RITA: Receiver Initiated *Just-in-Time* Tree Adaptation for Rich Media Distribution

Zhichen Xu, Chunqiang Tang[*], Sujata Banerjee, and Sung-Ju Lee
Internet Systems & Storage Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94304
{zhichen,sujata,sjlee}@hpl.hp.com, sarrmor@cs.rochester.edu

ABSTRACT

Application-level multicast networks overlaid on unicast IP networks are increasingly gaining in importance. While there have been several proposals for overlay multicast networks, very few of them focus on the stringent requirements of real-time applications such as streaming media. We propose RITA (Receiver Initiated Timely Adaptation) framework for an efficient overlay multicast infrastructure. RITA is based on a combination of landmark clustering and RTT measurements, and is particularly suitable for multimedia real-time applications. Our goal is to balance the network-oriented goals of building an efficient multicast tree with the application-oriented goals of providing good QoS with minimal disruptions. Using accurate global soft state information tables, our approach promptly constructs and reconfigures high quality trees. A distinguishing feature of our approach is that the tree reconfiguration is initiated *just-in-time* by the application client at the receiver when the media quality falls below a specific threshold. The goal is to achieve dynamic tree reconfiguration with very low switching delay such that end users do not perceive any application performance degradation.

Categories and Subject Descriptors

C.2.1 [**Computer Systems Organization**]: Computer-Communication Networks—*Network Architecture and Design*

General Terms

Algorithms, Performance, Design, Experimentation

Keywords

streaming media, overlay networks, multicast, DHT

[*]Chunqiang Tang is with Department of Computer Science, University of Rochester, Rochester, NY.

1. INTRODUCTION

We envision a future where a large fraction of applications use multimedia objects and streams. The significant drivers for this trend are the widespread proliferation of high quality digital cameras and multimedia authoring tools as well as the availability of the necessary computing, storage and networking resources to create and deliver rich-media content. Internet users today can host web content (primarily text and small images) from their home. In the future, it will be just as easy for anyone with a PC and a digital camera to be a cinematographer, editor and director of his or her own movie and then disseminate a high quality media stream to a very large remote audience over the high speed Internet. An alternative futuristic scenario may be for family members to "join" a Thanksgiving gathering in real time from remote locations using similar media and networking technologies. This vision is articulated in [3] with descriptions of several other scenarios. An attractive possibility is for these applications to function by setting up peer-to-peer (P2P) communities, requiring minimal support from the underlying infrastructure and thus quickly deployed.

Scalable and efficient multicasting is essential to enable the above vision. Multicasting provides significant bandwidth savings and is particularly crucial for the dissemination of live as well as stored high fidelity multimedia content because of the sheer size of the content, the relatively long duration of the session, and the correspondingly high bandwidth requirements. Moreover, multimedia applications have very stringent delivery requirements without which the user perceived quality will suffer. Thus, in addition to scalable multicasting, supporting some level of end-to-end quality of service (QoS) is also a key requirement in realizing this vision.

Due to a variety of deployment issues, including high management complexity and cost, either IP multicast or Internet QoS are not widely supported today in the Internet infrastructure. However, spurred by the demand for exciting multimedia applications, several application level overlay schemes have been recently proposed. Most schemes primarily deal with overlay multicasting [7] while some consider overlay QoS [25]. Some schemes such as Host Multicast Tree Protocol (HMTP) [29] leverage IP multicast if available. Very few overlay schemes consider both multicasting and end-to-end multimedia QoS issues, which is the subject of this work.

We propose an efficient application layer multicast infrastructure for multimedia real-time applications, which greatly benefits from a global view of the system stored in a distributed hash table (DHT). DHT systems such as Content Addressable Network (CAN) [19], Pastry [22], Chord [24], and Tapestry [30] offer an administration-free and fault-tolerant storage utility. Nodes in these systems collectively contribute towards a storage space in a self-organizing fashion. In these systems, there is a consistent binding between objects and nodes. Locating an object is reduced to the problem of routing to the destination node that stores the object. The logical structure of these systems provides some guarantee with respect to the number of logical hops that need to traverse to locate an object. Various techniques have been proposed to map the logical structure to fit the topology of the underlying physical network [6, 21, 27, 28].

In this paper, we propose RITA (Receiver Initiated Timely Adaptation) framework for an efficient overlay application layer multicast infrastructure. The global view is generated from landmark clustering [16]. Combining the landmark information with a small number of round-trip time (RTT) measurements to locate physically close-by neighbors, RITA provides very fast, high quality tree construction and adaptation. There are three key differences between RITA and that of prior work.

- None of the existing schemes addresses the problem of media quality disruption during the tree reconfiguration. Our goal is to develop mechanisms that transparently reconfigure the overlay tree in very short timescales such that the user's perceptual quality does not suffer during the reconfiguration process. We focus on perceptual quality because not all fluctuations in network quality negatively affect the application perceived QoS.
- Unlike other schemes which advocate periodic tree reconfiguration or event-triggered reconfiguration (e.g., when RTT increases), RITA performs *just-in-time* reconfigurations driven by application/user perceived QoS. This approach provides a natural reconfiguration timescale and avoids the overhead of unnecessary changes to the tree that do not affect the end client's perceived quality.
- Most of the schemes require several seconds to reconfigure or complete a join to the tree. Utilizing the landmark information stored in the DHT, RITA requires far fewer network measurements with an aim to perform tree reconfigurations under a second, while producing a tree that is reasonably close to the optimal in terms of bandwidth efficiency. Compared with an HMTP like, our algorithm speeds up the tree construction by a factor of up to 16, while often producing highly efficient trees.

The rest of the paper is organized as follows. We describe the RITA approach in Section 2. An evaluation of RITA using simulations is presented in Section 3. An overview of the related work is presented in Section 4. We conclude the paper with a discussion on open issues and future directions in Section 5.

2. RECEIVER INITIATED TIMELY ADAPTATION FRAMEWORK

The quality of a multicast tree, to a great extent, depends on how close its structure approximates to that of the underlying physical network. One simple way to construct efficient trees is to measure the end-to-end latency between each pair of the nodes and run a shortest path tree algorithm over the resulted graph. This approach however, is not practical for large trees due to the excessive measurements required. In this section, we present the approach taken by the RITA (Receiver Initiated Timely Adaptation) framework.

The fundamental idea of utilizing global information to create an efficient tree is crucial. To maintain global state, we propose to select nodes to form a DHT to serve as a rendezvous plane to store information of nodes in the tree. DHT systems such as CAN [19], Pastry [22], Chord [24], and Tapestry [30] provide a hash table abstraction that allows efficient storage and retrieval of `(key, object)` pairs. In such systems, each node only need to know the addresses of logarithmic number of other nodes in the system and a node can be reached in logarithmic number of routing steps. Nodes in these system collectively contribute towards an administration-free and fault-tolerant storage space in a self-organizing manner.

We use the *landmark vectors* [16] of nodes as keys to store their information in the DHT such that information of nodes that are physically close to each other are stored near each other in the DHT [28]. As a result, a node finds information of close-by nodes in an efficient and scalable way. Note that there are many Internet host distance measurement schemes (e.g., IDMaps [11], King [12]) in addition to the landmarks technique, and it is possible for RITA to work in conjunction with these schemes as well.

Our experiments show that if a new node finds the closest node in the tree and attaches to it, the tree cost (defined as the aggregated weight of tree edges) is comparable to that of a shortest path tree with less than 30% overhead for trees with up to 2,048 nodes. To find the closest node, we propose a technique that combines landmark clustering and actual RTT measurements [28]. In our model, a new node always attaches to the identified closest node using the above technique.

Next we provide the details involved in locating the closest node to any given node, followed by a description of the tree construction algorithm. Then a tree adaptation algorithm that quickly responds to changing network conditions is described. The goal of this algorithm is to minimize the disruption to the end users.

2.1 Locating the Closest Node

We use *landmark clustering* as a pre-selection process to identify nodes that are possibly close, and use RTT measurements to locate the actual closest node. The intuition behind landmark clustering is that if two nodes have similar latencies to a common set of *landmark* nodes, they are likely to be close to each other. In particular, we adopt the technique proposed in [16]. Landmark nodes measure the RTTs among themselves and use this information to compute coordinates in a Cartesian space for each of them. These coordinates are then distributed to the clients, which measure RTTs to the landmark nodes and compute their own coordinates. We call these coordinates *landmark vectors*. The Euclidean distance between nodes in the Cartesian space is directly used as an estimation of the network distance.

Each node uses its landmark vector as the key to store its *profile* in the DHT.[1] This controlled placement of node profiles has the effect that information of nodes that are

[1]We use CAN to store the landmark information. CAN provides a DHT over a Cartesian space and therefore, we use the landmark vectors as the DHT keys.

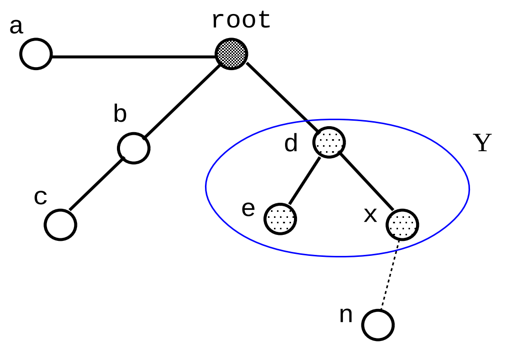


Figure 1: Tree construction.

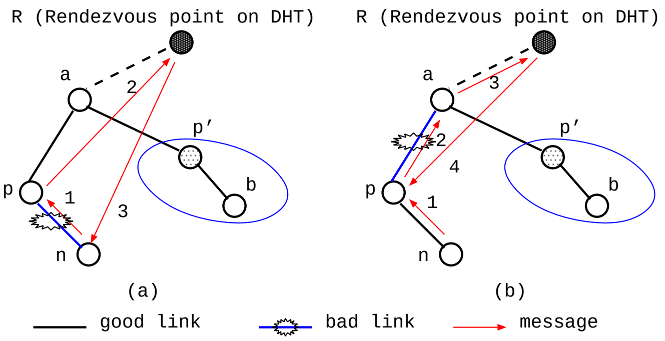


Figure 2: Tree adaptation.

physically close are stored nearby in the DHT. To find a set of physically close-by nodes, a node uses its landmark vector as the key to look up the DHT. The node ranks the identified neighbors according to the similarity in landmark vector, and measures RTTs to top candidates to locate the closest one. Our experiments show that for a typical topology with 10,000 nodes, using 15 landmarks and 20∼40 RTT measurements accurately locates a close-by node [28].

In reality, finding the right node to attach to in the tree is a multi-faceted problem that is more complex than just finding the closest node, especially when nodes are heterogeneous. For example, the network access link speed of a node is one of the deciding factors for the maximum media streaming rate to that node. If the source streaming rate is higher than the network access speed of a node, this node needs to attach to a tree node that can transcode the media down to the desired rate. To address this issue, each node includes rich information about itself in the profile and periodically updates the profile with its current state. The profile for a node may include its landmark vector, its network access type and speed (e.g., 56kb/s dialup line, 1.5Mb/s DSL or 100Mb/s LAN), current and maximum fan out in the multicast tree, processing power, current load, special capabilities such as transcoding, and so forth. When searching for candidate nodes to attach to, a node that is joining the tree considers not only network proximity but also its special QoS requirements.

2.2 Basic Tree Construction Algorithm

We illustrate our tree construction algorithm in Figure 1. We select nodes that have good capacity and network connectivity in the tree to form a DHT and store node profiles in the DHT. When a new node n wants to join the multicast tree, it computes its own landmark vector and carries out the following steps:

1. Uses its own landmark vector as the key to look up the DHT, obtaining information about a set of nodes X whose landmark vectors are similar to its own. Based on this information, it eliminates the nodes in X that do not satisfy the QoS requirements (bandwidth, CPU load, etc.). The remaining nodes form a set Y, which includes nodes d, e and x in Figure 1.
2. Performs concurrent RTT measurements to each node in Y and identifies the node that is the closest. Let us denote the closest node as x.
3. Attaches to x as its child.

This simple algorithm offers the following advantages: (i) Because RTT measurements to each node in Y are performed concurrently, a new node quickly locates the potentially closest node without level-by-level tree traversal. (ii) We have a global view of the system that enables us to find a good neighboring node that satisfies the QoS constraints without having to contact a significant number of nodes.

2.3 Tree Adaptation Algorithm

Driven by the inherent dynamics in the underlying infrastructure, we propose two kinds of tree adaptations—a *just-in-time* adaptation to address application quality issues, and a long-term adaptation to address tree efficiency issues. The long term adaptation has a relatively large period in the order of several minutes or even hours, and it is actually carried out only if doing so can result in significant bandwidth savings.

The *just-in-time* adaptation is driven by application perceived QoS that is impacted by the fluctuations in the quality of the links. To provide the end users with reasonable QoS, the tree must continuously adapt to these changing conditions and minimize any service disruption to the end users. This translates into finding the best location to perform the adaptation and minimizing the delay for each repair. We assume that all tree nodes can monitor its end application QoS or the quality of the end-to-end path and translate this to user-perceived QoS.

We demonstrate our tree adaptation algorithm using Figure 2. When a node n perceives a QoS degradation over its tolerance threshold, it sends a complaint to its parent p in the tree along with its own landmark vector. If p is not responsive, n switches to a new parent by performing a new join process initiated at the root. If p responds, we have two cases as shown in Figure 2 (a) and 2 (b), respectively.

1. p **is happy with its QoS**, which indicates that the bottleneck link lies on the path (p, n). p forwards the complaint initiated by n directly to the DHT infrastructure (depicted as a rendezvous node R on the DHT), which will provide n with a new set of candidate parent nodes that are close to n judging from the landmark vectors. In Figure 2 (a), this candidate set includes p' and b. Similar to the tree construction process, n chooses its new parent, e.g., p', based on the measured RTTs to candidates and the QoS they can provide. n then carries out the switching with the handoff process we describe later.
2. p **is also unhappy with its QoS**, which indicates that the bottleneck link exists on the upstream path, e.g., path

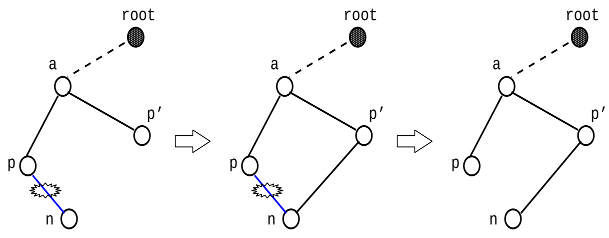


Figure 3: Multi-homed handoff process.

(a, p). In this case, p starts its own complaint process by sending a message containing its own landmark vector to its parent a. Note that by the time that the complaint from n arrives at p, p may already have sent a complaint to a based on its perceived QoS. In this case, p will suppress n's complaint. These concurrent complaints may save significant time in adaptation.

In Figure 2 (b), because a is happy with the QoS it perceives, it directs the complaint to the DHT node R, which will instruct p to switch to a new parent with the candidate set including p' and b. p then measures the RTTs to these nodes and switches to p'. During this process, n waits for the QoS to improve, or an instruction from R to switch to a new parent. If it is still unhappy after a timeout, i.e., there are multiple bottleneck links on its upstream path, it starts the complaint process again.

To prevent routing loops, each node records the path from the root to itself. If a node searching for a new parent finds that it is on the path from the tree root to a candidate parent node, then this candidate node is not selected as the new parent. Tree adaptation could also cause oscillations where a node keeps switching back and forth among a set of candidate parent nodes. To avoid this problem, each node caches the parent nodes of the recent past and does not choose a node in the cache as the new parent.

Our tree-adaptation algorithm minimizes the overall disruption by locating the problematic link and having the node incident to that link to adapt. For instance, when the quality of a link close to the root degrades, instead of asking every downstream node of that link to find a new parent, our local repair algorithm requires only the node incident to that link to attach to a new parent.

It typically takes three steps to obtain a set of parent candidates. Assume that the (n, p) and (n, p') distances are 20ms since n is close to p and p', and (p, R) and (n, R) are 100ms. Considering that routing in the DHT typically takes double the latency of IP routing [28], it takes approximately 320ms to obtain the candidate sets. Assume that we do three rounds of concurrent RTT measurements to all candidates and select the candidate that has the lowest RTT. This will take additional 120ms. This leaves us with 560ms to complete the entire switching under one second.

2.4 Smooth Application Handoff

One of our goals is to develop methods to ensure that the application performance suffers minimally and the tree reconfiguration is conducted transparently. Because switching to a new parent may incur some delay, it is essential to maintain the performance levels during the parent handoff process. For media applications, this is crucial as the user perceived media quality may suffer significantly, if there is a sudden high loss or delay inflicted by the handoff. To minimize disruption, we use multi-homing at the multicast overlay layer during the handoff period, similar to [23]. The idea is to have a child connected to both the new and the old parents, and receive application packets from both until the handoff is complete.

We illustrate the handoff process in Figure 3. When a node n needs to switch from its current parent p to a new parent p', it contacts p' for connection establishment. The connection between n and p will be torn down after the two flows from p and p' are synchronized. It is possible to design more sophisticated algorithms that reduce the amount of duplicate traffic sent to n. It should be noted that during a short period, certain links that lead to node n now may have to carry traffic from both parents. If one of the links is a bottleneck link, this can temporarily worsen the situation. If the QoS degradation is caused by a bottleneck link that is on the paths from all potential new parents, then no repair is possible. The challenge is to promptly detect this scenario so that unnecessary repair is halted.

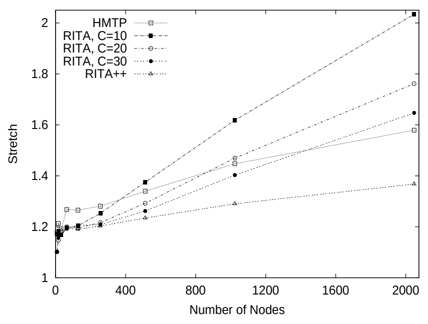
3. PERFORMANCE RESULTS

To evaluate the effectiveness and efficiency of RITA, we compare the quality of the trees constructed by RITA with that of trees produced by protocols that probe the tree level-by-level (e.g., HMTP). In HMTP-like protocols, a new node wishing to join the multicast traverses down the tree from the root to search for the closest node to itself at each level until it reaches a leaf. This process finds a shortest delay path starting from the root. Among all nodes on this path, the new node attaches to the closest one.

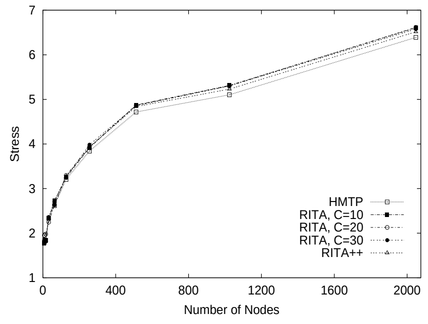
We use two 10,000-node topologies produced by GT-ITM [5]. Topology A has 25 transit domains, five transit nodes per transit domain, four stub domains attached to each transit node, and 20 nodes in each stub domain, with a maximum end-to-end one-way latency of 31.96ms. Topology B has a larger transit domain. It has 228 transit domains, two nodes in each stub domain, and a maximum latency of 34.53ms. We randomly select 100 nodes and use the k-mean method (described in [16]) to select 15 landmarks out of the 100 randomly selected nodes.

Two tree quality metrics - *stretch* and *stress* are used in the evaluation. *Stretch* is the ratio of the tree cost (the sum of delays of the links in the overlay tree) to that of a shortest path tree. The *stress* of an overlay multicast tree is the average number of overlay links over a given physical link in the underlying topology. The goal is to create trees with both stretch and stress as close to 1.0 as possible, which would be the case if IP multicast were used. We also compare the time that it takes to construct the trees and define *speedup* as the ratio of time taken by HMTP to that taken by RITA.

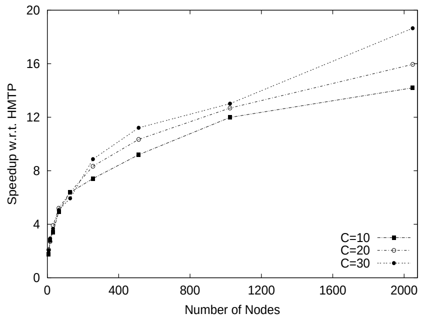
Note that the performance of RITA depends on the number of C closest candidates that the DHT presents to the node x wishing to join the tree. Node x performs concurrent RTT measurements to the C nodes and picks the one that has the lowest RTT. As an extreme case, we consider $C = N - 1$, where N is the number of overlay nodes in the tree. When $C = N - 1$, the actual closest candidate node is found and this case is labeled RITA++ in the results. As C is increased, the tree quality of RITA improves, and approaches that of RITA++, at the expense of the large tree construction latency.



(a) Stretch.

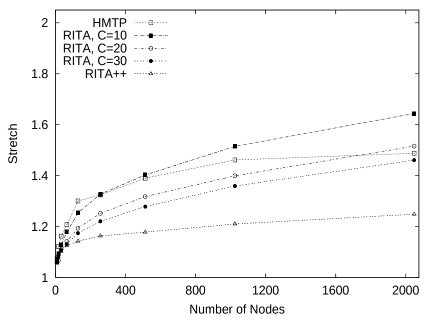


(b) Stress.

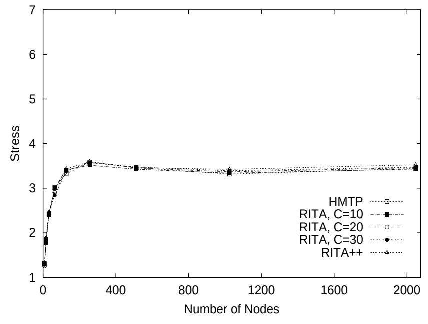


(c) Construction speedup.

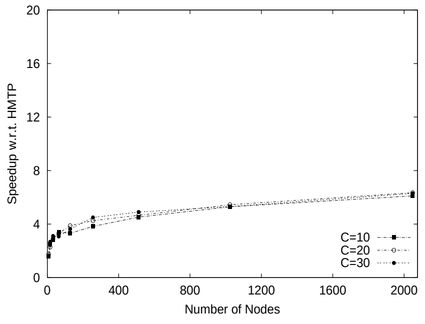
Figure 4: Topology A: stretch, stress and speedup of RITA, HMTP and RITA++.



(a) Stretch.



(b) Stress.



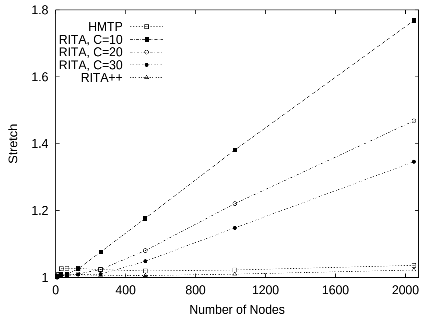
(c) Construction speedup.

Figure 5: Topology B: stretch, stress and speedup of RITA, HMTP and RITA++.

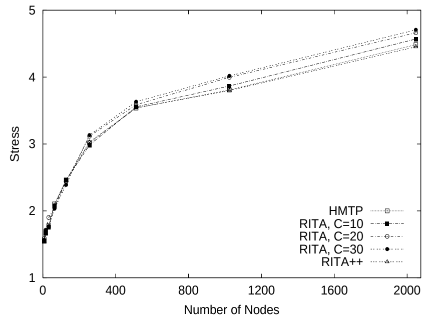
3.1 Tree Quality and Construction Speedup

In Figures 4 and 5, we present *stretch* and *stress* for RITA (with $C = 10, 20, 30$), HMTP and RITA++ for topologies A and B, as the number of overlay nodes is increased from 8 to 2,048 nodes. We also present the tree construction speedup of RITA as compared to HMTP. The results presented are averaged over 20 runs for each data point. We observe the following from these figures.

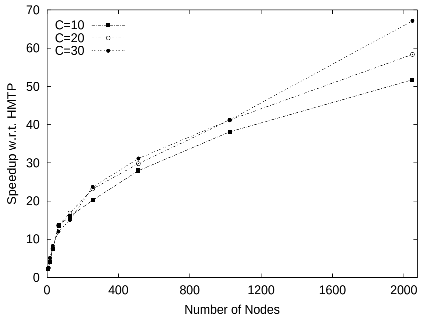
- As expected, all protocols produce lower quality trees as the number of overlay nodes increases.
- RITA++ produces trees with very low stretch, thus validating the basic approach of attaching to the closest node. Any further improvement in the accuracy of techniques to identify the close by nodes will have direct positive impact on the performance of RITA.
- An HMTP-like protocol produces very high quality trees, though RITA with $C = 20$ and $C = 30$ outperform HMTP, particularly in topology B with a large number of transit domains.
- There are some differences in the results for the two topologies. Both the stretch and stress in topology A are worse than in topology B. This is due to two reasons: (i) The landmark clustering and RTT scheme is not as effective for a topology with a small transit than for a network with a large transit. Increasing the number of candidate nodes (C) in RITA has a larger effect in the large-transit network. (ii) Even for RITA++, the stretch in the small transit graph is still worse than in the large-transit graph. Note that for the small transit network, although the relative overhead (i.e., stretch) seems large, the absolute overhead is small, because the latencies are short and the penalty for not finding the closest node is small.
- The stress for the three protocols are similar to each other at about 3.5 for the large-transit topology and do not vary much as the number of nodes is increased. Note that a stress of 3.5 is considered reasonable and better than what most other overlay multicast protocols provide. However, in the small-transit topology, the stress increases as the number of tree nodes increases, and is as large as 6.5 when there are 2,048 nodes in the tree. This is because with a small-transit, there are more nodes in the stub domains and nodes are more likely to share the shortest path links.
- The tree construction speedup results show that as the tree size grows, the cost of constructing the tree using a HMTP-like protocol increases more rapidly than that using RITA. As a result, the tree construction speedup enjoyed by RITA over HMTP is near 20 for the small



(a) Stretch.

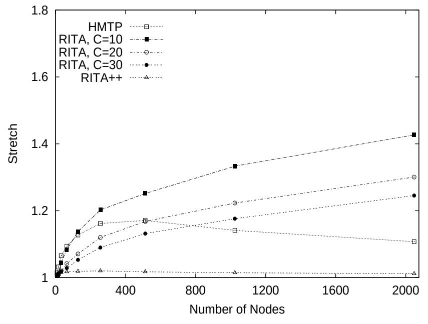


(b) Stress.

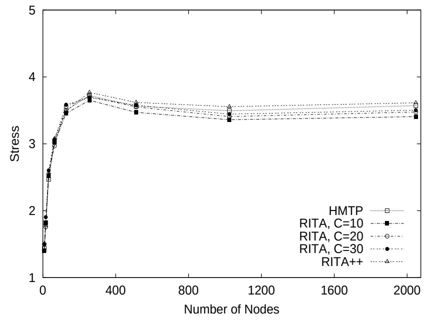


(c) Construction speedup.

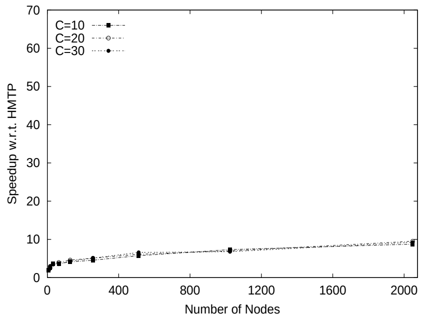
Figure 6: Topology A: stretch, stress and speedup with optimal joining order.



(a) Stretch.



(b) Stress.



(c) Construction speedup.

Figure 7: Topology B: stretch, stress and speedup with optimal joining order.

transit network, and can be as large as 6.4 for the large transit network. As the number of nodes in the tree increase, the speedup does not decrease much as we can identify the closest node as soon as the closest node (in the set of C candidates) responds to the measurements.

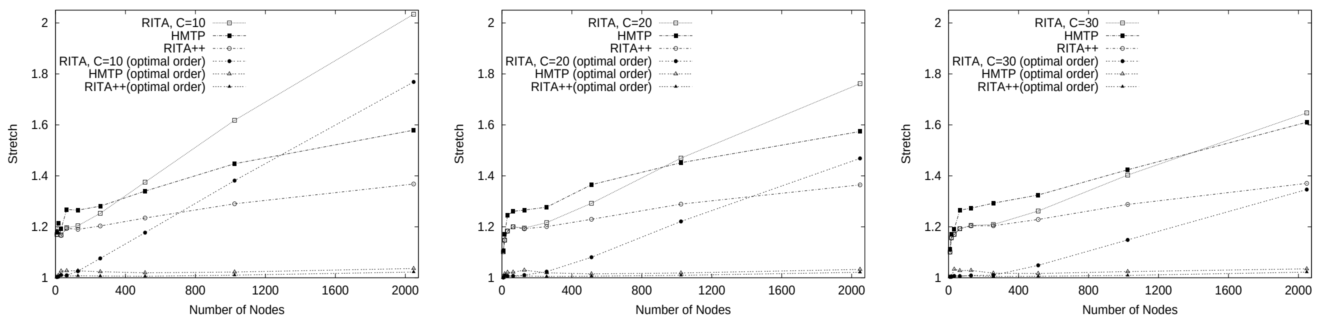
3.2 Benefits of Periodic Reconfiguration

In practice, overlay nodes join a multicast session at different times and in a random order. The results described so far mimic this reality and the nodes join in a random order. Here we study the effect of the joining order on the tree structure and quality. If the join order has a large impact, we can periodically reconfigure the tree in addition to the just-in-time adaptation to construct a more efficient tree. Intuitively, a near optimal tree is built if the nodes join in the order of the shortest distance to the multicast source; when a node joins the system, the most suitable parent for this new node is most likely already in the tree. By comparing the evaluation metrics (stretch, stress and speedup) when nodes join in this near optimal joining order versus a random order, we obtain insights into the benefits of periodic tree reconfiguration.

Figures 6 and 7 show the stretch and stress for RITA, HMTP, and RITA++ assuming a (near) optimal joining order of overlay nodes for topologies A and B, respectively. The resulting speedups of RITA over HMTP are also given. We compare these results with the results presented earlier in Figures 4 and 5. Figures 8 and 9 take a closer look and provide a more detailed comparison of the stretch metric for the random and optimal tree join orders for the two topologies. These figures show that the stretch of the trees produced by RITA, HMTP and RITA++ reduces when nodes join in the optimal order. For RITA++ and HMTP, the improvement is more dramatic than RITA (with $C = 10, 20, 30$). Particularly RITA++ now produces trees with a stretch very close to 1.0, the optimal case for both topologies. For the small transit network, HMTP also produces close to optimal stretch. The stress metric is unchanged when nodes join in the optimal order for the large transit network and decreases slightly for the other topology. The speedup on the other hand of RITA over HMTP increases to near 70 in the small transit network and near 10 for the other topology. This is mainly due to the fact that now the trees produced tend to be deeper rather than broader trees and this increases the HMTP costs more than that of RITA.

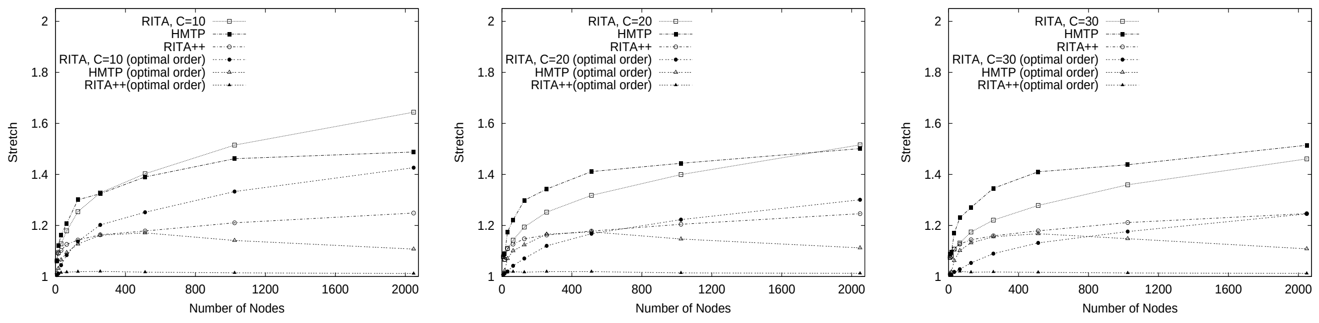
3.3 Tree Adaptation

The just-in-time tree adaptation in RITA addresses ap-



(a) C = 10. (b) C = 20. (c) C = 30.

Figure 8: Topology A: stretch comparison with random and optimal joining orders.



(a) C = 10. (b) C = 20. (c) C = 30.

Figure 9: Topology B: stretch comparison with random and optimal joining orders.

plication quality issues, and hence the time to complete the adaptation must be as little as possible. There are two components to the just-in-time adaptation. The first is the latency of the quality degradation complaints to propagate to the problem link and identify the node that needs to switch to a new parent. The second component is the re-attachment process to a new parent. It is the second component that is of greater interest and the delay to accomplish this task is computed in this section.

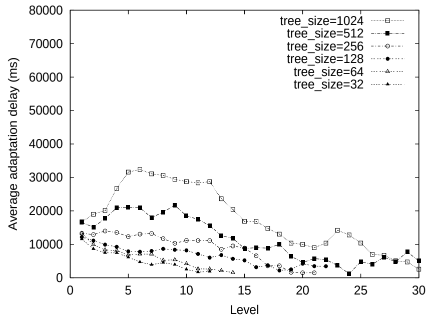
Intuitively, the overlay nodes at the same depth or level of the overlay multicast tree are expected to experience similar re-attachment delays. Thus our experiments were designed to take this into account by computing the average time to re-join the tree at each level for both topologies and at different tree sizes. Figures 10 and 11 plots the average re-joining delay versus the level of the multicast tree for topologies A and B, respectively and for $C = 10, 20, 30$.

These figures depict that the average re-joining latency for RITA increases as the number of overlay nodes increases. As expected, the maximum time to rejoin the tree is shorter for the small transit network at about 300ms, while for the large transit network, 720ms. We make two important observations:

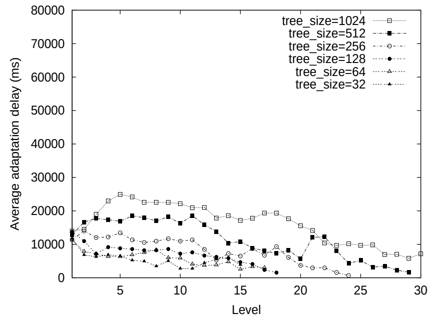
- As the number of candidates C increases, it is more likely to find an alternate parent that is close by, and hence reducing the overall re-joining latency.
- The average re-joining delay decreases as node level increases, i.e., as the node gets further away from the source. This result may seem unintuitive but since nodes cannot re-attach to children nodes, the choice set is larger for the nodes deeper in the tree.

4. RELATED WORK

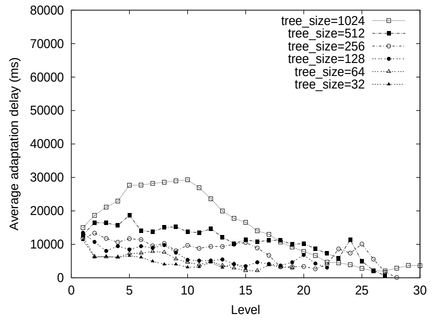
Several application-level multicast schemes achieve data distribution by *implicitly* building a multicast structure. For instance, Scribe [7] is a multicast infrastructure built on top of Pastry [22]. In Scribe, the multicast tree is formed by the union of the Pastry routes from multicast members to the rendezvous point (RP). The Content-Addressable Network (CAN) framework [19] is extended for multicast in [20]. In this work, the multicast group members establish a mini-CAN and multicast data is distributed by flooding over the mini-CAN, without explicitly building a tree. Bayeux [31] is an architecture built on top of Tapestry [30] and supports source-specific multicast. The NICE is the Internet Cooperative Environment (NICE) protocol [2] builds and maintains hierarchical topology of multicast members. The multicast routes are implicitly defined by the hierarchy struc-



(a) C = 10.

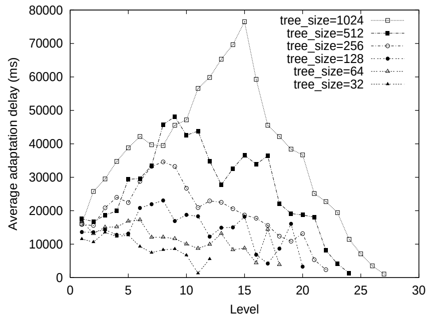


(b) C = 20.

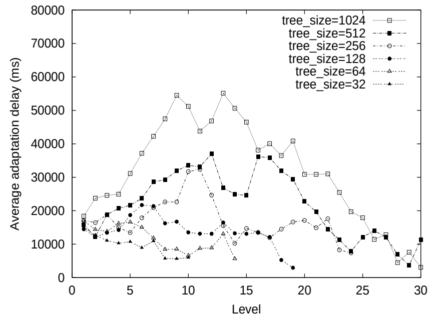


(c) C = 30.

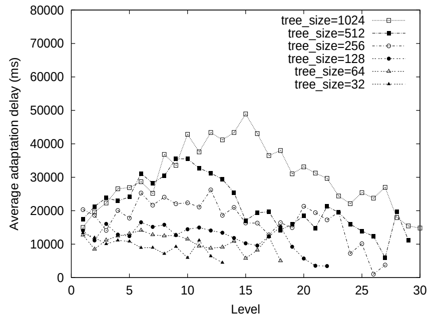
Figure 10: Topology A: RITA's average adaptation delay.



(a) C = 10.



(b) C = 20.



(c) C = 30.

Figure 11: Topology B: RITA's average adaptation delay.

ture. A protocol that uses a Delaunay triangulation as an overlay network topology is proposed in [15]. With the distributed construction of a Delaunay triangulation, multicast paths are embedded in the overlay without a routing protocol. Overlay Multicast Network Infrastructure (OMNI) [4] proposes a two-tier architecture and builds a multicast tree consisting of multicast service nodes (MSN) which in turn connect to clients. This distributed scheme is adaptive with changes in the client distribution and network conditions.

The following protocols *explicitly* form the multicast tree. Targeting at content distribution applications, Overcast [13] builds a single source multicast tree rooted at the source. The optimization goal of its "up/down" protocol is to provide each node in the tree with a high bandwidth path to the root. Yoid [10] forms a shared multicast spanning tree across the end hosts. Yoid also builds a mesh structure among members for routing stability. Similar to Yoid, Host Multicast Tree Protocol (HMTP) [29] builds a shared tree. When a new node joins, it probes the tree at each level, starting from the root, to find the nearest member node as a parent. CoopNet [17] focuses on using *multiple description coding* to handle flash crowd while reducing disruption. They rely on a centralized server for tree construction and maintenance. Application Level Multicast Infrastructure (ALMI) [18] uses a centralized approach to construct shared minimum spanning tree based on network measurements.

Narada [9] and Scattercast [8] build a mesh topology of all multicast members, and then compute a multicast spanning tree for each source. Both protocols periodically refresh the mesh to maintain the multicast topology.

ZIGZAG [26] proposes a peer-to-peer multicast for streaming media based on an administrative organization in which peers are organized in a multi-layer hierarchy of clusters. Given the administrative logical organization, the multicast tree is built using three given rules. The tree is periodically reconfigured to balance the load based on the node degree and capacity.

The Scalable Adaptive Randomized Overlay (SARO) protocol [14] has been recently proposed, built on top of a Random Subsets (*RanSub*) utility. The *RanSub* utility is used to deliver state information about a random subset of global nodes with each node selected in a subset with equal probability.

The RITA scheme differs from existing approaches in that previous P2P multicast systems embed the multicast trees in the overlay, and therefore are constrained by the logical structure of the overlay networks. In this aspect, RITA is similar to the ZIGZAG approach which decouples the administrative organization and the multicast data delivery paths. In addition, with the exception of OMNI, and ZIGZAG, none of the existing approaches take QoS into account in tree construction and maintenance for streaming media distribution. Unlike OMNI and ZIGZAG, the tree reconfiguration in our scheme is initiated by the receiver based on the application perceived QoS. The objectives of SARO are similar to that of RITA in terms of adapting quickly to network changes. However, RITA advocates the use of application QoS feedback to trigger tree transformations rather than use the periodic random subset distribution approach of SARO.

5. CONCLUDING REMARKS

This paper describes the RITA scheme for building an efficient overlay infrastructure for real-time multimedia applications, including live media distribution. Our goal is to balance the network-oriented goals of building an efficient multicast tree very quickly with the application-oriented goals of providing good QoS with minimal disruptions. Our initial results presented in this paper are encouraging. The quality of trees constructed using RITA are comparable (and often better than) to those constructed by a protocol like HMTP. Further, the tree construction latency is an order of magnitude lower. From the experiments conducted, the node switching times are under a second, making it possible to minimize application disruption. There are several open issues that need further investigation, which are briefly described below.

In our tree construction algorithm, a new node simply attaches to the closest node in the tree. The results in Section 3 show the importance of locating the closest node in multicast tree construction, and the effectiveness of our "landmark clustering + RTT measurement" scheme in finding the closest node. Our study shows that the performance of landmark clustering varies with topologies [28]. Consequently, we suggest several techniques to improve the accuracy of landmark clustering. The effectiveness of these schemes in the real Internet remains to be seen, and their ultimate limits due to the incomplete proximity information in landmark vectors need to be explored. Further, there are multiple approaches (e.g., IDMaps [11], King [12]) that could be used in estimating node proximity in the context of RITA and we plan to evaluate these in the future.

A cornerstone of our approach is that the tree reconfiguration is primarily initiated by the application client at the receiver when the perceptual application QoS (e.g., perceptual media quality) falls below a specific threshold. We realize that the translation between network QoS metrics and subjective perceptual media quality is non-trivial. We plan to leverage ongoing work in this regard [1]. Another alternative albeit intrusive method is for users to indicate their dissatisfaction with the deteriorating audio/video quality by pressing a button on the keyboard, which would then initiate the overlay reconfiguration.

Delivering real-time multimedia to clients is a complex process involving many factors such as caching, buffering, and transcoding. We are building a media service delivery infrastructure and will use real media applications to study the interplay of these factors. For instance, depending on the network dynamics, we may avoid frequent short-term overlay adaptation by a small increase of the client buffer and startup delay. When the network condition is degraded to the extent that adaptation is futile, degrading the quality by including a transcoding process (transparently) is perhaps the only option.

We intend to extend the performance evaluation in a number of directions. In the evaluation presented in this paper, node degree constraints are not considered. In reality, most nodes will allow only a limited number of nodes to attach to it in the multicast tree. The implication of this is that the constructed trees are deeper, causing end-to-end quality problems. As the size of the overlay multicast tree is increased, increasing the number of close by candidate nodes (C) to choose from produces more accurate node proximity results, which in turn improves the tree efficiency. Thus, automatically scaling the parameter C with the size of the overlay is one of our future work items.

6. ACKNOWLEDGMENTS

We thank Zhiheng Wang of the University of Michigan for his valuable contributions to this research, while he was an intern at HP Laboratories.

7. REFERENCES

[1] W. Ashmawi, R. Guerin, S. Wolf, and M. H. Pinson. On the impact of policing and rate guarantees in Diff-Serv networks: A video streaming application perspective. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 83–95, San Diego, CA, August 2001.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 205–220, Pittsburgh, PA, August 2002.

[3] S. Banerjee, J. Brassil, A. Dalal, S.-J. Lee, E. Perry, P. Sharma, and A. Thomas. CDNs for personal broadcasting and individualized reception. In *Proceedings of 7th International Workshop on Web Content Caching and Distribution (WCW)*, pages 279–284, Boulder, CO, August 2002.

[4] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, San Francisco, CA, April 2003.

[5] K. Calvert, M. Doar, and E. W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.

[6] M. Castro, P. Druschel, Y. Hu, and A. Rowstron. Exploiting network proximity in distributed hash tables. In *Proceedings of the International Workshop on Future Directions in Distributed Computing (FuDiCo)*, Bertinoro, Italy, June 2002.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, and A. I. T. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications*, 20(8):1489–1499, October 2002.

[8] Y. Chawathe. Scattercast: An adaptive broadcast distribution framework. *ACM Multimedia Systems Journal*, 2003.

[9] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471, October 2002.

[10] P. Francis. Yoid: Your Own Internet Distribution, March 2001. http://www.isi.edi/div7/yoid/.

[11] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, October 2001.

[12] K. P. Gummadi, S. Saroiu, and S. D. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Proceedings of the Second ACM SIGCOMM Internet Measurement Workshop (IMW)*, pages 5–18, Marseille, France, November 2002.

[13] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th USENIX Symposium on Operating System Design and Implementation (OSDI)*, San Diego, CA, October 2000.

[14] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. M. Vahdat. Using random subsets to build scalable network services. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems*, Seattle, WA, March 2003.

[15] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with delaunay triangulation overlays. *IEEE Journal on Selected Areas in Communications*, 20(8):1472–1488, October 2002.

[16] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, June 2002.

[17] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Miami, FL, May 2002.

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

[19] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 161–172, San Diego, CA, August 2001.

[20] S. Ratnaswamy, 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 (NGC)*, pages 14–29, London, UK, November 2001.

[21] S. Ratnaswamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, June 2002.

[22] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer sysstems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, Heidelberg, Germany, November 2001.

[23] S. Roy, B. Shen, V. Sundaram, and R. Kumar. Application level hand-off support for mobile media transcoding sessions. In *Proceedings of 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, Miami, FL, May 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 the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 149–160, San Diego, CA, August 2001.

[25] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: Offering QoS using Overlays. In *Proceedings of the ACM SIGCOMM First Workshop on Hot Topics in Networks (HotNets-I)*, Princeton, NJ, October 2002.

[26] D. A. Tran, K. A. Hua, and T. Do. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, San Francisco, CA, April 2003.

[27] Z. Xu, M. Mahalingam, and M. Karlsson. Turning heterogeneity into an advantage in overlay routing. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, San Francisco, CA, April 2003.

[28] Z. Xu, C. Tang, and Z. Zhang. Building topology-aware overlays using global soft-state. In *Proceedings of the 23rd IEEE International Conference on Distributed Computing Systems (ICDCS)*, Providence, RI, May 2003.

[29] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, New York, NY, June 2002.

[30] B. Y. Zhao, L. Huang, S. C. Rhea, J. Stribling, A. D. Joseph, and J. D. Kubiatowicz. Tapestry: A global-scale overlay for rapid service deployment. *IEEE Journal on Selected Areas in Communications*, 2003, to appear.

[31] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the 11th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, Port Jefferson, NY, June 2001.