide multicast functionality at network layer and application layer respectively. In practice, network operators choose intra-island multicast protocols, and applications choose inter-island multicast protocols. Since both network multicast and end-host multicast are in constant development and deployment, protocol independence gives HGMP great flexibility to accommodate different network environments and future progresses.

4.1. Single designated member

4.1.1. End-host only

When a host joins a UM group with the group identifier G, it first checks the well-known group directory in its local island for an announcement for G. If no such announcement is present, it assumes that it itself is the first group member in this island and becomes the local DM for G. It then creates three new IP multicast groups, DATA_GROUP, ASSERTION_GROUP and DM_GROUP, in the local island, associates them with G, and announces the mappings to the group directory. The DATA_GROUP, ASSERTION_GROUP and DM_GROUP are not well-known groups like the group directory. Instead, they are dynamically allocated and specific to each UM group G.

The DATA_GROUP is used for transmitting data packets. Applications send to DATA_GROUP their data packets, which will be received by all the group members in the same island, including the DM. The DM then sends these packets out to other islands through tunnels and multicast incoming tunneled packets to DATA_GROUP. Therefore applications still send and receive native IP multicast packets without being concerned about how they are transmitted in the network (Fig. 14).

The ASSERTION_GROUP is used for DM election. The DM of a group G periodically sends ASSERTION messages to G's ASSERTION_GROUP. All members of G must continuously listen to its local ASSERTION_GROUP. When the DM leaves G, it sends a QUIT message to the ASSERTION_GROUP. Remaining members will start an election upon receiving the QUIT message by scheduling sending their own ASSERTION messages after random delays. The first member to send out its ASSERTION message becomes the new DM, and others cancel their message

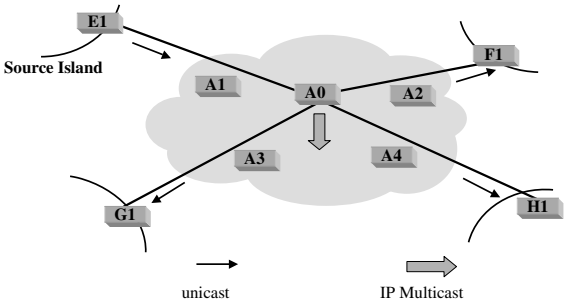

Fig. 14. Single-DM island.

sending. A tie can be resolved by picking the member with the smallest IP address. When the ASSERTION message is continuously missing for a number of periods, a new election will be triggered.

Since the DM is a normal user's computer, usually there is no incentive to sharing its bandwidth and computing power with others when the owner no longer has interest in the multicast group. Therefore when all applications on a local host leave the group, the DM will leave the group too. For some types of groups the DM is likely to change as members join and leave the group continuously. To reduce packet loss during change of DM, the old DM should continue to forward packets after sending its QUIT message for a short period of time. To ensure smooth transition, each ASSERTION message carries information on inter-island routing (e.g., the parent and children nodes on the HMTP tree). With this information, the new DM can quickly establish necessary tunnels to other islands or repair the inter-island multicast tree if the old DM crashes.

One enhancement to the basic DM election criterion is to favor hosts that have more resources. Each host has a priority computed as a function of its resources. This priority is included in the ASSERTION message. A message with higher priority always wins over, or suppresses, a message with lower priority, regardless of the messages' relative sent order. Therefore a host with Ethernet connection can take over the role of DM from a host with only a dial-up connection. However too many priority levels could slow down the election process and also lead to many DM changes as members join and leave the multicast group. Hence we stipulate the use of only a small number of priority levels, e.g., based only on the type of network access technology a host has.

4.1.2. Dedicated server

Using normal hosts as DM is necessary to meet UM's deployability requirement. Nevertheless, when a dedicated server with more computing power and network bandwidth is available, using it as the DM can improve performance and stability of data delivery. While a DM of normal host would leave a group when applications running on it are no longer interested in the group, a DM of a dedicated server can keep forwarding a group's traffic until the last group member in the local area network leaves the group. This reduces the number of DM changes and enhances the stability of the forwarding service.

A dedicated server can be set up to serve certain multicast groups within a service area (e.g., a departmental network). The server is configured with an election priority higher than that of normal hosts. The server periodically sends scoped multicast messages to its service area to query local membership, and normal members send back reports via multicast too, much like the IGMP query/report mechanism in IP multicast. In fact, if the service area is a local area network (LAN) and has a multicast router in it, the server can just passively monitor the IGMP traffic to learn group membership in the network. When there are group members in its service area, the server will participate in the DM election by sending its higher-priority ASSERTION messages, so that it will always supersede normal hosts in becoming a DM. A server ceases its role as DM when there is no longer any host in its service area interested in the group, even if there are still some group members in other network areas. This design decision removes a possible disincentive network operators may have in hosting a UM server.

Backup servers can be configured with a priority higher than normal hosts' but lower than that of the primary server of the same service area. During DM election, backup servers automatically take over if the primary server does not send out an ASSERTION message. If all servers are down, a normal host will be elected as DM according to the election algorithm. Thus the deployment of servers is totally transparent to hosts, except performance improvement experienced by applications.

4.2. Multiple designated members

When there is only one DM in an island, all the group traffic coming into or going out of the island must go through the DM. This leads to two potential problems in large IP multicast islands: longer latency and traffic concentration. For example, in

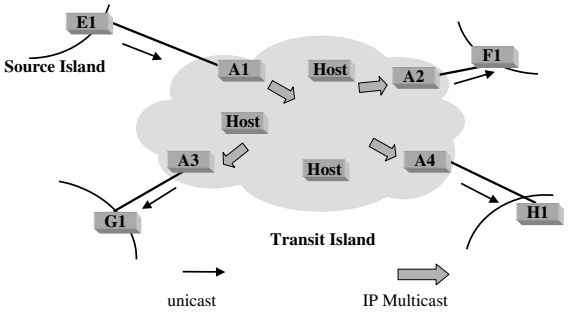

Fig. 15. Multi-DM Island.

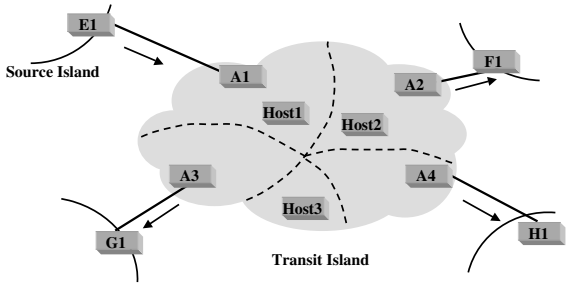

Fig. 16. Election of multiple DMs.

Fig. 14, even if E1 is close to A1, packets from E1 must take a detour via A0 before reaching A1. If A1 is chosen as the DM, longer delay will occur to packets coming from other islands. In general, these two problems become worse for large islands with many members scattering in the islands, which means that the design cannot take full advantage when IP multicast is widely deployed.

A natural solution is to allow multiple DMs per island. In Fig. 15, island A has four DMs. They share the workload of packet forwarding as each maintains one unicast tunnel. Packets coming into or going out of island A now take shorter paths than always having to detour through a single DM. In order to have multiple DMs per island, we need mechanisms to elect multiple DMs, and coordinate them with each other.

4.2.1. Multi-DM election

In the single-DM model, an ASSERTION message travels the full extent of an IP multicast island. Thus the DM's ASSERTION messages can effectively suppress all the other members from becoming a DM. To enable multiple DMs per island, we allow a smaller scope for sending the ASSERTION message. In this case, members outside the assertion scope will not hear the ASSERTION message, hence they will start electing a new DM among themselves, resulting in multiple DMs in the same island, and each DM has an assertion coverage around its network neighborhood. These DMs, however, share the same island-wide data scope, which means data packets multicast by one DM will still reach all group members (see Fig. 16).

The ideal way to set assertion scope is to let every DM dynamically adjust its assertion scope, so that there is no overlap between any two assertion scopes while the entire island is covered by the union of all assertion scopes. However, since IP multicast scoping is either a circular coverage (i.e., generic IP TTL scoping) or administratively configured (TTL threshold and administrative scope), a host cannot define a scope to cover an arbitrary network area, and in many cases overlap of assertion scopes is inevitable. Therefore we choose to tolerate overlap of assertion scopes. The only overhead is redundant ASSERTION messages in the overlapped areas, which should be low volume traffic. There will be no duplication of data packets since HGMP ensures that data packets enter an island only once and only via a single DM (see next subsection). Not requiring perfectly separated scopes enables HGMP to use any multicast scoping scheme available, which is in line with our deployment requirement. Better scoping schemes like administrative scope give HGMP more choices in setting a suitable assertion scope.

How to choose an assertion scope appropriate to a given network environment remains an open issue. Very small assertion scopes may generate an excessive number of DMs, which will complicate the inter-island topology and cause overhead to routing. Appropriate assertion scopes should give rise to a number of DMs adequate to serve the large island, yet not excessive to incur much overhead in inter-island routing. In an ASSERTION message, its initial IP TTL value is carried in the payload. By comparing the initial TTL value and the received TTL value, a receiver can esti-

mate how many hops away a DM is. If a DM sees another DM which is very close by, it can cease to be a DM. Conversely, if a normal member finds that its current DM is too far away, it can become a new DM. In this way, even if a DM is mis-configured with an assertion scope too small or too large, others can offset the effect to maintain an appropriate number of DMs in an island. In the current implementation, DMs use a pre-configured assertion scope corresponding to organization scope. More experiments and practical experiences are needed to refine how to set the assertion scope.

4.2.2. Inter-island routing with multiple DMs

A single-DM island has only one point for packets entering or exiting the island, therefore an island can be abstracted as a single node in inter-island topology. In contrast, a multi-DM island can have multiple entrances and exits for packets. In this case, IP multicast transmission becomes a transit part of the data delivery path, rather than just the last hop as in a single-DM island. The multiple DMs in the same island are connected by IP multicast implicitly, but these DMs are not aware of each other. This *invisible* connectivity breaks the inter-island routing regardless of which end-host multicast protocol is in use.

For example, in Fig. 17, assume island A's DMs, A1 through A4, run an end-host multicast protocol independently and build tunnels to other islands as shown. When a data packet is originated by E1, it will reach both A1 and A3. Not aware of the other's behavior, A1 and A3 will both accept the same packet and multicast it into island A. When the multicast packet reaches A4, it will be forwarded to H1, F1 and back to A2, resulting in more duplicates in island A due to the routing loop.

Therefore, explicit coordination among multiple DMs in the same island is necessary for preventing routing loops and packet duplicates. This coordination is done via the local multicast group DM_GROUP. We designed different coordination schemes for tree-based and mesh-based inter-island routing.

4.2.3. Tree-based inter-island routing

In tree-based end-host multicast protocols, such as HMTP and Yoid, multicast delivery is achieved by simply flooding the shared-tree. In order to keep this working with multi-DM islands, the DMs need to be organized into a "mini-tree" as follows.

All DMs in the same island use the DM_GROUP to dynamically elect one DM as the *Head DM*, while others are *Tail DM*s. The Head DM periodically sends ALIVE messages to DM_GROUP and sends a LEAVE message before it leaves the group. Tail DMs start a round of election upon receiving the LEAVE message or missing ALIVE message continuously. The Head DM runs the inter-island routing protocol as usual to find its parent in other islands. Tail DMs, on the other hand, must always take the Head DM as their parent, as illustrated by the dotted lines in Fig. 18 between Head DM A1 and Tail DMs

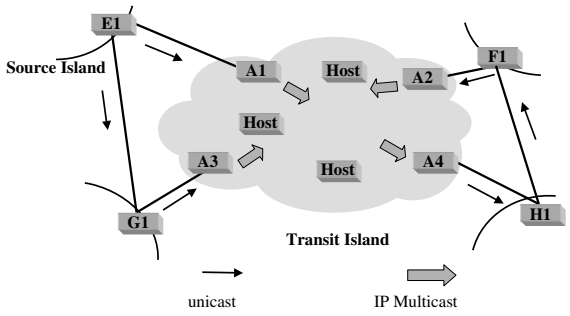

Fig. 17. Routing problems for multi-DM island.

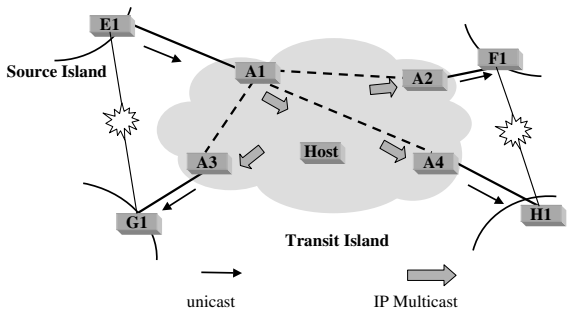

Fig. 18. Multi-DM island with shared-tree inter-island routing.

A2, A3 and A4. From another island's point of view, island A now has only one parent (i.e., E1) and multiple children (i.e., F1, G1, and H1), just as in a single-DM island. When all DMs run the inter-island routing protocol with this restriction, there will be no link E1–G1 or F1–H1 in Fig. 18, because considering the dotted lines they form loops explicitly, which is not allowed by the inter-island routing protocol. Therefore, the tree structure, composed of both unicast tunnels and IP multicast, is preserved through multi-DM islands, which ensures that the same forwarding scheme can be used without routing loop or packet duplicate. Note that the parent–children relationship between Head DM and Tail DMs is only logical. There is no actual unicast tunnels between them, and data packets are still transmitted via IP multicast inside the island.

Initially, the first DM in an island becomes the Head DM. When there are multiple DMs, the election of Head DM takes network latency into account. Every DM runs the inter-island routing protocol independently to determine its potential parent in other islands and measures rtt to the potential parent. The Head DM includes its rtt value in its ALIVE messages. If a Tail DM has a smaller rtt than that advertised by the Head DM, it assumes the role of Head DM by sending its own ALIVE message with the smaller rtt value. The result is that the Head DM is always the DM with the shortest rtt to its extra-island parent. This election criterion favors DMs at island edge over DMs inside the island, thus IP multicast is utilized better as packets are multicast at the earliest point into an island.

Techniques used in single-DM election (Section 4.1) can also be used to improve the stability of packet forwarding service during Head-DM election.

4.2.4. Mesh-based inter-island routing

Mesh-based end-host multicast protocols need to periodically exchange routing updates between neighbor nodes, and apply a specific forwarding rule based on routing table to achieve loop-free multicast delivery. For example, in Narada and TMesh, every node exchanges path vector routing tables with neighbors, and uses reverse path forwarding (RPF)[3] to decide how to forward data packets. Other protocols may use very different routing information (e.g., a node's logical address) and forwarding rule, but the basic behavior is the same: exchange routing updates and apply the forwarding rule.

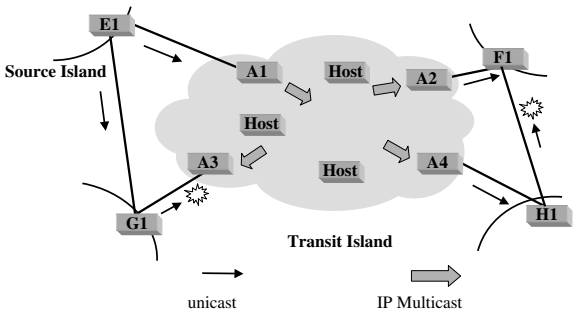

Fig. 19. Multi-DM island with source-tree inter-island routing.

To keep it working in multi-DM islands, in addition to regular routing exchange with neighbors in other islands, DMs also periodically multicast their routing exchanges to the DM_GROUP, so that all DMs in the same island will learn others' connectivity information. Take Fig. 19 as an example, and assume Narada is used as the inter-island routing protocol. After the inter-island routing exchange and intra-island routing exchange, all DMs in island A (i.e., A1, A2, A3 and A4) know that the shortest path from island E to island A is via link E1–A1. Similarly, F1 knows the shortest path from island E goes through A2, and H1 knows the shortest path from island E goes through A4. When E1 sends a packet, the copy going through E1–G1–A3 will be dropped because it fails the RPF checking. Similarly, the copy between H1 and F1 will be dropped too. Now, for the purposes of inter-island routing and packet forwarding, every island can be viewed as a single node, and every DM has complete routing information to apply the forwarding rule cor-

[3] In RPF, a packet originated by source S is forwarded by F to a receiver R only if R uses F as the next hop in R's shortest path to S. The distribution tree generated by RPF is the reverse shortest path tree rooted at the source. This technique is used in some IP multicast routing protocols like DVMRP and PIM-DM.

rectly. More details of the protocol operations are discussed in [46].

4.3. Simulation

We use simulations to evaluate the performance gain of the multi-DM model over the single-DM model. Results presented here are from simulations of 100 islands with 20 group members in each island. The number of routers in an island varies from 50 to 500, but is the same for all islands in the same simulation run. Intra-island IP multicast routing assumed a per-source shortest path tree similar to that used in DVMRP; Inter-island routing uses HMTP. In the single-DM model, the DM is randomly selected from member hosts with uniform probability. In the multi-DM mode, half of the members in an island are randomly selected as DMs, again with uniform probability. Clearly we do not expect half of an island's population to serve as DMs in practice. Since in our simulations DMs that are not very well placed, performance-wise, will not be selected by the end-host multicast protocol to be on the inter-island multicast tree, their existence does not effect the performance metrics studied. It does mean, however, that the performance numbers reported correspond to cases in which we can find well-placed hosts to serve as DMs. This is a topic of our future work.

Fig. 20 shows that as the island size increases, ARDP under the single-DM model case increases

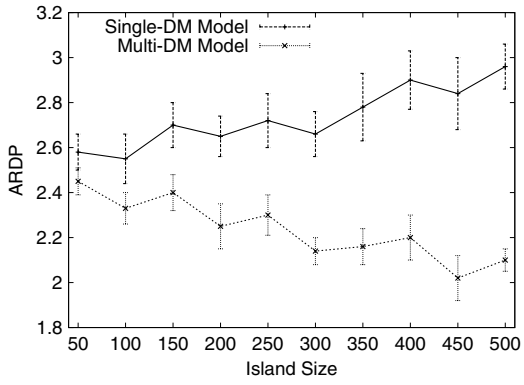

Fig. 20. ARDP vs. island size.

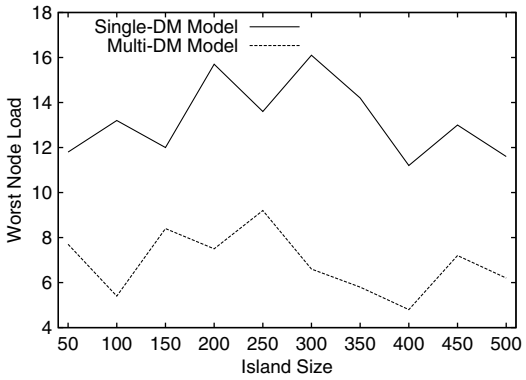

Fig. 21. Worst link load vs. island size.

whereas those under the multi-DM model actually decreases. We attribute this to the increasing significance of intra-island latency as the island size increases. The multi-DM model can factor out large intra-island delays in inter-island latencies. Fig. 21 shows that while having multiple DMs per island clearly reduces the maximum node load, for the scenarios simulated the effect of island size on maximum node load is not apparent.

5. Implementation and deployment

We have implemented HMTP together with the TMesh protocol [43]. The implementation consists of three components: a rendezvous server, an overlay daemon, and an overlay library. The rendezvous server runs on the rendezvous host (UMRP), whose main purpose is to bootstrap newly joining members into the multicast group. The UMRP replies to members' queries with the current root of the HMTP tree and a list of members already in the multicast group. The overlay daemon performs tree construction and maintenance functionality, such as join, improvement, partition recovery, triangle optimization, etc. The overlay library provides a list of APIs (e.g., `join_group`, `send_to_group`, `recv_from_group` and `quit_group`) for applications to utilize our multicast protocol.

By leveraging the overlay library, it is straightforward to build a multicast application. Fig. 22

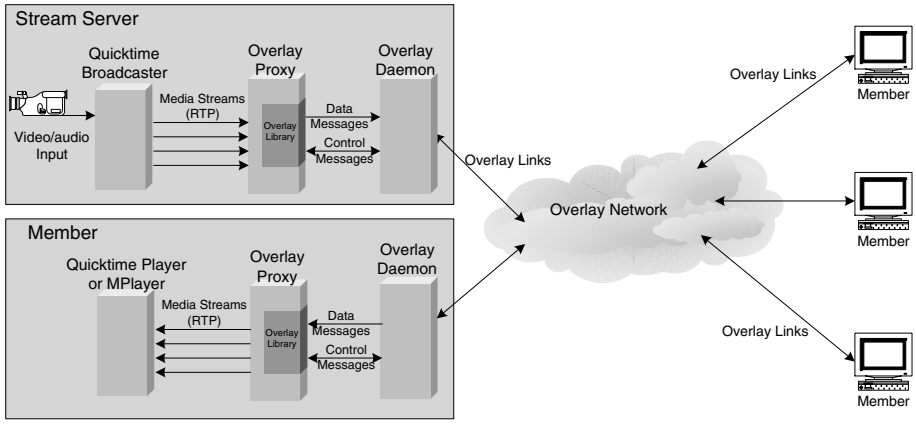

Fig. 22. Framework of TMeshV implementation.

shows the framework of *TMeshV*, a video/audio broadcasting tool we developed to run on our overlay network. (The UMRP is not shown in the framework as it does not participate in the tree overlay.) As shown in the *Stream Server* part of Fig. 22, we employ Quicktime Broadcaster [3] to encode the video/audio input from a camera into RTP/RTCP streams [38]. The overlay proxy, which utilizes our library, captures these RTP messages, wraps them up in TMesh data packets, and sends them to the overlay daemon. The overlay daemon then relays these TMesh data packets to other members in the multicast group. When the overlay daemon of a group member receives these packets, it forwards them to the overlay proxy, which feeds the application-level data to the media player. Since the encode software and media player share the same session description protocol (SDP) [27] setting, our overlay proxies can handle the network transmission part transparently, and there is no need to change either the encode software nor the media players.

We used TMeshV to broadcast the NetGames 2003 and 2004 workshops, the Network Troubleshooting 2004 workshop, and the ACM SIGCOMM 2004 conference live online [5,6]. In this section, we present some details of our implementation in the context of our TMeshV tool, together with statistics of the SIGCOMM 2004 broadcast.

5.1. A simple case

In the design of the HMTP protocol, we consider all members in the multicast group as identical. However, on the Internet, this is not the case. Before delving into the details of how we implement the HMTP protocol on the heterogeneous Internet, we first show a simple and "ideal" case to test the quality of the constructed HMTP tree. We deployed TMeshV on 10 PlanetLab nodes [13] from universities and research labs. We call this an "ideal" case because these nodes share the same operating system environment, are located in well-connected campus or corporate networks, and have fast Internet connections. In the experiment, we set the node degree limit to four. Fig. 23 shows the resulting HMTP tree. Our over-

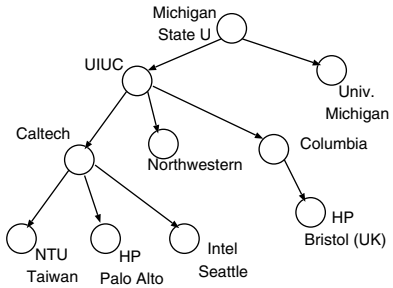

Fig. 23. HMTP tree built on a multicast group of ten PlanetLab nodes.

lay daemons can successfully organize themselves into an efficient tree as expected.

Compared to PlanetLab deployment, deploying TMeshV on the broader Internet is far more complex. We have to consider various network conditions, system configurations, and operating-system-dependent problems. Furthermore, since we run a video/audio broadcast application, we also need to tune some parameters of our protocol to provide better user perceived quality.

5.2. Bandwidth measurement

Due to the high bandwidth requirement of video/audio streaming, a host's available bandwidth is more likely to determine user's perceived video quality. This is particularly the case for end-host multicast where one host's bandwidth is not only used to receive the video but also to relay it to other hosts in the multicast group. Saroiu et al. studied the speed of Internet connections of hosts in the Gnutella [4] peer-to-peer file-sharing network and found that about 70% of the peers had connection speed between 100 Kbps and 10 Mbps [37]. This range of bandwidth indicates that a large number of these hosts use asymmetric Internet connections, such as DSL and cable modem. They may have sufficient bandwidth to receive the video stream, but may not have enough upstream bandwidth to serve content to other peers. Having such hosts act as internal nodes in the multicast tree will affect the video quality perceived by all their downstream peers. Therefore, we need to set a low limit (e.g., zero) on the number of peers these hosts serve.

To check whether a host (H) is behind a slow upstream connection, our overlay daemon conducts a simple bandwidth measurement when H initially joins the group. The daemon first sends a small (64 KB) packet to a randomly selected member of the group. If the time needed to send this packet is above a certain threshold, H's upstream speed is considered slow. Otherwise, the daemon sends a second, larger (256 KB) packet to the same target host. H's upstream bandwidth is estimated based on the time needed to transmit both packets. The estimated bandwidth is reported to the UMRP and is used to determine how many peers H can serve. To overcome inaccurate measurements caused by transient congestion, the daemon repeats this bandwidth measurement periodically at a low frequency. From their domain names and *whois* database information, we estimated that during the broadcast of SIGCOMM 2004, 39.4% of the 221 remote attendees were behind either DSL or a cable modem.

5.3. Firewalls and NAT gateways

Besides network capacity, we also have to deal with another heterogeneity of the Internet—the existence of guarded hosts. A host is *guarded* if it is unable to accept incoming connections from those outside its local area network. This may happen if the host is behind a NAT gateway [17] or a firewall. A host that permits incoming connections as well as outgoing connections is considered *open*.

In the presence of guarded hosts, network reachability is no longer symmetric for every pair of overlay hosts. This complicates our overlay implementation in two ways. An overlay link cannot always be formed between any given pair of overlay nodes. Trying and failing to connect to guarded hosts unnecessarily lengthen new members' join latency. Furthermore, these guarded hosts cannot receive data from the group unless it initiates TCP connections to its peers. We have limited choices on how to transmit data to these guarded hosts.

We studied the prevalence of guarded hosts on the Internet by conducting measurements on two existing peer-to-peer networks, eDonkey [1] and Gnutella. As many as 36% of the 180,000 peers encountered in our experiment were guarded [42]. Clearly, the lack of two-way communication capability is prevalent in the current Internet. Chu et al. reported even higher percentages of guarded hosts in their Internet overlay multicast experiments [11], even as high as 76%. As shown in Table 1, 74.2% out of the 221 remote attendees of SIGCOMM 2004 were on guarded hosts. To support guarded hosts in HMTP/TMesh, we implemented a guarded host detection mechanism and introduced several mechanisms to accommodate them, as described below.

Table 1
Statistics of the broadcast of SIGCOMM 2004

Conference	Broadcast hours	Total number of remote attendees	Attendees on NAT/firewall hosts (percentage)	Maximum concurrent attendees
NetGames 2004	09:25	70	49 (70.0%)	15
SIGCOMM04 Day1	10:01	85	53 (62.4%)	23
SIGCOMM04 Day2	09:59	62	47 (75.8%)	20
SIGCOMM04 Day3	11:01	59	44 (74.6%)	22
Network troubleshooting 2004	08:41	25	18 (72.0%)	12
Total	49:07	221	164 (74.2%)	

Guarded host detection is handled by the UMRP. To identify whether a host H is behind a NAT gateway, the UMRP compares the IP address it gets from the host's TCP connection with the source IP address H sets in its packet header. H is behind a NAT gateway if these two addresses are different. In most cases, hosts behind NAT gateways have known internal IP addresses, such as 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16, or 169.254.0.0/16. We also handled cases where an open host uses the loopback address (127.0.0.1) as its source address. Detecting hosts behind firewalls is not as straightforward as detecting a NAT-ted host. When a new member H joins the group, the UMRP randomly picks three open hosts in the group that are not in H's subnet to connect back to H. If any of these connections succeeds, the host is considered open. To pass through NAT gateways and firewalls, we use TCP instead of UDP to transmit both control messages and the multicast content. Once a guarded host initiates a TCP connection to another host, two-way communications can be established through this TCP connection.

If there are more guarded hosts than the total capacity of open hosts to serve them, new members will not be able to join the multicast group. To alleviate this problem, we seeded the multicast tree with a number of PlanetLab nodes to ensure sufficient number of resource-rich hosts residing on the open network. Next, we give open hosts higher priority in the tree construction process. When a new member H that is an open host tries to join the tree at a potential parent that already has its maximum number of children, if the parent has a child that is a guarded host, it can accept H as its new child and direct its guarded child to switch its parent to H.

The root of the HMTP tree obviously must be on an open host. To ensure a shallow tree, which reduces delivery latency, we further prefer the most resource-rich host as the root. To this end, we instrumented the UMRP with a root-contention mechanism. Hosts are assigned priorities based on their network reachability. The UMRP maintains a list of "trusted" hosts. These hosts have the highest priority. Guarded hosts have the lowest priority. When member H joins the group, if H has a higher priority that of the current root (R), instead of providing H with the current root, the UMRP accepts H as the new root, and instructs R to re-join the tree.

Finally, when possible, we direct hosts behind NAT gateways to connect to existing members that are located in the same subnet. When a host H that is behind a NAT gateway tries to join the tree, the UMRP returns both the root of the tree and a list of all members co-residing behind the same NAT gateway as H. Our daemon running on host H first tries to attach itself to the tree as a child of one of these co-located members. Usually hosts behind the same NAT gateway are open to each other, and also close to each other in terms of latency. Although we have not implemented HGMP in our TMeshV tool, this approach is similar to HGMP in essence. We did observe several hosts behind the same NAT gateway during our broadcast of SIGCOMM 04. Employing IP multicast in such a scenario per HGMP would have been more efficient.

Several mechanisms to accommodate guarded hosts have been proposed in the literature. IP next

layer (IPNL) extends the current Internet architecture by adding a layer above IPv4 to identify and address hosts behind NAT gateways [22]. We did not adopt IPNL in our implementation because it requires modifications to end-host system and the NAT gateways. Guha et al. design a mechanism that allows TCP connections to be established between guarded hosts in different NAT-ted subnets [25]. Their approach requires administrator privileges on end systems to run. Ganjam et al. propose a protocol in which open hosts give preference to guarded hosts in choosing a parent [24]. We did not adopt this protocol because we assume that a large number of guarded hosts will be behind DSL or cable modems and are thus not suitable parents anyway. Instead, we propose a location-based clustering algorithm, *e**, to make efficient use of resources available on open hosts [44]. The cluster centers in *e** are then inter-connected using HMTP.

While hosts behind NAT gateways and firewalls are usually not restricted in the port number to which they can open a TCP connection, some networks now allow outgoing TCP connections only to well-known ports, e.g., SMTP and HTTP ports only. Such restrictions are usually found in corporate or government-controlled networks. The usual method to support hosts behind such restricted networks is to run an HTTP proxy on the Internet that acts as a relay for the application. From Table 3, we see that only 10.4% of the SIGCOMM 2004 attendees were from corporate networks. Anecdotal evidence suggests that we would have seen a higher level of attendance from corporate and government-controlled networks if we had an HTTP proxy for TMeshV.

5.4. Portability issues

TMeshV currently runs on four operating systems: FreeBSD, Linux, MacOS X, and Microsoft Windows. To playback the media content, we leverage the Apple Quicktime player on MacOS X and Windows, and MPlayer [2] on FreeBSD and Linux. On Windows, Quicktime player is a very CPU-intensive application. To display streamed video in 320×240 frame size with 400 Kbps of data, Quicktime player alone consumes over 70% CPU on a 1.13 GHz PentiumIII machine with 512 MB of memory. When the CPU is fully utilized, a client cannot relay data fast enough even with adequate bandwidth. In such cases, the CPU power becomes the bottleneck. Although this kind of bottleneck clients does not appear frequently, it may compromise a large portion of the group if the bottleneck client appears near the top of the multicast tree.

To address this problem, the TMeshV daemon collects the hardware configuration of the Windows hosts from the system Registry and adjusts the node degree limit accordingly. Alternatively, a host can compare its received data rate against a published broadcast rate and switch parent if the received rate is too far below the published rate. Or we can monitor the real-time CPU load information and adjust a host's degree limit dynamically. Similar to the approach we take for hosts with low bandwidth, we assign a low (e.g., zero) node degree to these slow or busy machines so that they would stay at the fringes of the overlay and not slow down other members.

5.5. Statistics from the SIGCOMM 2004 broadcast

For the SIGCOMM 2004 broadcast, our stream server runs on a MacOS X laptop, which is connected to the Internet through a DSL with 1 Mbps upstream bandwidth. For the broadcast media streams, we set the video codec to MPEG4 with resolution 320×280 and the audio codec to MPEG4 32 kHz Stereo. The configured data rate is around 400 Kbps. When there are only small movements in the video, which happened quite often since we pointed our camera at the projector screen, Quicktime Broadcaster may produce streams with lower bit rate than the configured data rate. The broadcast lasted five days: three days for the main conference, one day for NetGames 2004 workshop, and one day for Network Troubleshooting 2004 workshop. The second column in Table 1 shows the durations of our broadcast for each day.

During the SIGCOMM 2004 broadcast, a total of 221 unique hosts joined the broadcast. Of these, 164 (74.2%) were guarded hosts (behind NAT or firewall). The maximum number of concurrent remote attendees was 23, which occurred during the

first day of the main conference. We also had the largest number of total remote attendees (85) in the first day. Since we employed about 26 PlanetLab nodes to accommodate the expected high number of guarded hosts, the maximum group size we encountered during the live broadcast was 49. Fig. 24 shows the number of concurrent remote attendees (not including PlanetLab nodes) for each day of the main conference.

Fig. 25 shows the total viewing time of each remote attendees. We sort the remote attendees in non-increasing order based on their total viewing time during the broadcast. Each remote attendee is then assigned an ID based on its rank. Of the 221 remote attendees, 78 (35.3%) of them watched the broadcast over one hour. Each remote attendee sends an "alive" message to the UMRP about once every 60 seconds. Hosts with IDs above 199 left the group before their first "alive" messages, which explains the flat tail on the "Total viewing time" graph. Of the 221 remote attendees, 136 (61.5%) of them joined the broadcast multiple times. Fig. 25 also shows the average viewing time per visit of each remote attendee, while Fig. 26 shows the number of visits of each remote attendee.

Tables 2 and 3 present the geographic distributions of the remote attendees and the locations they connected from (the locations of remote attendees are obtained from their domain names and their *whois* database information). The majority of the attendees were from the US (79.6%), about 11.3% from Europe, 5.9% from Asia, and 3.2% from Australia. Not surprisingly, the majority of remote attendees (57.0%) are from campus network or research labs. More interestingly, a high percentage (39.4%) were connected through DSL or cable modem.

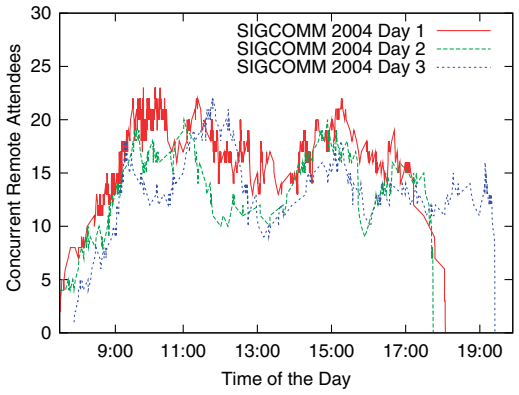

Fig. 24. Number of concurrent remote attendees for SIGCOMM 2004.

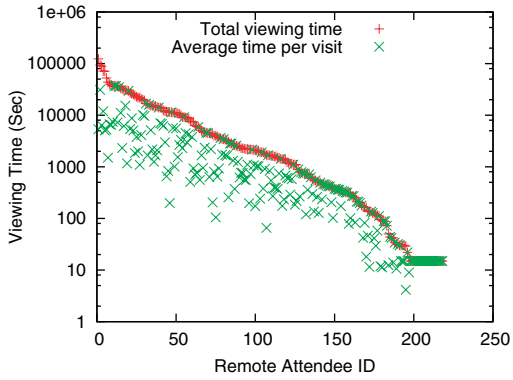

Fig. 25. Viewing time of remote attendees.

Table 2
Geographic distribution of the remote attendees

Country	Remote attendees
USA	176
Japan	9
Australia	7
France	6
Netherlands	6
Canada	4
Belgium	3
Italy	3
China	2
Korea	2
Sweden	2
Germany	1

Table 3
Locations of remote attendees

Location	Number of attendees
Universities and educational institutions	103 (46.6%)
Corporations and research labs	23 (10.4%)
Government (.gov)	5 (2.3%)
Home/office with broadband (DSL/cable modem)	87 (39.4%)
Unknown	3 (1.4%)

5.6. Discussions

As currently implemented, TMeshV provides delivery service without any transport layer control beyond those provided by TCP. Due to the loss-tolerant characteristic of audio/video applications, we simply throw away new data when the TCP socket buffer is full (i.e., when the socket is not ready for writing). One future direction we can pursue is to allow the application to set different levels of priority for its packets. For example, it may be more important to have audio packets delivered at the expense of video packets. When the TCP socket is full, high priority packets can be buffered at the daemon for later transmission. Several issues are immediately obvious with such a scheme. First, if congestion persists, buffered data may become outdated and it may become more useful to transmit new data. Second, the video and audio streams will become de-synchronized and receivers will need to re-synchronize them. Our implementation currently does not attempt to do any re-synchronization of the video and audio streams.

Under our current implementation, when a node switches parent in the multicast tree, data transmission may be interrupted. We did not anticipate how disruptive this interruption is to the audio/video playback. Instead of a few corrupted frames, playback completely freezes for a period of time that is clearly noticeable to the viewer. We will need to implement a smoother hand-off mechanism when switching parent.

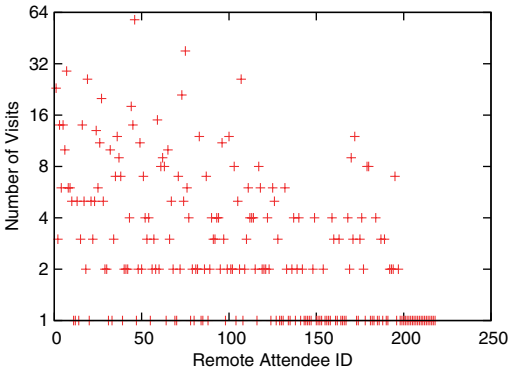

Fig. 26. Number of visits for each remote attendee.

As indicated earlier, HGMP is not currently implemented. Our experience from the SIGCOMM 2004 broadcast indicates that there will be opportunity to take advantage of native IP multicast on local subnets, where IP multicast is more likely to be available. Our future work on TMeshV includes implementing HGMP, adding better QoS support, and testing on a much larger scale.

6. Related work

The MBone [18] was designed to facilitate the deployment of IP multicast. It connects IP multicast islands using tunnels, obviating IP multicast support on intermediate routers between the islands. Nevertheless, setting up the tunnel requires manual configuration and administrative privileges on routers at both ends, which makes connecting to the MBone an expensive proposition. There are several proposals to automate the process of establishing tunnels to MBone. UMTP [20] uses an intermediate source-specific multicast (SSM) router to intercept group join requests sent to the source, and create tunnels on demand. AMT [41] uses dedicated servers (gateways and relays) and IGMP to set up tunnels for SSM. Castgate [32] uses a DNS-like hierarchical, distributed database to support tunnel management. These proposals automate only the setup of static tunnels; none of them support dynamic auto-reconfiguration of the tunnels. Furthermore, they all require operational support from routers or dedicated servers. Since UM makes use of native IP multicast networks, the entire MBone can be viewed as the biggest island in UM.

End-host multicast has minimal deployment barriers because the only requirement is for the users to install a program on their own computers. There are a plethora of end-host multicast protocols, which differ in their target applications and routing algorithms. Judging by the type of multicast overlay they use, these protocols can be categorized as either tree-based protocols or mesh-based protocols. Generally speaking, tree-based protocols maintain less state information at each node and have less overhead, while mesh-based

protocols utilize more overlay links and are able to achieve shorter end-to-end delay. Tree-based protocols include BTP [28], HMTP, NICE [7], TBCP [33], Yoid [21], etc., mesh-based protocols include Narada [12], Hypercast [31], Gossamer [9], TMesh [43], etc. UM uses end-host multicast protocols for inter-island routing. Due to HGMP's protocol independence, applications can choose the end-host multicast protocol that fits application needs the best.

Since end-hosts are less stable and less trustworthy, there are many proposals for building multicast server overlays, including Scattercast [9], Overcast [30], AMcast [39], OMNI [8], Broadcast Federation [10], etc. In order to provide multicast service, an application service provider places dedicated servers all over the Internet, runs an end-host multicast protocol among these servers, and connects end users by unicast. The design issues include server placement, and making routing decisions with the knowledge about server locations, traffic load and bandwidth. Multicast server overlays are expected to provide better multicast service than pure end-host multicast, but its deployability is compromised because it requires deploying a large number of servers by an application service provider. UM does not require dedicated servers, but both HMTP and HGMP can take advantage of them when they are available.

7. Conclusion

In this work, we designed and implemented the universal multicast (UM) framework to provide ubiquitous multicast delivery service on the Internet. As a basic service common to most group communication applications, multicast should be implemented as part of the network infrastructure for the sake of performance and scalability. From the deployment point of view, however, the path of least resistance evolves from network edges towards the core. UM solves this dilemma by having a flexible design that provides applications multicast delivery immediately, and allows infrastructure support to spread gradually at the same time. We designed HMTP for inter-island routing, HGMP for intra-island management, implemented a prototype, and conducted live conference broadcasting. In deploying our prototype implementation, we encountered and addressed several unexpected stumbling blocks with regards to guarded hosts and hosts with asymmetric bandwidth. To the best of our knowledge, this is the first effort to integrate both end-host multicast and native IP multicast for ubiquitous multicast delivery on the Internet that also takes into account the realities of security barriers on the Internet. Our framework is not limited to a particular routing protocol: HMTP can be the substrate for more sophisticated protocols, and HGMP can be integrated with other routing protocols.

References

[1] Edonkey Network. Available from: <http://www.edonkey2000.com>.
[2] Mplayer. Available from: <http://www.mplayerhq.hu/>.
[3] Quicktime Broadcaster. Available from: <http://www.apple.com/quicktime/products/broadcaster/>.
[4] Wellcome to gnutella. Available from: <http://gnutella.wego.com>.
[5] Netgames 2003. Second Workshop on Network and System Support for Games. Available from: <http://confman.eecs.umich.edu/netgames2003>.
[6] SIGCOMM 2004. ACM SIGCOMM 2004 Conference. Available from: <http://www.acm.org/sigcomm/sigcomm2004/>.
[7] S. Banerjee, B. Bhattacharjee, C. Kommareddy, Scalable application layer multicast, in: Proceedings of ACM SIGCOMM, September 2002.
[8] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, S. Khuller, Construction of an efficient overlay multicast infrastructure for real-time applications, in: Proceedings of IEEE INFOCOM, April 2003.
[9] Y. Chawathe, Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service, Ph.D. Thesis, University of California, Berkeley, December 2000.
[10] Y. Chawathe, M. Seshadri, Broadcast federation: an application-layer broadcast internetwork, In Proc. of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), May 2002.
[11] Y. Chu, A. Ganjan, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, H. Zhang, Early experience with an Internet broadcast system based on overlay multicast, in: USENIX Annual Technical Conference, 2004.
[12] Y. Chu, S.G. Rao, H. Zhang, A case for end system multicast, in: Proceedings of ACM SIGMETRICS, June 2000, pp. 1–12.

[13] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, M. Bowman, Planetlab: an overlay testbed for broad-coverage services, ACM Computer Communication Review 33 (3) (2003) 3–12.
[14] L.H.M.K. Costa, S. Fdida, O.C.M.B. Duarte, Hop by hop multicast routing protocol, in: Proceedings of ACM SIGCOMM, September 2001.
[15] F. Dabek, R. Cox, F. Kaashoek, R. Morris, Vivaldi: a decentralized network coordinate system, in: ACM SIGCOMM, 2004.
[16] S. Deering, D. Cheriton, Multicast routing in datagram internetworks and extended LANs, ACM Transactions on Computer Systems 8 (2) (1990) 85–110.
[17] K. Egevang, P. Francis, The IP Network Address Translator (NAT), 1994, RFC-1631.
[18] H. Eriksson, MBONE: the multicast backbone, Communications of the ACM (August) (1994).
[19] A. Fei, J. Cui, M. Gerla, and M. Faloutsos, Aggregated multicast: an approach to reduce multicast state, in: Proceedings of 6th Global Internet Symposium (GI2001) in Conjunction with Globecom, November 2001.
[20] R. Finlayson, R. Perlman, D. Rajwan, Accelerating the deployment of multicast using automatic tunneling. Internet Draft, IETF, February 2001.
[21] P. Francis, Yoid: your own internet distribution. Available from: <http://www.isi.edu/div7/yoid/>, March 2001.
[22] P. Francis, R. Gummadi, IPNL: A NAT-Extened Internet Architecture, in: Proceedings of ACM SIGCOMM'01, 2001.
[23] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, L. Zhang, IDMaps: a global internet host distance estimation service, ACM/IEEE Transactions on Networking (October) (2001).
[24] A. Ganjam, H. Zhang, Connectivity restrictions in overlay multicast, in: Proceedings of NOSSDAV, June 2004.
[25] S. Guha, Y. Takeda, P. Francis, NUTSS: A SIP-based approach to UDP and TCP network connectivity, Future Directions in Network Architecture Workshop, August 2004.
[26] M. Handley, Session directories and scalable Internet multicast address allocation, ACM Computer Communication Review 28 (4) (1998) 105–116.
[27] M. Handley, V. Jacobson, SDP: Session Description Protocol, 1998. RFC-2327.
[28] D. Helder, S. Jamin, End-host multicast communication using switch-trees protocols, in: Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC), May 2002.
[29] H. Holbrook, B. Cain, Source-specific multicast for IP. Internet Draft, IETF, November 2000.
[30] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, J. O'Toole, Overcast: reliable multicasting with an overlay network, in: Proceedings of the Symposium on Operating Systems Design and Implementation, October 2000.
[31] J. Liebeherr, T. Beam, HyperCast: a protocol for maintaining multicast group members in a logical hypercube topology, in: Networked Group Communication, 1999, pp. 72–89.
[32] P. Liefooghe, CastGate: an auto-tunneling architecture for IP multicast, Internet Draft, IETF, November 2001.
[33] L. Mathy, R. Canonico, D. Hutchison, An overlay tree building control protocol, in: Proceedings of the International Workshop on Networked Group Communication (NGC), May 2001.
[34] National Laboratory of Applied Network Research, NSF Cooperative Agreement No. ANI-9807479, NLANR active measurement project. Available from: <http://watt.nlanr.net/>.
[35] T.S.E. Ng, H. Zhang, Predicting Internet network distance with coordinates-based approaches, in: IEEE INFOCOM, 2002.
[36] R. Perlman, C. Lee, T. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, M. Green, Simple multicast: a design for simple, low-overhead multicast, Internet Draft, IETF, March 1999.
[37] S. Saroiu, P.K. Gummadi, S.D. Gribble. A measurement study of peer-to-peer file sharing systems, in: MMCN' 02, January 2002. Available from: <http://www.cs.washington.edu/homes/tzoompy/publications/mmcn/2002/abstract.html>.
[38] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, 1996, RFC-1889.
[39] S. Shi, J. Turner, Routing in overlay multicast networks, in: Proceedings of IEEE INFOCOM, June 2002.
[40] I. Stoica, T.S.E. Ng, H. Zhang, REUNITE: a recursive unicast approach to multicast, in: Proceedings of IEEE INFOCOM 2000, March 2000.
[41] D. Thaler, M. Talwar, L. Vicisano, D. Ooms, IPv4 automatic multicast without explicit tunnels (AMT), Internet Draft, IETF, February 2001.
[42] W. Wang, H. Chang, A. Zeitoun, S. Jamin, Characterizing guarded hosts in peer-to-peer file sharing systems, IEEE GLOBECOM Global Internet and Next Generation Networks Symposim November. (2004).
[43] W. Wang, D. Helder, S. Jamin, L. Zhang, Overlay optimizations for end-host multicast. in: Proceedings of the Int'l Workshop on Networked Group Communication (NGC), October 2002.
[44] W. Wang, C. Jin, S. Jamin, Network Overlay Construction under Limited End-to-End Addressability, Technical Report CSE-TR-489-04, EECS Department, University of Michigan, 2004.
[45] B. Zhang, S. Jamin, L. Zhang, Host Multicast: a framework for delivering multicast to end users, in: Proceedings of IEEE INFOCOM, June 2002.
[46] B. Zhang, S. Jamin, L. Zhang, Universal IP multicast delivery, in: Proceedings of the International Workshop on Networked Group Communication (NGC), October 2002.

Beichuan Zhang is an Assistant Professor in the Department of Computer Science at the University of Arizona. He received his Ph.D. in Computer Science from the University of California,

Los Angeles in 2003, and spent a year at USC/ISI as a Postdoctoral research associate. His research interests include Internet routing, overlay networks, multicast, network measurement and performance evaluation.

Wenjie Wang is a Ph.D. candidate in Electronic Engineering and Computer Science Department at the University of Michigan, Ann Arbor. His areas of research interest include end-host multicast, peer-to-peer system, DDoS resilient overlays, network protocol and security, video/audio streaming. Previously, he got his B.S. degree in the Department of Computer Science and Technology of Peking University, and he received his M.S. degree in EECS department of the University of Michigan.

Sugih Jamin is an Associate Professor in the Department of Electrical Engineering and Computer Science at the University of Michigan. He received his Ph.D. in Computer Science from the University of Southern California, Los Angeles in 1996 for his work on measurement-based admission control algorithms. He spent parts of 1992 and 1993 at the Xerox Palo Alto Research Center, was a Visiting Scholar at the University of Cambridge for part of 2002, and a Visiting Associate Professor at the University of Tokyo for part of 2003. He received the ACM SIGCOMM Best Student Paper Award in 1995, the National Science Foundation (NSF) CAREER Award in 1998, the Presidential Early Career Award for Scientists and Engineers (PECASE) in 1999, and the Alfred P. Sloan Research Fellowship in 2001.

Daniel Massey is an assistant professor at Computer Science Department of Colorado State University and is currently the principal investigator on DARPA and NSF funded research projects investigating techniques for improving the Internet routing infrastructures. He received his doctorate from UCLA and is a member of the IEEE, IEEE Communications Society, and IEEE Computer Society. His research interests include fault-tolerance and security for large scale network infrastructures.

Lixia Zhang is a professor in computer science department at UCLA. She received her Ph.D. in computer science from MIT and was on the research staff at the Xerox Palo Alto Research Center before joining UCLA in 1995. In the past she has served as vice chair of ACM SIGCOMM and Co-Chair of IEEE Communication Society Internet Technical Committee. She is currently a member of the Internet Architecture Board. Her research interest includes resiliency for large scale distributed systems such as Internet routing infrastructure and the global DNS system.