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

Zhichen Xu, Chunqiang Tang, Zhiheng Wang,
Sujata Banerjee, Sung-Ju Lee
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto
HPL-2002-314 (R.1)
February 28th , 2003*

streaming media, overlay network, multicast

Application-level multicast networks overlaid over 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 an efficient overlay application layer multicast infrastructure for multimedia real-time applications based on a combination of landmark clustering and RTT measurements. 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 distinguished 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.

* Internal Accession Date Only Approved for External Publication
[1] Department of Computer Science, University of Rochester, NY
[2] Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI

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

Zhichen Xu, Chunqiang Tang*, Zhiheng Wang†, Sujata Banerjee, and Sung-Ju Lee
Internet Systems & Storage Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94304

ABSTRACT

Application-level multicast networks overlaid over 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 an efficient overlay application layer multicast infrastructure for multimedia real-time applications based on a combination of landmark clustering and RTT measurements. 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 distinguished 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.

1. MOTIVATION

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 contents. Internet users today can host web contents (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 access networks. 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 be deployable very quickly.

Scalable and efficient multicasting technology 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 perceptual 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 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 (e.g., [6]), while some consider overlay QoS (e.g., [19]). Some schemes such as Host Multicast Tree Protocol (HMTP) [22], 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). The global view is generated from landmark clustering [12]. Combining the landmark information with a small number of round-trip time (RTT) measurements to locate physically close-by neighbors, our approach provides very fast, high quality tree construction and adaptation. There are three key differences between our approach 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., RTT increases), our approach 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.

*Chunqiang Tang is with Department of Computer Science, University of Rochester, Rochester, NY.
†Zhiheng Wang is with Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI.

- Most of the schemes require several seconds to reconfigure or complete a join to the tree. Utilizing the landmark information stored in the DHT, our approach 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 HMTP, our algorithm speeds up the tree construction by a factor of up to eight, while producing trees that are up to 50% more efficient.

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

2. OUR APPROACH

The quality of a multicast tree, to a great extent, depends on how close its structure approximates that of the underlying 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 minimal spanning tree algorithm over the resulted graph. This approach however, is not practical for large trees due to the excessive measurements.

The fundamental idea of utilizing global information is still crucial. To maintain global state, we propose to select tree nodes to form a DHT to serve as a rendezvous plane to store information of nodes in the tree. In particular, we use the *landmark vectors* [12] 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 [21]. As a result, a node can find information of close-by nodes in an efficient and scalable way.

Our experiments show that if a new node can find the closest node in the tree and attach to it, the tree cost (defined as the aggregated weight of tree edges) is comparable to that of a minimal spanning tree with less than 30% overhead for trees with up to 4,096 nodes. To find the closest node, we propose a technique that combines landmark clustering and actual RTT measurements [21]. In our model, a new node always attaches to the closest node that is identified using the above technique.

Given these techniques, we describe a tree-adaptation algorithm that quickly responds to changing network conditions. 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 one [21]. 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. Suppose that a node's distance to landmark i is d_i and there are l landmarks. We use the *landmark vector* $< d_1, d_2, .., d_l >$ to approximates this node's physical position in the network. More sophisticated techniques to produce landmark information are also possible [12].

Each node uses its landmark vector as the key to store its *profile* in the DHT. This controlled placement of node profiles has the effect that information of nodes that are physically close are stored near 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 can accurately locate a close-by node [21].

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., 56 Kb/s dialup line, 1.5 Mb/s DSL or 100 Mb/s LAN), current and maximum fan out in the multicast tree[1], 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.

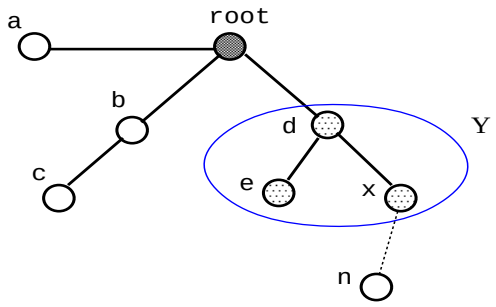

Figure 1: Tree construction algorithm.

2.2 Basic Tree Construction Algorithm

Our tree construction algorithm is illustrated 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 identify the node that is the closest. Let us denote the closest node as x.
3. Attaches to x as its child.

[1]The maximum fanout may be set to zero if the node is incapable of serving any other nodes (e.g., a node that has access to only a low speed dial-up network connection, etc.).

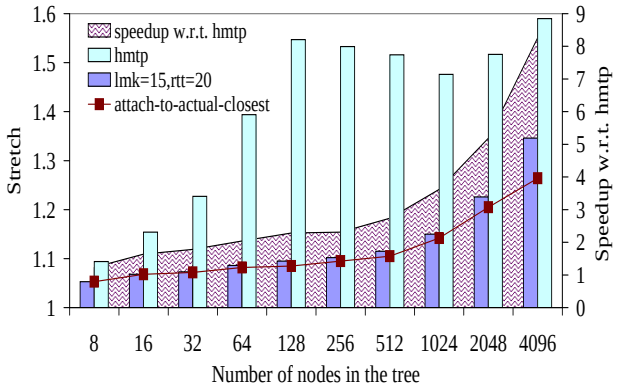

Figure 2: Tree quality and speedup with respect to HMTP.

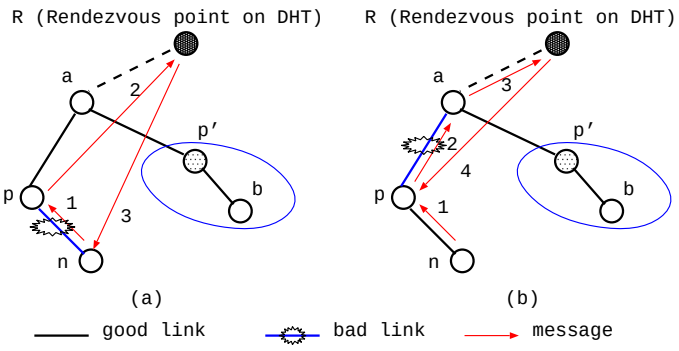

Figure 3: Tree adaptation algorithm.

This simple algorithm offers the following advantages: (i) Because RTT measurements to each node in Y are performed concurrently, a new node can quickly locate 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.

To evaluate the effectiveness of our algorithm, we compare the quality of the trees constructed using our algorithm with that of trees produced by protocols that probe the tree level-by-level (e.g., HMTP). We define *stretch* as the ratio of tree cost to that of a minimal spanning tree. 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 our algorithm.

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 low delay path starting from the root. Among all nodes on this path, the new node attaches to the closest one.

We have conducted preliminary experiments using a 10,000 node topology produced by GT-ITM [5]. Latencies in the topologies are set according to the following rules: 100ms for cross transit links, 20ms for links connecting nodes inside a transit, 5ms for links connecting a transit and a stub node, and 2ms for links connecting nodes inside a stub. We randomly select 15 nodes as landmark nodes.

Figure 2 shows the results. We also compare the performance of our trees with that of the cases where a new node locates the actual closest node (the curve labeled *attach-to-actual-closest*). We observe the following:

- In terms of tree construction time, our algorithm outperforms HMTP by a factor of up to eight.
- In terms of tree quality, our trees are up to 50% more efficient than those of HMTP and close to the optimal *attach-to-actual-closest* case. Comparing with the minimal spanning tree algorithm, our algorithm introduces less than 30% overhead for trees with up to 4,096 overlay 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 saving.

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 needs continuously adapt to these changing conditions and minimize any service disruption to the end users. This translates into finding the best place to perform the adaptation and minimizing the time for each repair. We assume that all nodes in the tree can monitor the quality of the link and translate this to user-perceived QoS (see Section 4).

We illustrate our tree adaptation algorithm in Figure 3. 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 responses, we have two cases as shown in Figure 3 (a) and 3 (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 rendezvous point R on the DHT, which will provide n with a set of new candidate parent nodes that are close to n judging from the landmark vectors. In Figure 3 (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 (a, p) in our example. 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 3 (b), because a is happy with the QoS it perceives, it directs the complaint to the rendezvous point R on the DHT, 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

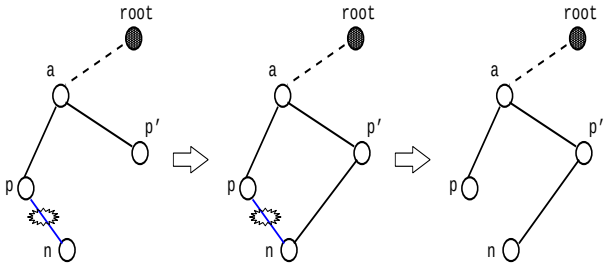

Figure 4: Multi-homed handoff process.

path, it starts the complaint process again. Note that routing loops are prevented in our scheme as each node records the path from the root to itself.

Our tree-adaptation algorithm minimizes the overall disruption by trying to locate the problematic link and asking 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 only requires the node incident to that link to attach to a new parent.

As shown in the figure, it typically takes three steps to obtain a set of parent candidates. Assume that the (n, p) and (n, p') distances are 20ms, and (p, R) and (n, R) are 100ms. Considering that routing in the DHT typically doubles the latency of IP routing [21], 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 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 high delay period inflicted by the handoff. To minimize disruption, we use multi-homing at the multicast overlay layer during the handoff period, similar to [18]. 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 4. 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 of time, certain links that lead to node n now may have to carry traffics 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. RELATED WORK

Several application-level multicast schemes achieve data distribution by *implicitly* building a multicast structure. For instance, Scribe [6] is a multicast infrastructure built on top of Pastry [17]. 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 [15] is extended for multicast in [16]. 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 [24] is an architecture built on top of Tapestry [23] and supports source-specific multicast. The NICE protocol [2] builds and maintains hierarchical topology of multicast members. The multicast routes are implicitly defined by the hierarchy structure. A protocol that uses a Delaunay triangulation as an overlay network topology is proposed in [11]. 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 [10] 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 [9] 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) [22] 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 [13] 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) [14] uses a centralized approach to construct shared minimum spanning tree based on network measurements.

Narada [8] and Scattercast [7] 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.

Our 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 P2P networks. In addition, with the exception of OMNI [4], none of the existing approaches take QoS into account in tree construction and maintenance. Unlike OMNI, the tree reconfiguration in our scheme is initiated by the receiver based on the application perceived QoS.

4. DISCUSSION

This paper describes our first step towards building an efficient overlay infrastructure for real-time multimedia 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. There are several open issues that need further investigation.

A cornerstone of our approach is that the tree reconfiguration is primarily initiated by the application client at the receiver when the 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 (e.g., [1]) in this regard. 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.

The results in Figure 2 show the importance of locating the closest node in multicast tree construction, and the effectiveness of our "landmark clustering + RTT" scheme in finding the closest node. Our study shows that the performance of landmark clustering varies with topologies [21]. Consequently, we suggest several techniques to improve the accuracy of landmark clustering. In particular, our initial results indicate that using artificial neural network to estimate Internet distance may produce good accuracy. Their effectiveness 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.

In our tree construction algorithm, a new node simply attaches to the closest node in the tree. Better performance may be achieved by replacing some links in the tree with links to the new node. We have devised a tree maintenance algorithm that utilizes local information to compute a local minimal spanning tree [20]. We have yet to evaluate the effectiveness of our algorithm using real topology, latency and dynamics of the Internet.

Delivering real-time multimedia to clients is a complex process involving many factors such as caching, buffering, and transcoding. We are building a media 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.

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 ACM SIGCOMM*, August 2001.

[2] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceedings of ACM SIGCOMM*, 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 WCW'02*, 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 Infocom*, April 2003.

[5] K. Calvert, M. Doar, and E. W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, June 1997.

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

[7] Y. Chawathe. Scattercast: An adaptive broadcast distribution framework. *ACM Multimedia Systems Journal*, 2002.

[8] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE JSAC*, 20(8), October 2002.

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

[10] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of USENIX OSDI*, October 2000.

[11] J. Liebeherr, M. Nahas, and W. Si. Application-layer multicasting with delaunay triangulation overlays. *IEEE JSAC*, 20(8), October 2002.

[12] T. S. E. Ng and H. Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings of IEEE Infocom*, June 2002.

[13] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of ACM NOSSDAV*, May 2002.

[14] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proceedings of USENIX USITS 2001*, San Francisco, CA, USA, March 2001.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM*, August 2001.

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

[17] 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 Middleware*, November 2001.

[18] S. Roy, B. Shen, V. Sundaram, and R. Kumar. Application level hand-off support for mobile media transcoding sessions. In *Proceedings of ACM NOSSDAV*, May 2002.

[19] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: Offering QoS using Overlays. In *Proceedings of HotNets-I*, October 2002.

[20] C. Tang and Z. Xu. pFilter: Global information filtering and dissemination. In *Proceedings of IEEE FTDCS*, May 2003.

[21] Z. Xu, C. Tang, and Z. Zhang. Building topology-aware overlays using global soft-state. In *Proceedings of IEEE ICDCS*, May 2003.

[22] B. Zhang, S. Jamin, and L. Zhang. Host multicast: A framework for delivering multicast to end users. In *Proceedings of IEEE Infocom*, June 2002.

[23] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.

[24] 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 ACM NOSSDAV*, June 2001.