Resilience in Overlay Multicast Protocols

Stefan Birrer Fabián E. Bustamante
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, IL 60208, USA
Email: {sbirrer,fabianb}@cs.northwestern.edu

Abstract— **One of the most important challenges of self-organized, overlay systems for large-scale group communication lies in these systems ability to handle the high degree of transiency inherent to their environment. While a number of resilient protocols and techniques have been recently proposed, achieving high delivery ratios without sacrificing end-to-end latencies or incurring significant additional costs has proven to be a difficult task. In this paper we review some of these approaches and experimentally evaluate their effectiveness by contrasting their performance and associated cost through simulation and wide-area experimentation.**

I. Introduction

Deployment issues with IP Multicast [22], [23] have motivated recent work on alternate, peer-to-peer approaches for supporting group communication applications over the Internet [19], [29], [26], [37], [17], [10], [11], [4], [16], [39], [50], [35], [45]. In this self-adaptive, application-layer approach, participating peers configure themselves as an overlay topology for data delivery. The topology is an overlay in that each edge corresponds to a unicast path between two end systems in the underlying Internet. All multicast-related functionality is implemented at the end systems instead of at the routers, and the goal of the multicast protocol is to construct and maintain an efficient overlay for data transmission.

As multicast functionality is pushed to autonomous, unpredictable end hosts, however, significant performance loss can result from their higher degree of transiency when compared to routers [6]. A good indicator of node's transiency is the peers' *median session time*, where *session time* is defined as the time between when a peer joins and leaves the network. Measurement studies of widely used P2P systems have reported median session times ranging from 90 to one minute [12], [18], [42], [27]. Although collected mostly from file-sharing applications, these measurements give us an idea of the high level of transiency that can be expected in large peer populations. Consequently, a number of techniques [5], [14], [10] have recently been proposed to improve overlays' resilience by exploiting path diversity [1], [43] and minimizing node dependencies [2].

Delivering high application performance at relatively low costs and under high degree of transiency has proven to be a difficult task [40], [18], [33]. Each of the proposed resilient techniques comes with a different trade-off in terms of delivery ratio, end-to-end latency and additional network traffic. To help guide further research, this paper reviews some of these approaches and evaluates their effectiveness by contrasting the performance and associated cost of representative protocols through simulation and wide-area experimentation. We restrict our comparison to tree-, stream-based protocols, where timely data delivery is a key requirement. The evaluation of resilient protocols for bulk data dissemination [30], [20], [36] is beyond the scope of this paper.

Our results show that while all resilient schemes have their particular merits, a combination of multiple techniques may offer the best cost/benefit trade-off. In particular, we found that combining a technique that improves tree resilience through path diversity, i.e. in-tree redundancy, with a multiple-tree approach yields excellent delivery ratios under a large range of peer transiency. In bandwidth-limited environments, multiple trees improve both delivery ratios and delivery latencies as they avoid bottlenecks in the distribution topology thanks to a more even distribution of forwarding load than conventional, single-tree approaches.

The remainder of the paper is organized as follows. Section II provides some useful background on overlay multicast and Section III overviews some of the techniques proposed for improving resilience, illustrating them with a more detailed description of a representative protocol. We outline our evaluation setting and report experimental results from simulations and wide-area experimentation in Sections IV and V. Section VI describes related work and Section VII concludes.

II. Overlay Multicast

Multicast is an efficient mechanism to support group communication applications such as audio and video conferencing, multi-party games and content distribution. It decouples the size of the receiver set from the amount of state kept at any single node and potentially avoids redundant communication in the network. Partially in response to the deployment issues of network-layer multicast [22], [23], a number of protocols recently proposed adopt an alternative peer-to-peer or overlay approach, with all multicast related functionality implemented exclusively at end-systems. Peers in most overlay multicast protocols self-organize in two topologies: a control overlay for group membership related tasks, and a delivery tree for data forwarding. Based on the sequence adopted in the construction of these topologies and their structuring approach, protocols can be classified as – tree-first, mesh-first, DHT-first and implicit [3]. In a *tree-first* approach [26], [29], [37], peers directly build the data delivery tree by selecting their parents from among known peers. Additional links are later added to define the control topology. With a *mesh-first* approach, (reverse) shortest path spanning trees, rooted at any peer, are defined over a more densely connected graph (mesh) [19]. Under the *DHT-first* approach, peers organize themselves into a well-defined geometrical structure over which a data delivery topology is built [41], [39]. Last, under the *implicit* approach, peers create only a control topology, while the data delivery tree is implicitly defined by packet forwarding rules based on the initial control tree. Resilient techniques for overlay multicast have been targeted mainly at the latter two, the DHT-first and the implicit approach.

Scribe [16] is probably one of the best known *DHT-first* overlay multicast protocols. It builds upon Pastry [41], a structured (DHT) P2P overlay. Every peer in Pastry [41] is assigned a randomly unique node identifier (nodeId), uniformly distributed in the circular identifier space formed by all possible identifiers. Given a message and an associated key, Pastry routes the message to the node with nodeId numerically closest to the message key. In order to route messages, each node maintains a routing table, where the node associated with each entry in row r of the routing table shares the first r digits with the local nodeId. A message is routed to a node whose nodeId shares a prefix with the message key of at least one digit longer than the current node's nodeId or, if no such node exists, is numerically closer to the key. Additionally, each node maintains a leaf set and a set of neighboring nodes. The leaf set contains nodes which are numerically closest to the local node's nodeId, whereas the neighborhood set consists of nodes which are closest based on a proximity metric. In order to provide routing through the network, the Pastry overlay requires a consistent mapping from keys to overlay nodes and depends on persistent intermediate nodes for successful message delivery. Scribe [16] group communication builds upon Pastry to support applications that demand large number of multicast groups. Each of these multicast groups may consist of a subset of all nodes in the Pastry network. Every multicast group in Scribe is assigned a random ID (known as the *topicId*), and the multicast tree for the group is formed by the union of Pastry routes from each group member to the root, identified by the topicId. Messages are then multicast from the root using reverse path forwarding [21]. Most recent implementations of Scribe and Pastry incorporate the suggestions in [40], in an attempt to minimize the impact of churn on DHT-based overlays.

Nice [4], on the other hand, belongs to a class of protocols generally known as *performance-centric*, in which the primary consideration for adding a link to the overlay topology is performance. This is in contrast to DHT-based systems such as Scribe, where the focus is on maintaining a structure based on a virtual-id space [7]. Participating peers are organized into clusters based on end-to-end latency, with every peer being a member of a cluster at the lowest layer. Clusters vary in size between d and $3d - 1$, where d is a constant known as *degree*. Each of these clusters selects a *leader* that has minimum maximum distance to all other cluster's members. The leader of a cluster becomes a member of the immediately superior layer. The process is repeated, with all peers in a layer grouped into clusters and new leaders elected and promoted to participate in the next higher layer. Hence peers can lead more than one cluster in successive layers of this logical hierarchy. Nice creates this hierarchically-connected control topology, but leaves the delivery path *implicitly* defined by the packet forwarding rules. Nice has been thoroughly evaluated and shown to perform well in a variety of scenarios [4], [10], [11].

III. Resilient Approaches to Overlay Multicast

Given the impact of node transiency on the performance of overlay multicast protocols, a number of techniques [5], [14], [10], [9] have been recently proposed aimed at improving overlay resilience by exploiting path diversity and minimizing node dependencies. The different techniques can be coarsely classified as: cross-link, in-tree, and multiple-tree redundancy. *Cross-link* and *in-tree* redundancy improve resilience by adding extra links to the original, single multicast tree [10], [49], [45], [5]. *Multiple-tree redundancy* creates several overlapping trees over which stripes of the multicast stream are forwarded [14], [35], [9]. Figure 1 illustrates each of these classes. This section reviews each class in the context of concrete, representative protocols.

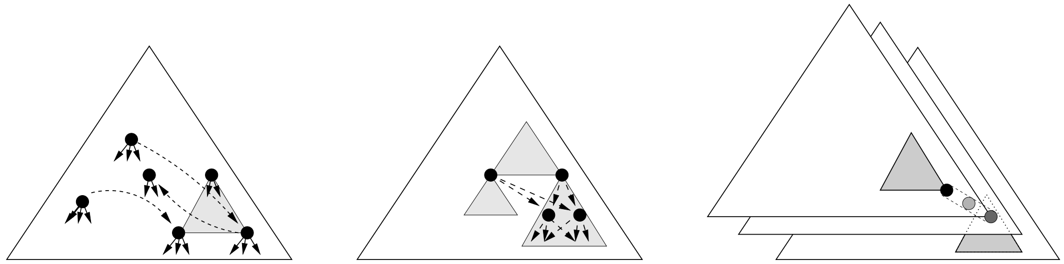
A. Cross-Link Redundancy

Probabilistic Resilient Multicast (PRM) is a general scheme to improve the resilience of overlay multicast [5] through randomized forwarding. PRM adopts triggered negative acknowledgment, as well as a *cross-link redundancy* approach, forwarding an additional fraction of the stream over extra cross-cutting links connecting random peers in the tree [5]. The PRM scheme is defined as $PRM(n, p)$ where n is the number of random peering partners and p the forwarding probability. In PRM, a node continuously discovers random session members and forwards every received data packet to any of those members with a specified probability. These random packets help detect and recover from temporary tree partitions. Under stable conditions, however, such an approach introduces $n{\cdot}p$ duplicate packets that "waste" part of the peers' available bandwidth.

B. In-Tree Redundancy

Nemo is a performance-centric, overlay multicast protocol targeted at large-scale, heterogeneous and highly-dynamic environments. It adopts an *implicit* approach to overlay multicast, organizing peers into a logical hierarchy over which the data delivery network is defined, implicitly, by a set of forwarding rules. Similarly to Nice, Nemo organizes nodes into clusters based on network proximity, [1] with every peer being a member of a cluster at the lowest layer. Each of these clusters selects a *leader* that becomes a member of the immediately higher layer. The process is repeated, with all peers in a layer being grouped into clusters, crew members selected, and leaders promoted to participate in the next higher layer.

[1] Although other factors such as bandwidth [46], [19] and expected peer lifetime [12] can be easily incorporated.



(a) Cross-Link Redundancy. (b) In-Tree Redundancy. (c) Multiple-Tree Redundancy.

Fig. 1: Common Resilience Techniques. The *cross-link* and *in-tree redundancy* approaches improve resilience by adding extra links to a single multicast tree. *Cross-link redundancy* forwards an additional fraction of the stream over extra cross-cutting links connecting random peers in the tree, while *in-tree* redundancy creates alternative paths only around nodes with forwarding responsibility. The *multiple-tree* redundancy approach creates several overlapping trees over which stripes of the multicast stream are forwarded.

Nemo achieves resilience through the introduction of *co-leaders*, alternative leaders that share the forwarding load of clusters' leaders, and their responsibility for triggered negative acknowledgments. Co-leaders improve the resilience of multicast groups by avoiding dependencies on single nodes and providing alternative paths for data forwarding. Thus, Nemo's *in-tree redundancy* approach creates alternative paths only around nodes with forwarding responsibility. As co-leaders share the message-forwarding load with leaders, they also help reduce the forwarding bandwidth demand of cluster leaders improving overall system's scalability. In addition, Nemo reduces overlay maintenance cost through the adoption of a probabilistic approach where operations are executed with some probability or, alternatively, deferred to the next interval. In the presence of high churn, many of these operations can be completely avoided as follow-up changes may revert a previously triggering situation.

C. Multiple-Tree Redundancy

In conventional tree-based multicast systems, a relatively small set of nodes is responsible for forwarding all multicast messages. This may introduce bottlenecks in the forwarding topology as the induced load may easily overwhelm a specific end host. To address this problem, Castro et al. [14] propose the use of multiple interior-node disjoint trees over which stripes of the data stream are disseminated. By forwarding different stripes over each tree and making each peer an interior node in at least one tree, the *multiple-tree redundancy* approach distributes the forwarding load more equally among the participating peers. To efficiently create and maintain this forest, the **SplitStream** protocol leverages the inherent properties of the DHT routing model, building on Scribe [16] to provide a relatively simple and efficient method for forest construction that neither requires of costly network monitoring nor depends on a centralized coordinator.

Targeted at cooperative, group-communication applications in heterogeneous environments, **Magellan** adopts the same *multiple-tree redundancy* approach, but building instead on a forest of performance-centric trees [10], [4]. Magellan ensures that every participating peer contributes resources to at least one tree in the forest and that all trees have a set of assigned peers to serve as their interior nodes. The set of interior nodes to a tree is made of *primary peers*, i.e. peers for which the tree is their *primary tree*, and additional *secondary peers*. If a tree's set of primary peers does not collectively have the required resources to support the tree's stripe, e.g. due to peers with low bandwidth capacity, Magellan assigns additional secondary peers with spare resources as needed. Thus, Magellan guarantees that no tree will run out of forwarding capacity before the full system is saturated while still supporting the participation of non-contributors in the system. By relying on balanced, multicast trees, Magellan reduces the total end-to-end hop distance in the distribution topology, thus lessening the tree vulnerability to node failures and minimizing performance overhead. For detecting peer failures/departures and repairing the topology, Magellan relies on an efficient, per-tree maintenance protocol. In addition to the frequency of interruptions a node experiences, the second factor determining application performance is the efficiency of the detection/repair protocol. All multicast messages in Magellan are uniquely identified and lost messages are recovered via *lateral error recovery*, i.e. recovered from any of the trees, not only the forwarding one [47]. Magellan was originally implemented using Nemo, inheriting the latter's churn-resilience properties. To understand the benefits of each *multiple-tree* and *in-tree redundancy* to performance-centric multicast protocols, for our evaluation we implemented a variant of Magellan that relies on Nice [4] as its underlying tree construction protocol.

IV. Evaluation

One of the most important challenges of self-organized, overlay systems for large-scale group communication lies in these systems ability to efficiently handle the high degree of churn inherent to their environment. Measurement studies have reported median session as short as one minute [12], [18], [42], [27]. Effectively delivering high application performance in an efficient manner under such levels of transiency has proven to be a difficult task [40], [18], [33]. Our evaluation aims at determining the trade-off brought in by each of the proposed resilient techniques (Section III) in terms of delivery ratio, response time and additional network traffic. To this end we employ two non-resilient protocols, as baselines, and five resilient protocols implementing the different techniques described.

We carried out our evaluation both through simulation and Internet experimentation in the PlanetLab wide-area testbed [38]. We used our own implementation of all the evaluated protocols. Each protocol implementation closely follows the descriptions from the corresponding literature, incorporating most published improvements, and its performance is consistent with what has been previously reported.

The remainder of this section describes our evaluation setup, provides some details on the implementations of the compared protocols and the metrics employed in our evaluation. Section V presents results from our simulation and Internet experiments.

A. Evaluation Setup

Our simulation experiments are conducted using SPANS a locally-written, packet-level, event-based simulator. For wide-area experimentation we employed 100 PlanetLab [38] nodes.

We ran simulations using GridG [31], [32] topologies with 8,115 nodes, and a multicast group of 256 and 512 members. GridG leverages Tiers [24], [13] to generate a three-tier hierarchical network structure, before applying a power law enforcing algorithm that preserves the hierarchical structure. Multicast members are randomly attached to nodes, and a random delay between 0.1 and 80 ms is assigned to every link. Each end host uses per-connection buffers, dropping data packets first in the presence of congestion. We chose not to model bandwidth to avoid side effects due to control traffic competing for available bandwidth [14].

Each simulation experiment lasts 40 minutes of simulation time. All peers join the multicast group by contacting the rendezvous point at uniformly distributed, random times during the first 600 seconds of the simulation. The multicast session is enabled after 20 minutes. Warmup time is set to 30 minutes for all protocols to allow sufficient time to adjust to the topology under load. This time is omitted from the figures. Starting at 20 minutes and lasting to the end of the simulation, each simulation run has a membership changing phase. During this phase the evaluated protocols are exercised with end system failures. Node failures are independent and their time is sampled from an exponential distribution with mean, *Mean Time To Failure*, varying from 5 to 120 min. [48], [27]. By exploring a wide-range of MTTF, we avoid biases toward any particular deployment environment [28]. Failed nodes rejoin shortly after, with a delay sampled from an exponential distribution with mean, *Mean Time To Repair*, set to $\frac{1}{6}$ of MTTF. Setting the MTTR as a fraction of MTTF assures that the average online population is constant at different failure rates. Note that node departures and re-joins commonly require a number of expensive reorganization procedures. The alternative approach of immediately replacing every departing node potentially underestimates the impact of transiency as it may fail to factor in the cost of these repair operations.

For our wide-area experiments, we employ a network of 100 end hosts. The order of the protocol setups is randomly chosen for each experiment. At the beginning of every run we start one client per host and select the least loaded nodes to participate in the run. The experiment procedure is identical to the one employed for simulations. To estimate the end-to-end delay, we make use of a global time server. Every peer estimates the difference of its local time to the time at the server. The algorithm is inspired by [34] and leads to sufficient accuracy for our application.

In all experiments, we model a single-source multicast stream to a group of nodes. The source sends constant bit rate (CBR) traffic of 1,000 Byte packet payload at a rate of 10 packets per second.

B. Details on Protocol Implementations

As previously mentioned, we used our own implementation of all the evaluated protocols. These implementations, as well as the values assigned to their configuration parameters, follow closely the descriptions from the corresponding literature [4], [5], [10], [9], [41], [16], [14]. We have made them available to the community from our research group's resource page. [2]

For Nice [4] and Nice-PRM [5], the cluster degree, k, was set to three (3). We used PRM with three random peers chosen by each node, and with two percent forwarding probability. Evaluation results with other, less optimal, forwarding probabilities were reported in [10]. For Nemo, the cluster degree k and the crew size were set to three. The grace period was set to 15 seconds for Nice, Nice-PRM, Nemo and Magellan.

For Scribe and SplitStream, [3] we employ a leaf-set maintenance interval of 15 seconds and a route-set maintenance interval of 1,200 seconds. We opted for this configuration with the maintenance interval set to values four times lower than those in [14], to give SplitStream the same ability to detect failures as Nice and Nemo. The outdegree for SplitStream nodes is unlimited, yielding maximal performance for the unlimited bandwidth scenario [14]. In addition to the performance-optimized variant of SplitStream evaluated in this paper, the protocol can also be used with perfect fairness in mind. With this configuration, each peer contributes bandwidth resources corresponding to one full-rate split. Enforcing such tight outdegree requirements, however, results on deep delivery trees with high latencies. As we focused our evaluation on streaming media with low latency requirements, the evaluation of the fairness-optimized SplitStream variant is outside the

[2] AquaLab: *http://www.aqualab.cs.northwestern.edu*

[3] For the evaluation of Scribe and SplitStream we employ NUScribe and NUSplitStream, our own implementations of these protocols. NUScribe builds on top of NUPastry and thus leverages its churn-optimized algorithms [33], [40].

scope of this paper. The interested reader is referred to the detailed analysis in the original publication [14].

We evaluate SplitStream and Magellan with 2, 4, 8 and 16 stripes (s) (trees), where each of the stripes is responsible forwarding $\frac{1}{s}$ of the full rate stream to each client. Consequently, an outdegree of one in a SplitStream tree corresponds to a physical outdegree of $\frac{1}{s}$.

For the wide-area implementation, we employed UDP with TCP-friendly rate control [25]. We limited the number of retransmissions to ten attempts for heartbeats and five for all other control traffic. Data communication did not employ retransmission.

C. Evaluation Criteria

We used several metrics to evaluate the different resilient overlay multicast protocols, capturing both delivered performance to the application and protocol overhead.

- **Delivery Ratio.** Ratio of subscribers which have received a packet within a fixed time window. Disabled receivers are not accounted for.
- **Delivery Latency.** End-to-end delay (including retransmission time) from source to receivers, as seen by the application. This includes path latencies along the overlay hops, as well as queuing delay and processing overhead at peers along the path.
- **Physical Outdegree.** The physical outdegree is the fanout of a node and indicates the number of full rate streams a peer has to support in the distribution topology. Whenever multicast data is split into smaller messages and peers only forward part of them (e.g. SplitStream), the effective outdegree can be a non-integer value. One could consider using the number of successors to define the outdegree of nodes in a given protocol. This metric, however, could not be used to compare different protocols, as each employs different forwarding rules with some forwarding only part of the total data stream from a given node to its successors. The physical outdegree, on the other hand, serves as a good indicator of nodes' total bandwidth contributions.
- **Overhead.** Total control traffic in the system, in packets per peer, per second, during the observation interval. We measure the total control traffic by accounting packets at the end hosts.

V. Evaluation Results

The effectiveness of a group communication protocol can be measured in terms of delivery ratio, i.e. the ratio of subscribers that has received packets within a given time window, and the end-to-end delay for this delivery as seen by the application. The protocol's efficiency can be measured, on the other hand, in terms of the add-on overhead for a given delivery-ratio and latency.

The following subsections present results from our simulation and wide-area experimentation. Unless otherwise noted,

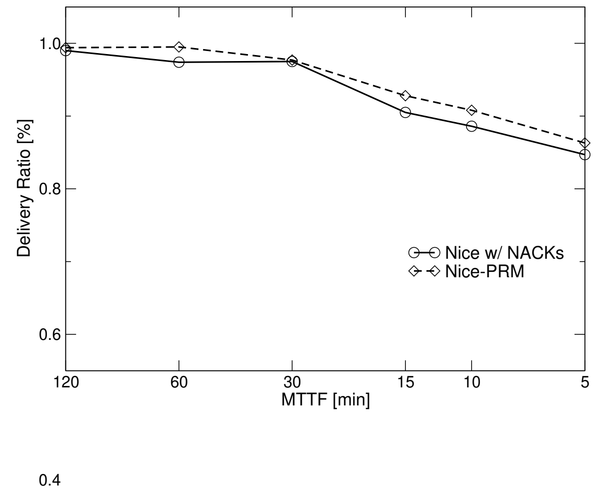


Fig. 2: Delivery ratio (256 end hosts, unlimited bandwidth). The x-axis shows the degree of churn specified in terms of MTTF (lower the MTTF means higher churn). Degree of churn increases toward the right side of the axis. The x-axis crosses at 55% delivery ratio. The plot shows how PRM increase the average delivery ratio for different degrees of churn.

all reported results are based on five independent runs per protocol and setup, for both the simulation and wide-area experiments.

A. Cross-link Redundancy

Tree-based protocols organize participating peers into a logical or physical tree over which the multicast data is distributed. Trees are highly dependent on the reliability of non-leaf nodes as their failure may result in temporary tree partitions. Cross-link redundancy addresses this issue by adding random links that improve the overlay robustness to churn. Figure 2 illustrates the delivery ratios achieved under different degrees of transiency by Nice, a tree-based protocol, and by the same protocol now *enhanced* with cross-links, Nice-PRM. Note, however, that in this case Nice is configured with NACKs to isolate the contribution of cross-links to the resilience of the protocol. The standard deviation of the measurements range from 0.5% to 2% with various degrees of churn. Cross-links provide a relatively minor increase to delivery ratio with this parameter setting and at different levels of churn.

Randomly forwarding data packets over cross-link potentially creates duplicate packets at some nodes. Table I illustrates the data overhead for Nice and Nice-PRM at two failure rates. With MTTF of 2 hours, PRM suffers from about 2 % extra data packets which corresponds to its configured forwarding probability. As one increases the failure rate, some of the randomly forwarded packets help restore missing packets and, consequently, the relative overhead of PRM decreases. In general, each overlay protocol uses a number of control messages to maintain and optimize the control-topology. Nice,

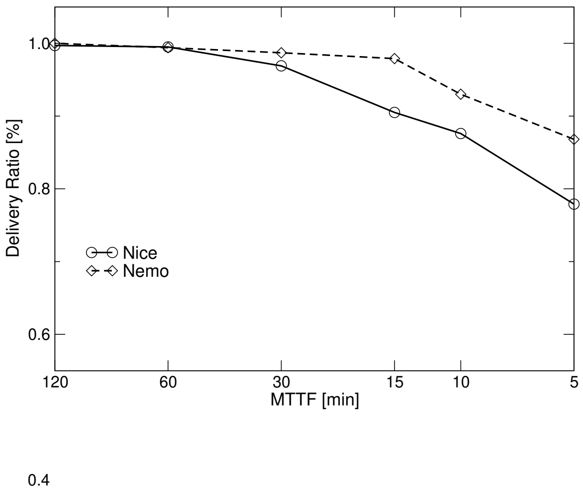


Fig. 3: Delivery ratio (512 end hosts, unlimited bandwidth). The x-axis shows the degree of churn specified in terms of MTTF (lower the MTTF means higher churn). Degree of churn increases toward the right side of the axis. The x-axis crosses at 55% delivery ratio. The graph shows how in-tree redundancy significantly improves delivery ratio at high degree of churn.

Protocol	Duplicate Packets [%]	Control Packets [/sec]
MTTF=120 min, MTTR=20 min		
Nice	0.1	1.64
Nice-PRM	2.1	3.89
Nemo	0.4	1.67
MTTF=30 min, MTTR=5 min		
Nice	0.2	1.57
Nice-PRM	2.3	3.63
Nemo	3.1	1.67

TABLE I: Overhead (512 end hosts, unlimited bandwidth). The overhead in data communication is reflected in the number of duplicated packets. We see that PRM adds a constant rate of duplicates even with a very low failure rate. In contrast, Nice and Nemo have a very low number of duplicates with low churn rates. The generic overhead required to maintain the tree topology is reflected in the number of control packets send by Nice. Additionally, Nemo adds a few packets per second to distribute information about its crew members. PRM, on the other hand, uses control messages to discover and maintain the list of random forwarding peers.

Protocol	Delivery Latency [s]	Delivery Ratio [%]
MTTF=120 min, MTTR=20 min		
Nice	0.085	0.997
Nemo	0.093	1.000
MTTF=30 min, MTTR=5 min		
Nice	0.141	0.969
Nemo	0.161	0.987

TABLE II: Delivery Latency (512 end hosts, unlimited bandwidth). While in-tree redundancy potentially results in additional delivery latency, we see that this effect is neglectable considering the higher delivery ratio of Nemo when compared to Nice.

in particular, uses about 1.6 packets per peer, per second to manage its topology. In addition to this general overhead, PRM uses control messages to discover and maintain the list of the random forwarding peers.

B. In-Tree Redundancy

A number of reinforced tree structures have been proposed to avoid the potential overhead of cross link redundancy while further reducing the dependency on single nodes. By introducing alternate forwarding path, in-tree redundancy increases overall resilience by lessening the impact of a specific node's departure has on the overall message delivery topology. Figure 3 shows the delivery ratios of a tree-based protocol with in-tree redundancy, Nemo, and its corresponding non-resilient tree-based protocol, Nice. The figure plots the delivery ratio for increasing degrees of churn. The outermost left value of the x-axis corresponds to a MTTF of 2 hours whereas the outermost right one to a MTTF of 5 minutes. The standard deviation observed reaches 0.5% for very low failure rates, about 4% for MTTF of 30 minutes and up to 9% for very high failure rates. The figure shows that in-tree redundancy substantially increases the delivery ratio under churn.

Some of the alternate delivery paths introduced by in-tree redundancy could, on the other hand, result in additional delays when compared to the best available path, thus negatively affecting end-to-end latency. Table II shows that this potential overhead is small especially considering the higher delivery ratio of in-tree redundancy when compared to its conventional, tree-based counterpart. In the presence of failures, lost packets help reduce the overall delivery latency as packets delivered over more hops are more likely to be dropped than packets delivered closer to the source.

Overlay protocols commonly incur some default control overhead necessary to maintain their distribution topologies. In addition, maintaining and using multiple paths for resilience requires additional control traffic and could result on higher number of duplicate data packets. Table I also shows that Nemo's and Nice's overheads in terms of extra data (duplicates) and control packets, at low churn rates, are comparable. The maintenance of Nemo's in-tree redundancy additionally introduce a small relative increase in control packets when compared to the baseline protocol Nice.

C. Multiple-Tree Redundancy

Using multiple disjoint trees reduces the bandwidth requirements of the participants and yields potentially flatter distribution trees than single-tree approaches, reducing the overlay's dependency on any single node. Figure 4 shows the latter benefits as NUSplitStream with 2 to 16 trees yields a significantly better delivery ratio under churn than NUScribe. The standard deviation only exceeds 2% for most of the single-tree measurements and, only slightly, for very high failure rates with 2, 4 and 8 trees. The highest observed standard deviation is 4.5% with MTTF of 5 minutes and using a single-

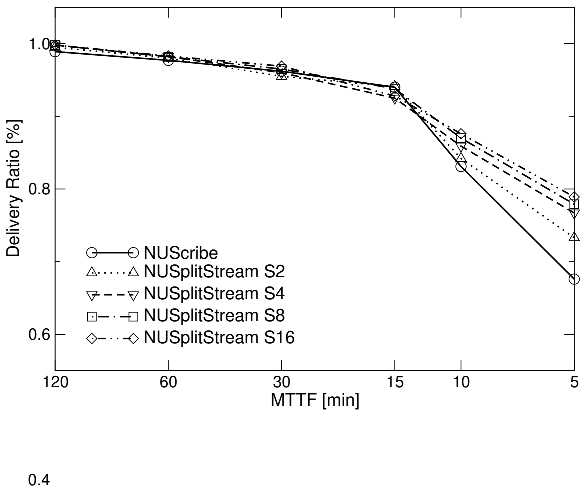


Fig. 4: Delivery ratio (512 end hosts, unlimited bandwidth). The x-axis shows the degree of churn specified in terms of MTTF (lower the MTTF means higher churn). Degree of churn increases toward the right side of the axis. The x-axis crosses at 55% delivery ratio. Multiple-tree redundancy helps increasing resilience at very high churn rates.

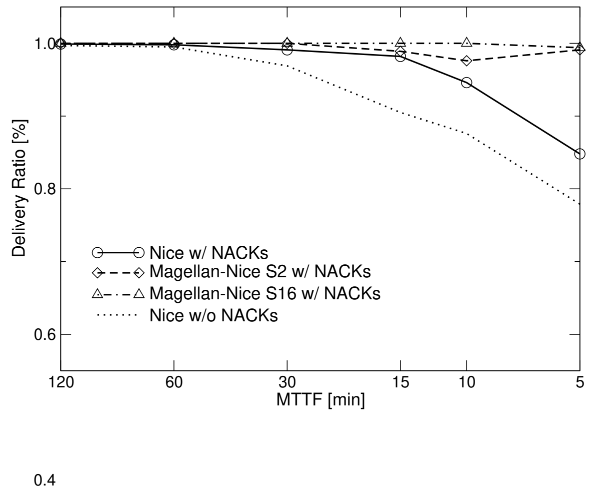


Fig. 5: Delivery ratio (512 end hosts, unlimited bandwidth). Lateral-error recovery, as introduced here by Magellan, has a significant impact on the resilience of multiple-tree protocols.

Protocol	Delivery Ratio [%]
Nice	0.90
Magellan (Nice) S4	0.94

TABLE III: Delivery ratio (100 end hosts, wide-area, MTTF=10 min, MTTR=2 min). Magellan uses 4 Nice trees without lateral error recovery for this experiment. We see that using multiple trees increase the overall delivery ratio.

Protocol	Duplicate Packets [%]	Control Packets [/sec]
MTTF=120 min, MTTR=20 min		
NUScribe	0.0	1.01
NUSplitStream S2	0.0	1.02
NUSplitStream S16	0.0	1.11
MTTF=30 min, MTTR=5 min		
NUScribe	0.1	1.22
NUSplitStream S2	0.1	1.25
NUSplitStream S16	0.3	1.56

TABLE IV: Overhead (512 end hosts, unlimited bandwidth).

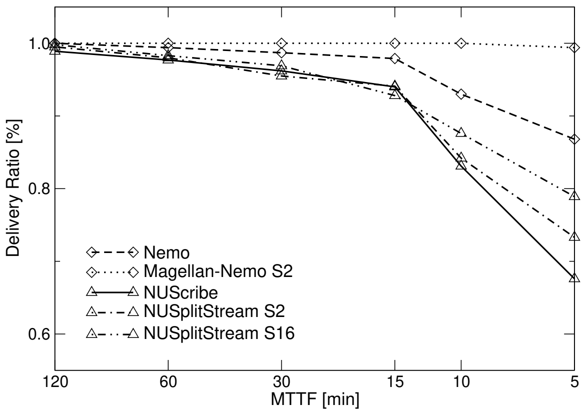
tree approach. The benefits become more clear as the level of churn exceeds the maintenance rate, defined in terms of the maintenance interval (set at 20 min).

In wide-area environments, multiple trees help increase the resilience of the distribution topology by increasing path diversity. Table III summarizes the delivery ratio of Nice and Magellan-Nice with four trees without lateral error recovery. Using multiple trees increases the delivery ratio by 4% due, in part, to a lower packet loss rate resulting from a better distribution of the forwarding load.

By combining multiple trees with lateral error recovery, peers can now restore missing packets from other trees (i.e. not only the forwarding one) and thus better overcome a temporary delivery outage in any one tree. Figure 5 shows the advantage of using multiple trees with lateral error recovery. For a MTTF of 30 minutes, we observe a standard deviation of 4.1% for Nice without NACKs, 0.2% for Magellan-Nice S2 with NACKs, 0.0% for Magellan-Nice S16, and 0.9% for Nice with NACKs. The maximal registered standard deviation is 7.6% for Nice with NACKs and a MTTF of 5 minutes. Magellan-Nice shows a substantially higher delivery, ratio under different levels of transiency, than the alternative protocols. When contrasted with Figure 4, we can see the significant contribution of lateral error recovery to a protocol's resilience to churn.

As Table IV illustrates, maintaining multiple trees clearly results in an increase on maintenance traffic, directly related to the number of trees employed. Combining multiple trees with other resilient techniques allows us to reduce the number of trees, and so the associated overhead, needed to achieve a given delivery ratio under churn. Figure 6 illustrates the benefits of combining in-tree and multiple-tree redundancy. The observed standard deviation for in-tree redundancy and two trees is smaller than 0.3% for all failure rates except for 5 minutes MTTF where the standard deviation is 1.9%. Two instances of a protocol with in-tree redundancy using lateral error recovery are sufficient to provide near perfect delivery ratios under the highest evaluated degree of churn.

Building multiple disjoint trees restricts the options for internal nodes and potentially results in increased delivery latencies, an important metric for a wide range of applications such as multimedia streaming. Figure 7 shows how delivery latency increases with the number of trees and different churn rates. The standard deviation for Magellan-Nemo S2 ranges from 0.03 seconds at low failure rates to 0.13 seconds at the highest simulated failure rate. In general, the standard deviation exceeds 0.3 seconds only in very few cases. The highest



0.4

Fig. 6: Delivery ratio (512 end hosts, unlimited bandwidth). The combination of in-tree redundancy with multiple trees substantially increases the latter's effectiveness, i.e. two trees achieve near perfect delivery at high churn rates.

Protocol	Max. Outdegree	Non-contributors [%]
Nice	18.0	83.0
Nemo	10.0	52.7
Magellan (Nemo) S2	10.8	17.8
NUScribe	138.0	94.1
NUSplitStream S2	66.7	91.2
NUSplitStream S16	11.8	25.2

TABLE V: Outdegree (512 end hosts, unlimited bandwidth, MTTF=120 min, MTTR=20 min).

observed standard deviation is 0.4 seconds for SplitStream S2 at a MTTF of 15 minutes. As the degree of churn increases, the delivery latency of a given system tends to increase as the system lacks sufficient time to run its tree optimization algorithms. At very high levels of churn, some protocols' delivery latencies will seem to improve again as result of the reduced delivery ratio observed (Figure 4) and the tendency of packets to reach peers closer in the distribution topology with higher probability than those farther away.

The increased delivery latency of DHT-based protocols results from some of these protocols use of a unique node identifier to build their routing tables and their consideration of latency only as a tie breaker. Reverse path forwarding on the resulting routing topology may thus impose considerable overhead in terms of latency, especially for small peer populations. We expect this effect to be less pronounced for large groups due to the density of the participating peers' population.

Beyond peer population transiency, the resilience of a protocol is also affected by the available bandwidth capacities at the end hosts. Imposing high forwarding responsibilities on some nodes may easily overload them, resulting in significant packet losses. Figure 8 illustrates the distribution of the forwarding responsibility for some of the evaluated protocols. The graph shows the physical outdegree of the nodes on the x-axis and

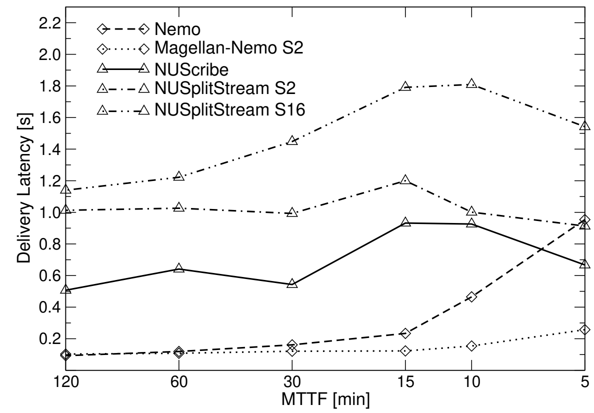


Fig. 7: Delivery latency (512 end hosts, unlimited bandwidth). Delivery latency is negatively affected by churn as node departure and arrival events work against the protocols optimization strategies. Eventually, all protocols will experience lower delivery latencies from significant packet losses and the fact that delivered packets reach nearby nodes with higher probability.

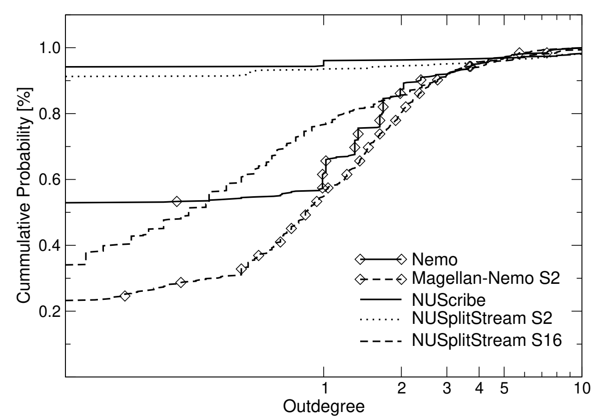


Fig. 8: Outdegree (512 end hosts, unlimited bandwidth, MTTF=120 min, MTTR=20 min). The outdegree partially determines the scalability of an overlay system in bandwidth limited scenarios.

the cumulative fraction of the nodes on the y-axis. The physical outdegree is the forwarding capacity used at each node in terms of a basic stream rate. That is, a physical outdegree of one indicates that the peer contributes exactly the equivalent of one full rate stream to the system. While conventional tree-based protocols with no alternate paths put significantly high forwarding load on a few nodes in the system, path diversity (as offered by in-tree redundancy) and multiple disjoint trees substantially reduce forwarding responsibility for most peers. Table V summarizes the maximal outdegree and the number of non-contributors in each of the delivery topologies, clearly illustrating the advantages of path diversity and multiple-tree

Protocol	Delivery Latency [s]	Improvement [%]
Nice	0.464	-
Magellan (Nice) S4	0.369	20.5

TABLE VI: Delivery latency (100 end hosts, wide-area, MTTF=10 min, MTTR=2 min).

redundancy.

In addition, in bandwidth-limited environments the use of multiple trees can help reduce the queuing delay at each overlay hop, thus improving total delivery latency. Table VI shows the delivery ratio of multiple trees. We see that Magellan-Nice with four trees reduces the average delivery latency by more than 20% when compared to Nice.

All the proposed resilient techniques scale well when compared to their baseline, illustrating a powerful, if logical, synergy of goals between high resilience and scalability for overlay multicast protocols. In particular, SplitStream overcomes the inherent scalability problem of conventional tree-based multicast schemes in homogeneous environments. Nemo also greatly reduces the physical outdegree requirement for interior nodes when compared to conventional tree-based protocols like Nice.

D. Summary

We have evaluated three alternative techniques for higher resilience: cross-links, in-tree path diversity and multiple-tree redundancy. While all of the evaluated techniques are able to substantially increase the resilience of their base protocol, each achieves this at different relative costs. For example, while employing multiple trees improves delivery ratio and reduces the bandwidth requirement of individual peers, it may also result in higher delivery latencies. The latter effect becomes particular pronounced when using several disjoint trees. In bandwidth-limited environments, the outdegree distribution may become a key factor of the overall resilience as bandwidth limited peers may become bottlenecks in the system resulting in substantial packet losses. Using path diversity and/or multiple trees helps to address this problem. Overall, a combination of in-tree and multiple-tree redundancy seems to efficiently achieve the highest delivery ratio under different failures scenarios.

VI. Related Work

To the best of our knowledge this is the first study of alternative resilient techniques for overlay multicast.

In [3] the authors describe and analytically compare a set of non-resilient application layer protocols including DHT-based and tree-based techniques. Castro et al. [15] contrast CAN-style versus Pastry-style overlay networks using multicast communication workloads running on an identical simulation infrastructure. They conclude that the DHT-based, tree-building approach achieves lower delay and overhead than flooding regardless of the underlying DHT system, and that multicast trees built on Pastry provide higher performance than those using CAN [39].

The feasibility of streaming applications in heterogeneous environments has recently drawn significant attention. Chu et al. [18] report on their experience in deploying an overlay service in these environments. Sripanidkulchai et al. [44] study the feasibility of supporting large-scale groups using an application end-point architecture and conclude that the end hosts have sufficient resources in most scenarios to support such an overlay structure. Bharambe et al. [7] analyze the impact of heterogeneous bandwidth constraints on DHT-based multicast protocols and conclude that Scribe tends to create deep unbalanced distribution trees under these conditions. Our work extends this study by analyzing the implications of transiency and bandwidth heterogeneity on the effectiveness of DHT-based, cooperative multicast approaches [8]. Deep unbalanced trees posit a challenge to resilient protocols as they introduce additional points of failure in the overlay. Rhea et al. [40] show, through an emulation-based evaluation, the potential impact of realistic levels of peer transiency on the performance of some earlier DHT implementations. The authors propose a number of techniques for more churn-resilient DHTs, a few of which have found their way into recent systems.

VII. Conclusions

We have evaluated different techniques for resilient peer-to-peer multicast and analyzed their effectiveness in the context of their non-resilient alternatives. Our experimental study, the first one of its class, compares the performance of several resilient and non-resilient, tree-based and DHT-based overlay multicast systems [4] through simulation and wide-area experimentation in the PlanetLab testbed. The resilience and overhead of the different protocols is evaluated under a continuous stream of failures at different rates obtained from the literature.

While each technique on its own achieves promising performance gains, a combination of in-tree and multiple-tree redundancy exhibits the highest degree of resilience and the lowest relative cost, among the evaluated protocols. In bandwidth-limited scenarios, multiple-tree redundancy lessens the forwarding load and potentially the height of trees, thus improving both delivery ratio and end-to-end latency. In wide-area evaluations, Magellan-Nice with four trees significantly improves delivery latency (by over 20%) in contrast with the baseline, single-tree Nice.

References

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proc. of the 18th ACM SOSP*, October 2001.

[2] T. Anderson, S. Shenker, I. Sotica, and D. Wetherall. Design guidelines for robust Internet protocols. In *Proc. of HotNets-I*, October 2002.

[3] S. Banerjee and B. Bhattacharjee. A comparative study of application layer multicast protocols, 2002. Submitted for review.

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

[4] Source code for many of the these protocols, including Nemo, NUScribe and NUSplitStream is publicly available from our research group's resource page at *http://www.aqualab.cs.northwestern.edu*.

[5] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *Proc. of ACM SIGMETRICS*, June 2003.
[6] M. Bawa, H. Deshpande, and H. Garcia-Molina. Transience of peers & streaming media. In *Proc. of HotNets-I*, October 2002.
[7] A. R. Bharambe, S. G. Rao, V. N. Padmanabhan, S. Seshan, and H. Zhang. The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols. In *Proc. of IPTPS*, February 2005.
[8] S. Birrer and F. E. Bustamante. The feasibility of dht-based streaming multicast. In *Proc. of the IEEE/ACM MASCOTS*, Atlanta, GA, September 2005.
[9] S. Birrer and F. E. Bustamante. Magellan: Performance-based, cooperative multicast. In *Proc. of IWCW*, September 2005.
[10] S. Birrer and F. E. Bustamante. Resilient peer-to-peer multicast without the cost. In *Proc. of MMCN*, January 2005.
[11] S. Birrer, D. Lu, F. E. Bustamante, Y. Qiao, and P. Dinda. FatNemo: Building a resilient multi-source multicast fat-tree. In *Proc. of IWCW*, October 2004.
[12] F. E. Bustamante and Y. Qiao. Friendships that last: Peer lifespan and its role in P2P protocols. In *Proc. of IWCW*, October 2003.
[13] K. L. Calvert, M. B. Doar, and E. W. Zegura. Modeling internet topology. *IEEE Communications Magazine*, 35(6):160–163, June 1997.
[14] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proc. of the 19th ACM SOSP*, October 2003.
[15] 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 *Proc. of IEEE INFOCOM*, March 2003.
[16] M. Castro, A. Rowstron, A.-M. Kermarrec, and P. Druschel. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication*, 20(8), 2002.
[17] Y. Chawathe. *Scattercast: an architecture for Internet broadcast distribution as an infrastructure service*. Ph.D. Thesis, U. of California, Berkeley, CA, Fall 2000.
[18] Y.-H. Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early experience with an Internet broadcast system based on overlay multicast. In *Proc. of USENIX ATC*, June 2004.
[19] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communication*, 20(8), October 2002.
[20] B. Cohen. BitTorrent. bitconjurer.org/BitTorrent/, 2001. File distribution.
[21] Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of broadcast packets. *Communication of the ACM*, 21(12):1040–1048, 1978.
[22] S. E. Deering. Multicast routing in internetworks and extended LANs. In *Proc. of ACM SIGCOMM*, August 1988.
[23] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1), January/February 2000.
[24] M. B. Doar. A better model for generating test networks. In *Proc. of Globecom*, November 1996.
[25] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. In *Proc. of ACM SIGCOMM*, August 2000.
[26] P. Francis. Yoid: Extending the Internet multicast architecture. http://www.aciri.org/yoid, April 2000.
[27] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, modeling and analysis of a peer-to-peer file-sharing workload. In *Proc. of ACM SOSP*, December 2003.
[28] A. Haeberlen, A. Mislove, A. Post, and P. Druschel. Fallacies in evaluating decentralized systems. In *Proc. of IPTPS*, February 2006.
[29] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable multicasting with and overlay network. In *Proc. of the 4th USENIX OSDI*, October 2000.
[30] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proc. of the 19th ACM SOSP*, October 2003.
[31] D. Lu and P. A. Dinda. GridG: Generating realistic computational grids. *ACM Sigmetrics Performance Evaluation Review*, 30(4):33–41, March 2003.
[32] D. Lu and P. A. Dinda. Synthesizing realistic computational grids. In *Proc. of Supercomputing*, November 2003.
[33] R. Mahajan, M. Castro, and A. Rowstron. Controlling the cost of reliability in peer-to-peer overlays. In *Proc. of IPTPS*, February 2003.
[34] D. L. Mills. Improving algorithms for synchronizing computer network clocks. In *Proc. of ACM SIGCOMM*, August 1994.
[35] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient peer-to-peer streaming. In *Proc. of IEEE ICNP*, 2003.
[36] V. S. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proc. of IPTPS*, February 2005.
[37] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An application level multicast infrastructure. In *Proc. of USENIX USITS*, March 2001.
[38] PlanetLab Consortium. PlanetLab. http://www.planet-lab.org.
[39] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proc. of NGC*, November 2001.
[40] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz. Handling churn in a DHT. In *Proc. of USENIX ATC*, December 2004.
[41] A. Rowstron and P. Drushel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM Middleware*, November 2001.
[42] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. In *ACM SIGCOMM Internet Measurement Workshop*, November 2002.
[43] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-based content routing using xml. In *Proc. of the 18th ACM SOSP*, October 2001.
[44] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang. The feasibility of supporting large-scale live streaming applications with dynamic application end-points. In *Proc. of ACM SIGCOMM*, August/September 2004.
[45] D. A. Tran, K. A. Hua, and T. Do. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proc. of IEEE INFOCOM*, April 2003.
[46] Z. Wang and J. Crowcroft. Bandwidth-delay based routing algorithms. In *Proc. of IEEE GlobeCom*, November 1995.
[47] K.-F. S. Wong, S. G. Chan, W.-C. Wong, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang. Lateral error recovery for application-level multicast. In *Proc. of IEEE INFOCOM*, March 2004.
[48] J. Xu, Z. Kalbarczyk, and R. K. Iyer. Networked Windows NT system field failure data analysis. In *Proc. of PRDC*, December 1999.
[49] M. Yang and Z. Fei. A proactive approach to reconstructing overlay multicast trees. In *Proc. of IEEE INFOCOM*, March 2004.
[50] 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 *Proc. of NOSSDAV*, June 2001.