Borg: A Hybrid Protocol for Scalable Application-Level Multicast in Peer-to-Peer Networks

Rongmei Zhang and Y. Charlie Hu
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
{rongmei, ychu}@purdue.edu

ABSTRACT

Multicast avoids sending repeated packets over the same network links and thus offers the promise of supporting multimedia streaming over wide-area networks. Previously, two opposite multicast schemes – forward-path forwarding and reverse-path forwarding – have been proposed on top of structured peer-to-peer (p2p) overlay networks. This paper presents Borg, a new scalable application-level multicast system built on top of p2p overlay networks. Borg is a hybrid protocol that exploits the asymmetry in p2p routing and leverages the reverse-path multicast scheme for its low link stress on the physical networks. Borg has been implemented on top of Pastry, a generic, structured p2p routing substrate. Simulation results based on a realistic network topology model shows that Borg induces significantly lower routing delay penalty than both forward-path and reverse-path multicast schemes while retaining the low link stress of the reverse-path multicast scheme.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.4 [**Computer Systems Organization**]: Performance of Systems

General Terms

Algorithms, Design, Performance, Experimentation

Keywords

Group communication, Application-level multicast, Peer-to-peer, Scalability

1. INTRODUCTION

Multicast avoids sending repeated packets over the same network links and thus offers the promise of supporting large-scale distributed applications such as subscribe-publish services and multimedia streaming over wide-area networks.

While IP multicast was proposed over a decade ago [12], the use of multicast has so far been limited due to its slow deployment in the Internet. In the meantime, application-level multicast has gained popularity. Numerous protocols have been proposed and systems been built [1, 14, 2, 16, 13]. However, the ability to scale up to thousands of nodes remains a challenge.

Recent developments of self-organizing and decentralized peer-to-peer (p2p) overlay networks ([19, 24, 22, 28]) point to a new paradigm for building distributed applications. Each of these overlays implements a scalable, fault-tolerant distributed hash table, by which any data object can be located within a bounded number of routing hops. In addition, these systems exploit proximity in the underlying Internet topology in performing object location and routing. Multicast can be built on top of such p2p overlays to leverage the inherent scalability and fault tolerance of these systems.

Several multicast schemes have been proposed on top of p2p overlay networks. In particular, Scribe [6] and Bayeux [29] are built on top of two similar p2p substrates, Pastry [22] and Tapestry [28]. Both Pastry and Tapestry are based on prefix-routing, and use the proximity neighbor selection mechanism [4] to exploit proximity in the underlying physical network. However, Scribe and Bayeux use opposite schemes in building multicast trees: Scribe uses reverse-path forwarding, whereas Bayeux uses forward-path forwarding. Both schemes offer comparable routing delay characteristics, but Scribe induces lower link stress than Bayeux. This is due to the use of many short links as a result of the reverse-path construction.

In this paper, we propose Borg, a hybrid multicast scheme in p2p networks. Borg is motivated by the asymmetry in routing in structured p2p networks – the overlay path taken in routing a message from node A to node B is likely to be distinct and therefore has a different routing delay from the path taken in routing a message from node B to node A. Borg exploits this asymmetry by building the upper part of a multicast tree using a hybrid of forward-path forwarding and reverse-path forwarding and leverages the reverse-path multicast scheme for its low link stress by building the lower part of the multicast tree using reverse-path forwarding. The boundary nodes of the upper and lower levels are defined by the nodes' distance from the root in terms of the number of overlay hops. Simulation results show that setting the boundary to be half of the average number of

NOSSDAV'03, June 1–3, 2003, Monterey, California, USA.

routing hops in routing a random message gives optimal hybrid multicast trees.

The rest of the paper is organized as follows. Section 2 gives an overview of Pastry and Tapestry and the forward-path and reverse-path multicast systems built on top of them. Section 3 motivates the proposed hybrid multicast protocol, and Section 4 describes the design of Borg. In Section 5, we evaluate the performance of Borg by comparing it with Scribe and Bayeux. We then discuss related work in Section 6 and conclude in Section 7.

2. BACKGROUND

We give a brief review of Pastry routing and Scribe, a reverse-path forwarding multicast system built on top of Pastry, as well as a forward-path multicast system, Bayeux, and the Tapestry p2p network on which Bayeux is built.

2.1 Pastry and Tapestry

We highlight the prefix-routing aspects of Pastry and Tapestry and discuss their implications on the delay characteristics of routing hops.

2.1.1 Pastry

Each Pastry node has a unique, uniform randomly assigned *nodeId* in a circular 128-bit identifier space. Given a 128-bit key, Pastry routes the associated message towards the live node whose nodeId is numerically closest to the key.

For the purpose of routing, nodeIds and keys are thought of as a sequence of digits in base 2^b (b is a configuration parameter with typical value 4). A node's routing table is organized into $128/b$ rows and 2^b columns. The 2^b entries in row n of the routing table contain the IP addresses of nodes whose nodeIds share the first n digits with the present node's nodeId; the $(n+1)$th nodeId digit of the node in column m of row n equals m. A routing table entry is left empty if no node with the appropriate nodeId prefix is known. Each node also maintains a *leaf set*, consisting l nodes with nodeIds that are numerically closest to the present node's nodeId, with $l/2$ larger and $l/2$ smaller nodeIds than the current node's id. The leaf set ensures reliable message delivery and is used to store replicas of application objects.

Pastry performs prefix routing. At each routing step, a node seeks to forward the message to a node whose nodeId shares with the key a prefix that is at least one digit (or b bits) longer than the current node's shared prefix. If no such node is found in the routing table, the message is forwarded to a node whose nodeId shares a prefix with the key as long as the current node, but is numerically closer to the key than the present node's id. Experiments and analysis [22, 5] show that the expected number of forwarding hops is slightly below $\lceil log_{2^b} N \rceil$.

2.1.2 Tapestry

Tapestry is very similar to Pastry in that they both use prefix-based routing but differs in its approach to mapping keys to nodes in the sparsely populated id space, and in how it manages replication. In Tapestry, there is no leaf set and neighboring nodes in the name space are not aware of each other. When a node's routing table does not have an entry for a node that matches a key's nth digit, the message is forwarded to the node with the next higher value in the nth digit, modulo 2^b, found in the routing table. This procedure, called *surrogate routing*, maps keys to a unique live node if the routing tables are consistent.

2.1.3 Delay Characteristics of Routing Hops

Both Pastry and Tapestry perform topology-aware routing via *proximity neighbor selection* [4], a mechanism that chooses each routing table entry to refer to a nearby node in the proximity space, among all candidate nodes for that routing table entry. Since Tapestry and Pastry are prefix-based, the upper levels of the routing table allow a large number of candidate nodes, with lower levels having exponentially fewer and fewer candidate nodes. As a result, when candidate nodes are randomly located in the proximity space, as assumed in many network topology models, the expected delay of the first hop is very low, it increases exponentially with each hop, and the delay of the final hop dominates. These delay characteristics of routing hops have direct implications to the shape of the multicast tree built on top of the p2p networks, as explained below.

2.2 Multicast in Peer-to-Peer Networks

We briefly review multicast systems built on top of Pastry and Tapestry.

2.2.1 Scribe

Scribe uses *reverse-path forwarding* [11]. A Scribe multicast tree is formed by the union of the paths from receivers to the root. In the following, we focus on the process of nodes joining the multicast tree.

To join a group, a node routes a JOIN message using Pastry with the destination key set to the group's groupId. This message is routed towards the root, the node whose nodeId is closest to the groupId. Each node along the route checks whether it is either subscribed to the group or is a forwarder for that group. If it is, it registers the source node as its child in the multicast tree and stops routing the message any further. Otherwise, this node creates an entry for the group, adds the source node as a child and then attempts to join the group using the same algorithm. The properties of Pastry routes ensure that this mechanism produces a tree.

Bottleneck remover In the reverse-path forwarding scheme, the fan-out of individual internal nodes in the resultant multicast tree can be potentially unbounded. To limit the fanout of a node and thus the stress on that node during data dissemination, an algorithm called the bottleneck remover was proposed in [6]. When a node detects that it is overloaded, it chooses the multicast group which consumes the most resource, e.g., the fanout from which is the highest. The farthest child node in the correspondent multicast tree is then chosen for offloading. The new parent for the offloaded node is chosen from all its previous siblings by comparing the overall distance to the current parent through the new parent. The node can attach itself to the new parent through which the distance to the old parent is the smallest.

2.2.2 Bayeux

While Tapestry routing is very similar to Pastry, Bayeux has a fundamental difference from Scribe in that it uses *forward-path forwarding* to build the multicast tree. A Bayeux multicast tree is formed by the union of the paths from the root down to the receivers. Bayeux uses Tapestry for group management and data dissemination.

To join a group, a node routes a JOIN message using

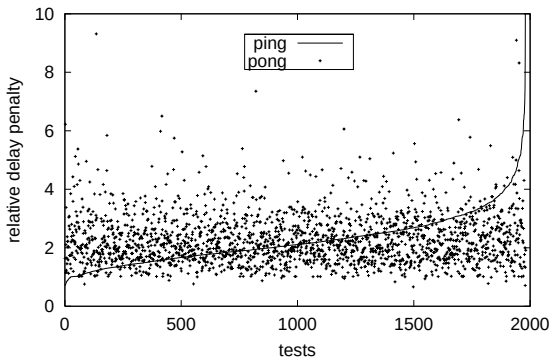

Figure 1: Asymmetric routing delay in Pastry routing. Each test picks a random source node X and a random message key Y. Ping routes message Y to its destination node Y', and pong routes a message from node Y' back to node X with message key X. The tests are sorted according to the relative delay penalty of pings.

Tapestry with the destination key set to the group's groupId. The JOIN message is routed to the root which has the nodeId closest to the groupId. The root then sends back a TREE message towards the new member, which sets up the forwarding state at intermediate nodes. Tapestry routing also ensures that this mechanism produces a tree. Note that in both Tapestry and Pastry routing, the reverse-path from the joining node to the root and the forward-path in the other direction might be different, due to the asymmetric nature of prefix-based routing.

2.3 Comparison

As discussed in [6], while both forward-path multicast and reverse-path multicast exploit network proximity in a similar manner and thus have similar RDPs, forward-path multicast induces a higher link stress than reverse-path multicast. This is because a forward-path multicast tree built with Pastry or Tapestry consists of longer and longer edges in moving from the root towards the tree leaves, while a reverse-path multicast consists of shorter and shorter edges in moving from the root towards the tree leaves. As a result, messages traverse many longer overlay links in forward-path multicast, but only a few long links near the root in reverse-path multicast.

3. MOTIVATION

This section describes the motivation behind the hybrid multicast protocol. Routing in structured p2p networks, including CAN, Chord, Pastry, and Tapestry, is asymmetric in that the overlay path taken in routing a message from node A to node B is likely to be distinct and therefore has a different delay from the path taken in routing a message from node B to node A. This is either due to the asymmetry in the routing table construction, as in Chord, Pastry, and Tapestry, or due to the asymmetry in the underlying physical links, as in the CAN which selects low-latency neighbor nodes during dimensional routing. In this paper, we focus on the asymmetry in the overlay networks.

As an example, we measured the extent of asymmetry in Pastry routing between 2000 randomly chosen pairs of nodes. The tests were performed on a network topology with 1050 routers (1000 are stub routers) generated using the Georgia Tech transit-stub model [26]. We randomly attached 64,000 end nodes to the 1000 stub routers. The routing policy weights generated by the Georgia Tech random graph generator were used to perform IP unicast routing in the IP network. Figure 1 plots the routing delay penalty (RDP) for the 2000 tests. RDP is defined as the delay going through the overlay hops divided by the delay if the message is sent following the shortest path in the underlying network. Figure 1 shows that the asymmetry is significant: the RDPs between two nodes are almost never the same; 48.50% of tests result in larger RDP in ping than in pong; 51.45% of the tests result in larger RDP in pong than in ping; and over 68% of the tests experience larger than 20% difference in ping and pong delays and over 82% experience larger than 10% difference.

The above asymmetry immediately suggests that for each individual subscriber in a multicast group, with about 50% of chance the reverse-path scheme will result in a shorter multicast path, and with about 50% of chance the forward-path scheme will result in a shorter multicast path. This suggests that if routing delay is the only performance metric, a hybrid scheme that simply chooses the shorter path out of the reverse path and the forward path between each subscriber and the multicast root will construct a better multicast tree than one from using either the forward-path scheme or the reverse-path scheme alone. This simple scheme, however, may incur a high link stress on the underlying physical network, since the multicast tree contains both forward paths and reverse paths, and forward paths in a multicast tree may incur higher link stress than reverse paths, as explained in Section 2.3.

The asymmetry in p2p routing and the link stress characteristics of forward-path and reverse-path multicast schemes discussed above motivate a hybrid multicast scheme that is hybrid in two aspects: First, the multicast tree is constructed as a subtree at the top and many subtrees at the bottom, with the roots of the bottom subtrees coincide with the leaves of the top subtree. Second, the bottom subtrees are built using the reverse-path scheme for low link stress, and the top subtree is constructed by exploiting the overlay routing asymmetry for reduced routing delay, i.e., by always using the shorter paths out of the forward paths and the reverse paths.

4. BORG: A HYBRID MULTICAST

Borg is a new scalable application-level multicast built on top of Pastry. Like Scribe or Bayeux, it builds a multicast tree per multicast group on top of a Pastry (or Tapestry) overlay, and relies on Pastry's routing substrate to optimize the routes from the root to each group member based on some network metric (e.g. latency). Unlike Scribe and Bayeux, it builds the upper part of a multicast tree using a hybrid of forward-path forwarding and reverse-path forwarding and the lower part using reverse-path forwarding. The boundary that separates the upper and lower part of the tree is defined by a configuration parameter δ, as explained in the multicast operations of Borg below.

Group Creation Each multicast group has a key called the *groupId*. To create a group, a Borg node asks Pastry to route a CREATE message using the groupId as the key. The destination node to which Pastry routes the message to becomes the root of the tree and is ready to accept node

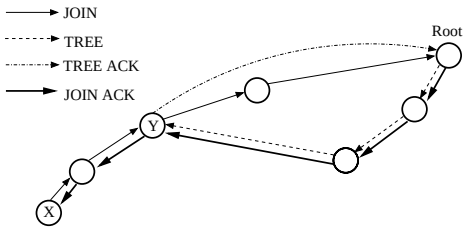

Figure 2: Node X joining in Borg, $\delta = 2$. The forward path from the root to node Y is shorter than the reverse path from node Y to the root.

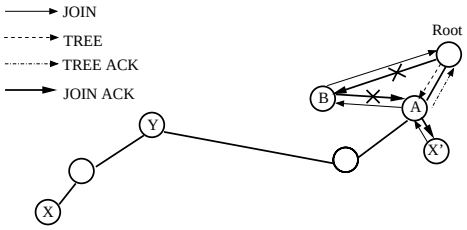

Figure 3: Node X' joining in Borg, $\delta = 2$. The reverse path from the root to node A by way of node B is shorter than the forward path from the root to node A. As a result, new branches from the root to node B and from node B to node A are first added to the multicast tree. Node A notifies node B that it already has a parent node, and node B prunes the branch from itself to node A just added. Similarly, node B in turn notifies the root that the branch from the root is unnecessary.

joining and leaving requests.

Node Joining The node joining process is shown in Figure 2. A node X joins a multicast group by sending a JOIN message addressed to the multicast groupId. This message is routed towards the root. The JOIN message records every forwarding node and the delay of each overlay forwarding hop on its path to the root. The message is forwarded on until it is received at the root or at an intermediate node which can intercept the message. A node A can intercept JOIN messages if it is already on the multicast tree and is at least δ hops away from the root. The joining process guarantees that every on-tree node also knows its own distance from the root, in terms of on-tree hops. Node A takes the previous node on the forwarding path as its child on the multicast tree and sends back a JOIN ACK message. If a node is not on the multicast tree or it is within δ hops from the root, it simply appends itself together with the delay measurement to the previous hop to the JOIN message and forwards it on.

When a JOIN message is received at the root, the root examines the reverse forwarding path from the subscriber embedded in the message and takes the forwarding node that is δ hops away from itself, e.g., node Y. If there are less than δ forwarding nodes on the reverse forwarding path, the first node (the source node of the JOIN message, i.e., the new node X itself) on the path is selected. The root then sends a TREE message towards node Y using the p2p routing. A TREE message in Borg is similar to the TREE message in Bayeux and it discovers the forward path to node Y. Each forwarding node and the delay of each forwarding hop are also recorded in the TREE message. After the TREE message arrives at the destination node Y, the forward path discovered by the TREE message is sent back directly to the root in a TREE ACK message. The root then compares the total delay to node Y by way of the reverse path and the forward path, and chooses the shorter path to construct an on-tree path to node Y. Specifically, the root takes the previous hop on the chosen path as its child and sends back a JOIN ACK message.

A JOIN ACK message is propagated back to the new node X following the forwarding path discovered by the JOIN and/or TREE message. If the JOIN ACK message is originated from an intermediate on-tree node, it simply follows the reverse of the forwarding path that the JOIN message has traveled. Otherwise, a JOIN ACK message from the root may follow a hybrid path consisting of a forward sub-path to the node Y and a reverse sub-path from node Y to the subscriber node X. At each hop on the path of a JOIN ACK message, the node adds the next hop as its child in the multicast tree and also learns about its parent on the tree (the previous hop) and its own distance (on-tree hops) from the root at the same time.

Every on-tree node keeps track of its upstream (parent) node and downstream (child) nodes in the multicast tree. A subtle situation occurs when an on-tree node A receives a JOIN ACK message from an upstream node B different from its current parent node, for example, as the result of a new node X' joining the multicast tree, as shown in Figure 3. In this case, node A notifies node B that it already has an on-tree parent, and node B removes node A from its children table. Node B in turn notifies its own upstream node if its children table becomes empty after node A is removed. As a result, node A remains attached to its current parent node, and the tree structure is preserved.

The optimal δ value which defines the boundary between the upper and lower half of the multicast tree is experimentally determined to be $\lceil \frac{2^b-1}{2^b} \log N/2 \rceil$, as discussed in Section 5.

Node Leaving A node can gracefully leave the multicast group when it is neither a receiver nor a forwarder. It leaves by sending a LEAVE message to its parent in the multicast tree. After the parent removes the node from its list of child nodes, it checks if it itself needs to leave the multicast group. This process can be recursive and a node leaving a lower reverse-path subtree may cause another node leaving the upper hybrid subtree.

Data Dissemination Senders (publishers) send data messages to the root of a multicast group, using the *groupId* as the key. The data messages are then forwarded down the multicast tree.

Reliability Like Scribe and Bayeux, Borg provides best-effort delivery of messages. Like Scribe, it uses Pastry to repair the multicast tree when a forwarding node fails. A forwarding node on the multicast tree sends a HEARTBEAT message to each of its child nodes periodically. A child node (e.g. node X) assumes that the parent node has failed if it fails to receive HEARTBEAT messages. When a parent node failure is detected, node X tries to re-join the multicast tree by sending a JOIN message to the multicast group. Following the same node joining protocol described above, node X is re-attached to another node on the multicast tree and the tree is repaired. If the number of on-tree hops of node X is different after the re-joining with its previous number of on-tree hops, node X sends an update message down the subtree

rooted at itself and every node in the subtree corrects its own number of on-tree hops accordingly.

The entries in the children table maintained at each forwarding node are also refreshed periodically; an entry is discarded unless a periodic message is received from the corresponding child node stating its desire to remain in the group.

Borg can also tolerate root node failures by replicating group management information. If the underlying p2p overlay is a Pastry network, the group management information can be replicated at the nodes in the root node's leaf set. When the root node fails, the tree repair procedure described above re-attaches the child nodes of the failed root to one of the nodes from its leaf set, i.e., the one whose nodeId is next closest to the groupId, and this node becomes the new root for the multicast group.

Scalability A potential scalability limitation of the forward-path multicast scheme (e.g. Bayeux) is that it generates more traffic when handling group membership changes. In particular, all group management traffic go through the root. In contrast, the reverse-path scheme (e.g. Scribe) distributes such traffic over the multicast tree nodes. In this regard, Borg is similar to Scribe, since subtrees in the lower part of the multicast tree are built using reverse-path forwarding, and these subtrees consist of a large portion of all the group members.

5. EVALUATION

This section presents the results of simulation experiments comparing the performance of Borg against Bayeux, a forward-path multicast scheme, and Scribe, a reverse-path multicast scheme. All three schemes are implemented on top of Pastry.

5.1 Experimental Setup

The simulations were performed using the same topology model as in Section 3, i.e., the Pastry network is formed with 64,000 end nodes randomly attached to the 1000 stub routers in the topology. Pastry is configured with a value 2 for b and a leaf set size 4. With this configuration, the average number of routing hops is 6, or $\frac{3}{4}\lceil \log N \rceil$ ($\frac{2^b-1}{2^b}\lceil \log N \rceil$). This is because with probability 1/4, the nodeId of each intermediate node during Pastry routing already shares the next digit with the key of the message being routed, and thus skipping a total of $\frac{1}{4}\lceil \log N \rceil$ hops during routing.

We simulated 500 multicast groups. Each multicast group has a rank that ranges from 1 to 500. The group size distribution follows the function: $Sub(r) = \lfloor Nr^{-1.25} + 0.5 \rfloor$, where r is the group rank (see Figure 4, as used in [6]). The largest multicast group (rank 1) has a subscription of 64,000 (every node is a subscriber, equivalent of broadcast) and the minimum number of subscribers is 27 (rank 500). Subscribers to each group were selected from the 64,000 nodes randomly and uniformly. In the simulation, a single message is sent to each of the 500 groups.

Three metrics were measured: relative delay penalty (RDP), node stress, and link stress. RDP was measured at each subscriber of each multicast group, and then averaged over all the subscribers of the group. Node stress is measured by the sum of fanout per group over all the groups at each node. Link stress is characterized by the number of messages sent over physical network links.

The bottleneck remover algorithm explained in Section 2.2.1 is applied to all three multicast schemes. The maximal fanout at each node is set to 100. Each time a new node is attached, the overall fanout is checked, and if the limit is exceeded, the bottleneck remover is invoked.

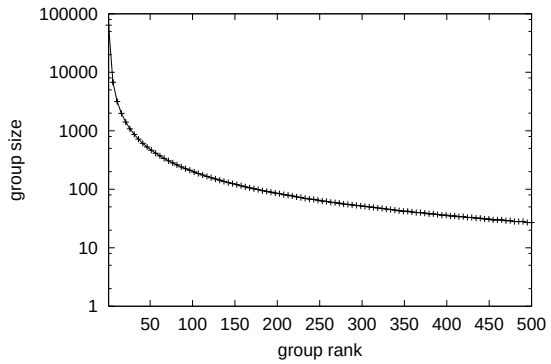

Figure 4: Distribution of multicast group size.

5.2 Choice of δ in Borg

We experimentally measured the performance of a broadcast using Borg varying the δ value. The RDP and link stress results for a broadcast to all 64,000 end nodes are shown in Figure 5 and Table 1, respectively. Figure 5 shows that Borg always gives lower RDP than the pure forward-path or reverse-path schemes, and the higher the δ value, the lower the RDP value. This is because longer segments of the paths will be optimized via exploiting the routing asymmetry. Table 1 shows that the higher the δ value, the higher the link stress incurred by Borg. This is because the higher the δ value, the shallower the lower subtrees which consist of short links from using reverse paths, and thus the higher the overall link stress. Overall, the δ value of 3 optimizes both the RDP and the link stress.

We also measured the optimal choice of the δ value for broadcasts to two Pastry overlays with 32,000 and 16,000 end nodes each. These two Pastry overlay networks were generated in a similar fashion as the Pastry overlay with 64,000 nodes, i.e., by randomly attaching 32,000 and 16,000 end nodes to the 1000 stub routers, respectively. The results are shown in Figure 6 and Table 2, and Figure 7 and Table 3, respectively. The results show that the value of δ that optimizes both the RDP and the link stress is 3 for both cases.

The above results show that when taking both RDP and link stress into account, setting the δ value of Borg to be half the average number of actual routing hops, $\lceil \frac{2^b-1}{2^b} \log N/2 \rceil$, gives a balanced low RDP and link stress. We therefore configure Borg to set its δ value as $\lceil \frac{2^b-1}{2^b} \log N/2 \rceil$. Note each Pastry node can estimate the overlay size N based on the density of nodes in its leaf set fairly accurately [3].

Simulations also show that only a small number of JOIN messages are received at the root of a Borg multicast tree. The percentage of JOIN messages that are received at the root for the broadcasts is below 10% for all the three Pastry overlay networks of 64,000, 32,000 and 16,000 nodes.

5.3 Results

Figures 8-10 compare the cumulative distribution of relative delay penalty, node stress, and link stress from sending a message to all 500 multicast groups using the three multicast schemes. First, it shows the RDP of Borg is 16% and

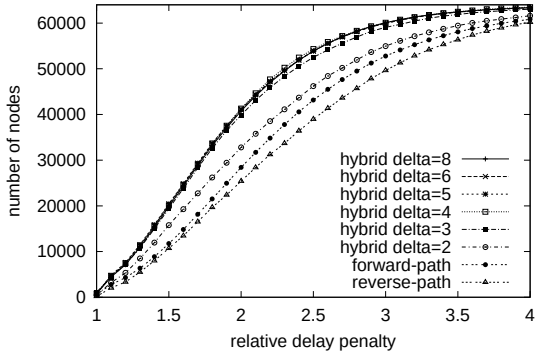

Figure 5: Cumulative distribution of RDP for a broadcast (averaged over 12 runs), varying δ in Borg, with 64,000 nodes. For clarity, the x-axis is cut off at 4.

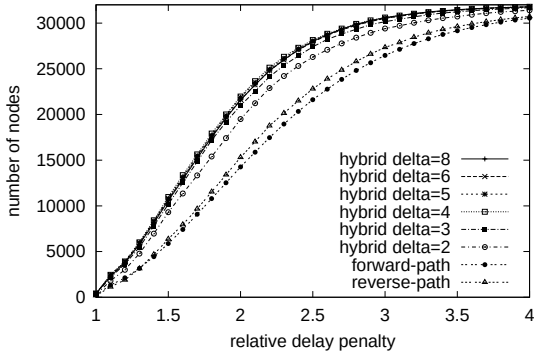

Figure 6: Cumulative distribution of RDP for a broadcast (averaged over 12 runs), varying δ in Borg, with 32,000 nodes. For clarity, the x-axis is cut off at 4.

Table 1: Link stress comparison for a broadcast (averaged over 12 runs), varying δ in Borg, with 64,000 nodes.

	mean	median	max	total
forward-path	4.5	1	2306.8	536807.3
reverse-path	1.2	1	79.0	140681.6
hybrid (δ=2)	1.2	1	79.4	140709.3
hybrid (δ=3)	1.2	1	91.9	141735.8
hybrid (δ=4)	1.2	1	172.2	149492.8
hybrid (δ=5)	1.4	1	404.5	177929.3
hybrid (δ=6)	1.9	1	767.6	235424.3
hybrid (δ=8)	2.7	1	1344.8	336911.9

Table 2: Link stress comparison for a broadcast (averaged over 12 runs), varying δ in Borg, with 32,000 nodes.

	mean	median	max	total
forward-path	4.4	1	1396.7	268547.4
reverse-path	1.2	1	74.6	75219.8
hybrid (δ=2)	1.2	1	74.7	75251.3
hybrid (δ=3)	1.3	1	73.8	76401.1
hybrid (δ=4)	1.4	1	123.8	84019.3
hybrid (δ=5)	1.7	1	363.2	105148.2
hybrid (δ=6)	2.1	1	640.9	135089.0
hybrid (δ=8)	2.5	1	839.9	157990.0

17% lower than those of the forward-path and the reverse-path schemes, respectively. Over the 500 multicast groups, the average RDP in using Borg, the forward-path scheme, and the reverse-path scheme are 1.96, 2.32, and 2.37, respectively. Second, it shows that the bottleneck remover effectively limits the maximal node stress for all three schemes. Third, Figure 10 and Table 4 show that the link stress for Borg is very close (within 4%) to that for the reverse-path scheme, and both are significantly lower (a factor 2.6) than that for the forward-path scheme.

6. RELATED WORK

Another multicast scheme built on top of p2p overlay networks is CAN multicast [21] built on top of CAN [20]. CAN multicast creates a separate CAN overlay for each multicast group, and then floods multicast messages to all nodes. Recently, Castro et al. [7] experimentally compared multicasts built on CAN-style overlays versus on Pastry-style overlays and showed that tree-based multicast built on top of Pastry provides better performance than on top of CAN.

There have been a large body of work on application-level multicast. Overcast [17] implements an overlay network consisting of a single source and internal nodes placed at different locations of an existing network. Narada [16] uses a two-step process to build the multicast tree: First, it builds a mesh per group containing all the group members. Second, it constructs a spanning tree of the mesh for each source to multicast data. REUNITE [25] separates the branching points of the multicast tree from the non-branching points and only the branching points keep multicast state information. Thus the scalability is improved. Chu et al. [15] investigate the performance of ESM in a dynamic and heterogeneous Internet environment in the context of audio and video conferencing. TAG [18] exploits network topology information to build the multicast overlay. It utilizes the path overlap among members to reduce delay penalty. Chalmers and Almeroth [8] investigate the fundamental characteristics of global multicast trees to understand the efficiency gains of multicast over unicast. Cohen and Kaempfer [9] propose a unicast-based multicast for real-time multimedia streaming. Costa et al. [10] takes into account the unicast asymmetries of the Internet when building the multicast tree. Shi and Turner [23] propose and compare algorithms to optimize the degree distribution of multicast overlays composed of specialized Multicast Service Nodes (MSNs) which serve end hosts by unicast. Zhang et al. [27] seek to bridge the IP multicast enabled islands with the rest of the Internet smoothly.

7. CONCLUSION

A scalable implementation of multicast is the first step towards supporting multimedia streaming over wide-area networks. Building multicast on top of decentralized, scalable, and reliable peer-to-peer overlay networks offers a promising approach. This paper presented Borg, a hybrid multicast protocol that combines forward-path multicast and reverse-path multicast to build multicast trees. Simulation results in a Pastry network of 64,000 nodes based on a realistic network topology model have confirmed that Borg offers significantly lower routing delay penalty than both forward-path

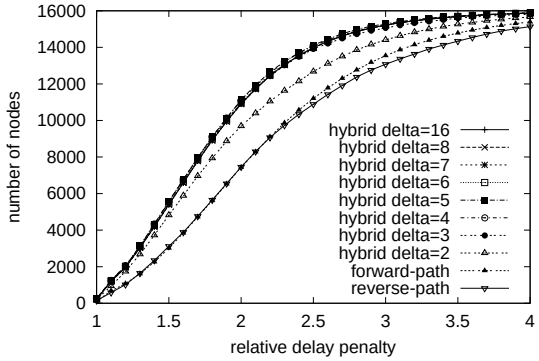

Figure 7: Cumulative distribution of RDP for a broadcast (averaged over 12 runs), varying δ in Borg, with 16,000 nodes. For clarity, the x-axis is cut off at 4.

Table 3: Link stress comparison for a broadcast (averaged over 12 runs), varying δ in Borg, with 16,000 nodes.

	mean	median	max	total
forward-path	4.3	1	663.3	134287.8
reverse-path	1.3	1	69.5	41655.6
hybrid (δ=2)	1.3	1	69.3	41678.8
hybrid (δ=3)	1.3	1	75.4	42422.3
hybrid (δ=4)	1.4	1	102.4	47087.5
hybrid (δ=5)	1.8	1	169.3	59434.1
hybrid (δ=6)	2.2	1	306.0	75278.8
hybrid (δ=8)	2.5	1	375.3	83962.8

and reverse-path multicast schemes while retaining the low link stress of the reverse-path multicast scheme.

Acknowledgments

This work was supported by an NSF CAREER award (ACI-0238379).

8. REFERENCES

[1] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, et al. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[2] L. F. Cabrera, M. B. Jones, and M. Theimer. Herald: Achieving a global event notification service. In *HotOS VIII*, May 2001.

[3] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, , and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *5th Symposium on Operating Systems Design and Implementaion (OSDI'02)*, December 2002.

[4] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting Network Proximity in Distributed Hash Tables. In *Proceedings of International Workshop on Future Directions in Distributed Computing*, June 2002.

[5] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks. Technical report, Technical report MSR-TR-2002-82, 2002.

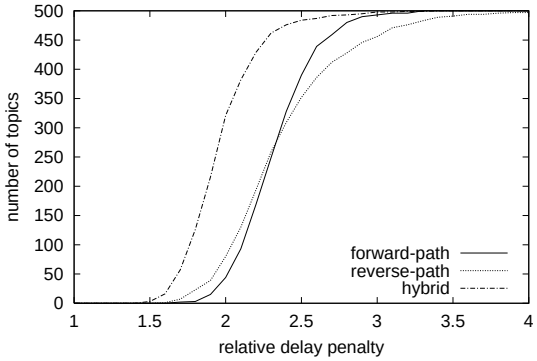

Figure 8: Cumulative distribution of RDP for the three multicast schemes.

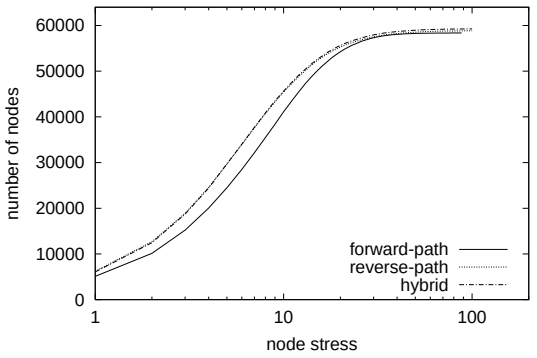

Figure 9: Cumulative distribution of node stress for the three multicast schemes.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication (JSAC)*, 20(8), October 2002.

[7] M. Castro, M. B. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman. An evaluation of scalable application-level multicast built using peer-to-peer overlays. In *Infocom'03*, April 2003.

[8] R. C. Chalmers and K. C. Almeroth. Modeling the Branching Characteristics and Efficiency Gains in Global Multicast Trees. In *Proceedings of IEEE INFOCOM*, April 2001.

[9] R. Cohen and G. Kaempfer. A Unicast-based Approach for Streaming Multicast. In *Proceedings of IEEE INFOCOM*, April 2001.

[10] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte. Hop By Hop Multicast Routing Protocol. In *Proceedings of ACM SIGCOMM*, August 2001.

[11] Y. K. Dalal and R. Metcalfe. Reverse Path Forwarding of Broadcast Packets. *Communications of the ACM*, 21(12):1040–1048, 1978.

[12] S. E. Deering and D. R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.

[13] P. Eugster, P. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe.

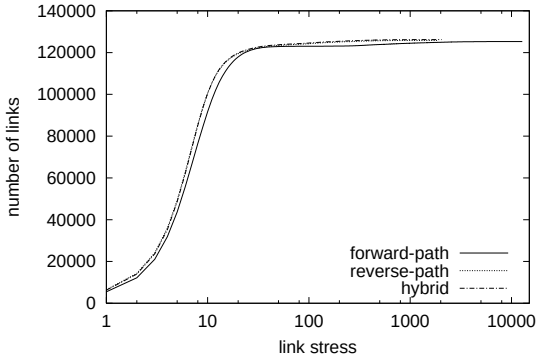

Figure 10: Cumulative distribution of link stress for the three multicast schemes.

Table 4: Link stress comparison.

	mean	median	max	total
forward-path	29.6	7	12791	3703785
reverse-path	11.1	7	1839	1392766
hybrid	11.5	7	2047	1446655

Technical Report DSC ID:2000104, EPFL, January 2001.

[14] P. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, and P. Kouznetsov. Lightweight Probabilistic Broadcast. In *Proceedings of The International Conference on Dependable Systems and Network*, July 2001.

[15] Y. hua Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM*, August 2001.

[16] Y. hua Chu, S. G. Rao, and H. Zhang. A Case For End System Multicast. In *Proceedings of ACM Sigmetrics*, pages 1–12, June 2000.

[17] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, pages 197–212, October 2000.

[18] M. Kwon and S. Fahmy. Topology-Aware Overlay Networks for Group Communication. In *Proceedings of NOSSDAV*, May 2002.

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM*, August 2001.

[20] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM*, August 2001.

[21] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level Multicast using Content-Addressable Networks. In *Proceedings of the Third International Workshop on Networked Group Communication*, November 2001.

[22] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *International Conference on Distributed Systems Platforms (Middleware)*, November 2001.

[23] S. Y. Shi and J. S. Turner. Routing in Overlay Multicast Networks. In *Proceedings of IEEE INFOCOM*, June 2002.

[24] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*, San Diego, California, August 2001.

[25] I. Stoica, T. S. E. Ng, and H. Zhang. REUNITE: A Recursive Unicast Approach to Multicast. In *Proceedings of IEEE INFOCOM*, March 2000.

[26] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of IEEE INFOCOM*, March 1996.

[27] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A Framework for Delivering Multicast To End Users. In *Proceedings of IEEE INFOCOM*, June 2002.

[28] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An Infrastructure for Fault-Resilient Wide-area Location and Routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.

[29] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination. In *Proceedings of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video*, June 2001.