A Comparison of Resilient Overlay Multicast Approaches

Stefan Birrer, *Student Member, IEEE* and Fabián E. Bustamante *Member, IEEE*

***Abstract*— Overlay-based multicast has been proposed as a key alternative for large-scale group communication. There is ample motivation for such an approach, as it delivers the scalability advantages of multicast while avoiding the deployment issues of a network-level solution. As multicast functionality is pushed to autonomous, unpredictable end systems, however, significant performance loss can result from their higher degree of transiency when compared to routers. Consequently, a number of techniques have recently been proposed to improve overlays' resilience by exploiting path diversity and minimizing node dependencies. Delivering high application performance at relatively low costs and under high degree of transiency has proven to be a difficult task. 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. In this paper, we review some of these approaches and evaluate their effectiveness by contrasting the performance and associated cost of representative protocols through simulation and wide area experimentation.**

***Index Terms*— Peer-to-Peer, Overlay Network, Multicast, Resilience.**

I. Introduction

OVERLAY-BASED multicast has been proposed as a key alternative for large-scale group communication [1]–[12]. With an overlay-based approach, all multicast-related functionality is implemented at the end systems instead of at the routers. The participating hosts configure themselves in an overlay topology, with each edge in the overlay corresponding to a unicast path between two end systems in the underlying Internet. The goal of a multicast protocol is thus to construct and maintain an efficient overlay for data transmission.

As multicast functionality is pushed to autonomous, unpredictable end systems, however, significant performance loss can result from their higher degree of transiency when compared to routers [13]. A good indicator of peers' 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 [14]–[17]. Although collected mostly from file-sharing applications, these measurements offer an idea of the level of transiency that can be expected in large peer populations. Consequently, a number of techniques [6], [18], [19] have recently been proposed to improve overlays' resilience by exploiting path diversity [20], [21] and minimizing node dependencies [22].

Delivering high application performance at relatively low costs and under high degree of transiency has proven to be a difficult task [15], [23], [24]. 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. While the vast majority of streaming protocols follow a tree-based approach, there is growing interest in a new class of protocols adopting a mesh or data-driven model inspired in the pull-based, swarming mechanisms of systems like BitTorrent [25]. The evaluation of mesh-based streaming protocols [26], [27] as well as of those targeted for bulk-data dissemination [28]–[30] is outside 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

Manuscript received March 2, 2007; revised August 30, 2007. An early version of this work appeared in the Proc. of the IEEE/ACM MASCOTS, 2006.

The authors are with the Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, IL 60208, USA (e-mail: {sbirrer,fabianb}@cs.northwestern.edu).

technique that improves tree resilience through in-tree redundancy with a multiple-tree approach yields excellent delivery ratios under a large range of peer transiency and scale. 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 background on overlay multicast and overviews some of the techniques proposed for improving resilience. The alternative strategies are illustrated with more detailed description of representative protocols. In Sections III and IV, we outline our evaluation setting and report experimental results from simulations and wide-area experimentation. We describe related work in Section V and conclude in Section VI.

II. Resilient Approaches to Overlay Multicast

We begin this section with a brief overview of overlay multicast before discussing alternative approaches for improving overlays' resilience.

A. Overlay Multicast

Peers in overlay multicast protocols self-organize in two topologies: one used for group-membership related tasks and a second one for data dissemination. 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. In a *tree-first* approach [2]–[4], peers directly build a 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, the data delivery overlay is defined over a partially connected graph (mesh). The structure for data delivery can be explicitly defined, as (reverse) shortest path spanning trees [1], or implicitly defined based on data availability [26], [27].[1] Under the *DHT-first* approach, peers organize themselves into a well-defined geometrical structure over which a data delivery topology is built [10], [32]. Last, following the *implicit* approach, peers create only a control topology, while the data delivery tree is implicitly defined by packet forwarding rules based on the control tree. Resilient techniques for overlay multicast have been targeted mainly at the latter two, the DHT-first and the implicit approach. Thus, the following paragraphs discuss them into more detail based on representative example protocols.

[1]This last class is sometimes refer to as mesh-based, treeless or data-driven [25], [27], [31].

Scribe [9] is probably one of the best known *DHT-first* overlay multicast protocols. It builds upon Pastry [32], a structured (DHT) P2P overlay. Every peer in Pastry [32] 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 [9] 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 [33]. Most recent implementations of Scribe and Pastry incorporate the suggestions in Rhea et al. [23], in an attempt to minimize the impact of churn on DHT-based overlays.

Nice [8] is one of the earliest *implicit* multicast protocols. It belongs to a general class of protocols 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 where the focus is on maintaining a structure based on a virtual-id space [34]. Participating peers in Nice 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 clusters' 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 [6]–[8], [35].

B. Alternative Approaches to Resilient Overlay Multicast

Given the impact of node transiency on the performance of overlay multicast protocols, a number of techniques [6], [18], [19], [31], [36] has 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 [6], [12], [18], [37]. *Multiple-tree redundancy* creates several overlapping trees over which stripes of the multicast stream are forwarded [11], [19], [36]. Figure 1 illustrates each of these classes. This section reviews each of them in the context of concrete, representative protocols.

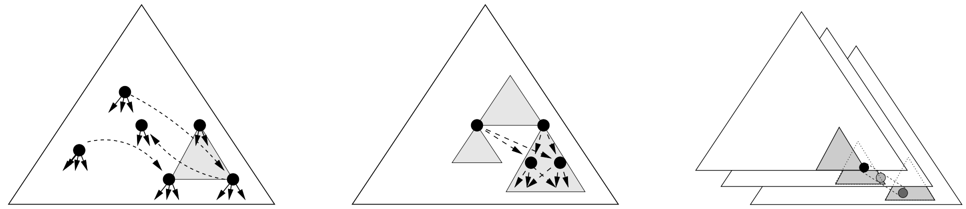
1) Cross-Link Redundancy: Probabilistic Resilient Multicast (PRM) is a general scheme to improve the resilience of overlay multicast [18] through randomized forwarding. PRM adopts triggered negative acknowledgment, as well as a *cross-link redundancy* approach, forwarding an additional fraction of the stream over extra cross-cutting links connecting random peers in the tree [18]. In $PRM(n,p)$, a node continuously discovers n random session members and forwards every received data packet to any of those members with a specified probability p. These random packets help detect and recover from temporary tree partitions. Under stable conditions, however, such an approach introduces $n{\cdot}p$ duplicate packets and can incur a high overhead in heterogeneous settings.

2) 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 [6] organizes nodes into clusters based on network proximity,[2] 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 and each cluster leader recruits a number of coleaders to form its crew. 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.

Nemo achieves resilience through its 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 within each cluster (subtree) in the tree. 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 previous situation which would have required intervention.

3) Multiple-Tree Redundancy: In conventional tree-based multicast systems, a relatively small set

[2]Other factors such as bandwidth [1], [38] and expected peer lifetime [14] 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 within each cluster (subtree) in the tree. The *multiple-tree* redundancy approach creates several overlapping trees over which stripes of the multicast stream are forwarded.

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. [19] propose the use of multiple interior-node disjoint trees (a forest) 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 [9] 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 heterogeneous environments, *Magellan* adopts the same *multiple-tree redundancy* approach, but building instead on a forest of performance-centric trees [6], [8]. 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, 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 [39]. Magellan was originally implemented using Nemo, inheriting the latter's churn-resilience properties. For our evaluation, we also implemented a variant of Magellan that relies on Nice [8] as its underlying tree construction protocol to understand the independent contributions of *multiple-tree* and *in-tree redundancy* to performance-centric multicast protocols.

III. Evaluation

Our evaluation aims at determining the tradeoff brought in by each of the proposed resilient techniques (Section II) 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 (Table I).

We carried out our evaluation both through simulation and Internet experimentation in the PlanetLab wide-area testbed [40]. 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 base

TABLE I
RESILIENT PROTOCOLS EVALUATED.

Resilience Approach	Protocol	Class	Baseline
Cross-Link	Nice-PRM	Implicit	Nice
In-Tree	Nemo	Implicit	Nice
Multi-Tree	SplitStream	DHT-first	Scribe
	Magellan-Nice	Implicit	Nice
Multi-Tree/In-tree	Magellan-Nemo	Implicit	Nemo

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 IV 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 [40] nodes.

We ran simulations using GridG [41], [42] topologies with 8,115 nodes, and a multicast group of 256 and 512 members. GridG leverages Tiers [43], [44] 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. For the comparison, we chose not to model bandwidth to avoid side effects due to control traffic competing for available bandwidth [19]. For the scalability analysis, the nodes' bandwidth capacities are assigned based on bandwidth traces gathered through real-world measurements on the Gnutella network [45].

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. [17], [46]. By exploring a wide-range of MTTF, we avoid biases toward any particular deployment environment [47]. 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 [48] 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 [6], [8], [9], [18], [19], [32], [36]. We have made them available to the community from our research group's resource page. [3]

For Nice [8] and Nice-PRM [18], the cluster degree, k, was set to three. We used PRM with

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

three random peers chosen by each node, and with two percent forwarding probability. Evaluation results with other, suboptimal, forwarding probabilities were reported in [6]. For Nemo, the cluster degree k and the crew size were set to three as well. The grace period was set to 15 seconds for Nice, Nice-PRM, Nemo and Magellan.

For Scribe and SplitStream,[4] 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 [19], to give SplitStream the same ability to detect failures as Nice and Nemo. The outdegree for SplitStream nodes is unlimited, yielding maximum performance for the unlimited bandwidth scenario [19]. In addition to the performance-optimized variant of SplitStream evaluated in this paper, the protocol can also be configured with perfect fairness in mind. In this case, each peer contributes bandwidth resources corresponding to one full-rate split. Enforcing such tight outdegree requirements, however, results in 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 scope of this paper. The interested reader is referred to the detailed analysis in the original report [19].

We evaluate SplitStream and Magellan with 2, 4, 8 and 16 stripes (s) (trees), where each stripe is responsible for forwarding $\frac{1}{s}$ of the content to each client. Thus, 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 [49]. 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.

- **Throughput:** The total amount of data received measured in Megabits per second (Mbps). In general, throughput is expected to scale with the number of receivers.
- **Delivery Ratio:** Ratio of subscribers which have received a packet within a fixed time window.
- **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 packet-forwarding capacity, as a fraction of a basic stream rate, contributed by a node. It serves as a good indicator of nodes' total bandwidth contributions. Although the outdegree of a node is sometimes defined as the number of a node's successors, this definition impedes comparison with protocols in which nodes can forward only part of the total data stream.
- **Overhead:** Total control traffic in the system, in bits per second (bps) per peer, during the observation interval. We measure the total control traffic by accounting packets at the end hosts access link.

IV. 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. We first discuss the effectiveness of the different resilient techniques and their combination in terms of delivery ratio and overhead. We then report on their impact on delivery latency and analyze the implication of forwarding load balancing on resilience. We conclude our evaluation of resilient techniques with a discussion of their scalability. Unless otherwise noted, all reported results are based

[4]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 [23], [24].

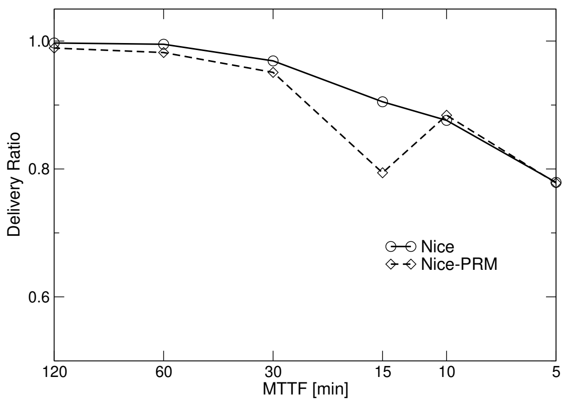


Fig. 2. PRM increases the average delivery ratio for different degrees of churn, specified in terms of MTTF (512 end hosts, unlimited bandwidth). Note that the x-axis crosses at 0.5 delivery ratio.

on five independent runs per protocol and setup, for both the simulation and wide-area experiments.

A. Delivery Ratio and Overhead

Tree-based protocols organize participating peers into a logical 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 that 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 in delivery ratio at different levels of churn.

Randomly forwarding data packets over cross-links potentially creates duplicate packets at some nodes. Table II illustrates the data overhead for Nice and Nice-PRM at two failure rates. With MTTF of 2 hours, PRM incurrs a 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, 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.

TABLE II

OVERHEAD OF CROSS-LINK REDUNDANCY IN TERMS OF DUPLICATE PACKETS AND CONTROL MESSAGES (512 END HOSTS, UNLIMITED BANDWIDTH). PRM ADDS A CONSTANT OVERHEAD THAT IS INDEPENDENT OF THE FAILURE RATE.

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

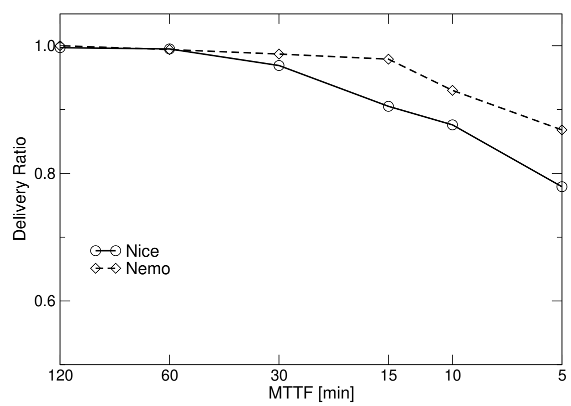


Fig. 3. In-tree redundancy significantly improves delivery ratio at high degree of churn, specified in terms of MTTF (512 end hosts, unlimited bandwidth). Note that the x-axis crosses at 0.5 delivery ratio.

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 that 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

TABLE III

OVERHEAD OF IN-TREE REDUNDANCY IN TERMS OF DUPLICATE PACKETS AND CONTROL MESSAGES (512 END HOSTS, UNLIMITED BANDWIDTH). NEMO'S IN-TREE APPROACH ADDS A SMALL OVERHEAD.

Protocol	Duplicate Pkts [%]	Control Pkts [/sec]
MTTF=120 min, MTTR=20 min		
Nice	0.1	1.64
Nemo	0.4	1.67
MTTF=30 min, MTTR=5 min		
Nice	0.2	1.57
Nemo	3.1	1.67

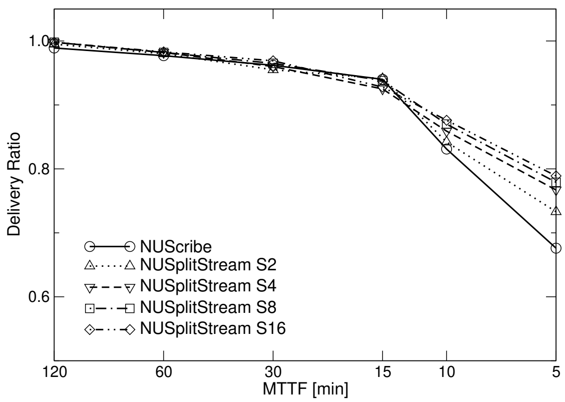


Fig. 4. Multiple-tree redundancy helps increasing resilience at very high churn rates, specified in terms of MTTF (512 end hosts, unlimited bandwidth). Note that the x-axis crosses at 0.5 delivery ratio.

TABLE IV

DELIVERY RATIO OF MULTIPLE-TREE RESILIENCE (100 END HOSTS, WIDE-AREA, MTTF=10 MIN, MTTR=2 MIN).

Protocol	Delivery Ratio
Nice	0.90
Magellan (Nice) S4	0.94

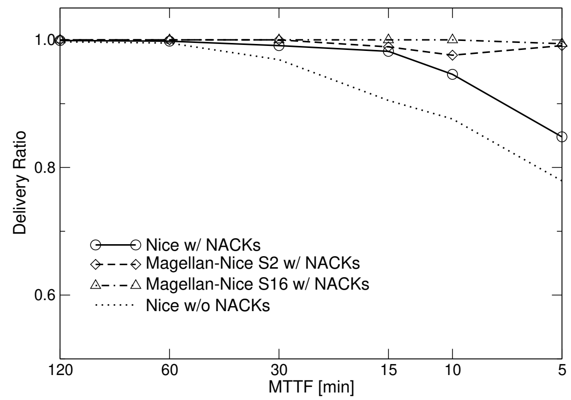


Fig. 5. Lateral-error recovery significantly increases the resilience of multiple-tree protocols (512 end hosts, unlimited bandwidth).

2 hours whereas the outermost right one corresponds 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.

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 in higher number of duplicate data packets. Table III also shows that Nemo's and Nice's overheads, at low churn rates, are comparable. The maintenance of Nemo's in-tree redundancy additionally introduces a small relative increase in control packets when compared to the baseline protocol Nice.

Using multiple disjoint trees, *multiple-tree redundancy* 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 exceeds 2% only for (most of) the single-tree measurements and for 2, 4 and 8 trees under very high failure rates. The highest observed standard deviation is 4.5% using a single-tree approach with MTTF of 5 minutes. The benefit of multiple-tree redundancy becomes 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 IV 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.

Having lateral error recovery with multiple trees allows peers to restore missing packets from other trees (i.e. not only the forwarding one) and thus better overcome a temporary delivery outage in any

TABLE V

OVERHEAD OF MULTIPLE-TREE RESILIENCE ON DHT-FIRST PROTOCOLS (512 END HOSTS, UNLIMITED BANDWIDTH).

Protocol	Duplicate Pkts [%]	Control Pkts [/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

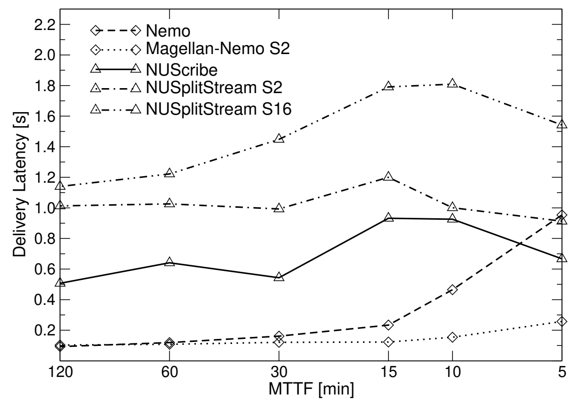


Fig. 7. Delivery latency is negatively affected by churn as node departure and arrival events work against the protocols' optimization strategies (512 end hosts, unlimited bandwidth). With high packet losses protocols experience low delivery latencies since packets reach nearby nodes with higher probability.

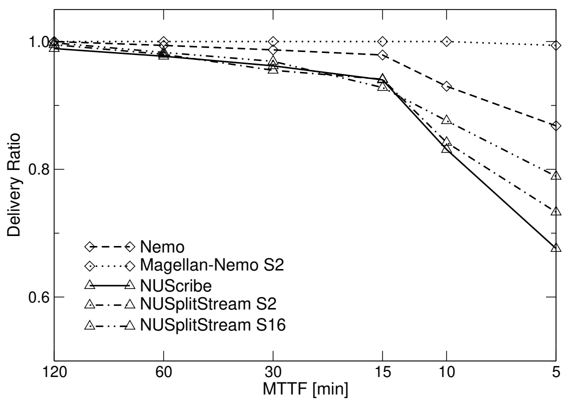


Fig. 6. Combining in-tree redundancy with multiple trees substantially increases the latter's effectiveness in terms of delivery ratio (512 end hosts, unlimited bandwidth).

TABLE VI

DELIVERY LATENCY OF IN-TREE REDUNDANCY (512 END HOSTS, UNLIMITED BANDWIDTH). SUBOPTIMAL PATHS DO NOT HAVE TO YIELD INCREASED DELIVERY LATENCY.

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

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.

As Table V illustrates, maintaining multiple trees naturally results in higher 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.

B. Delivery Latency

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 VI 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 latency as packets addressed to peers nearby the source in the overlay are more likely to succeed than destined to further away nodes.

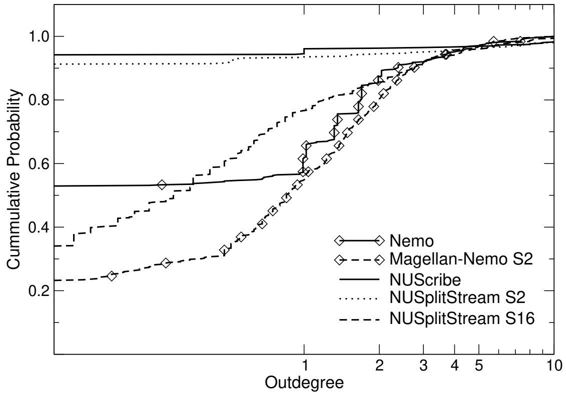


Fig. 8. Physical outdegree is the packet-forwarding capacity, as a fraction of a basic stream rate, contributed by a node. It determines the scalability of an overlay system in bandwidth limited scenarios (512 end hosts, unlimited bandwidth, MTTF=120 min, MTTR=20 min).

Building multiple disjoint trees restricts the options for internal nodes and potentially results in increased delivery latencies. 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 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.

TABLE VII

OUTDEGREE (512 END HOSTS, UNLIMITED BANDWIDTH, MTTF=120 MIN, MTTR=20 MIN).

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 VIII

DELIVERY LATENCY (100 END HOSTS, WIDE-AREA, MTTF=10 MIN, MTTR=2 MIN).

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

C. Outdegree and Resilience

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 the cumulative fraction of the nodes on the y-axis. The physical outdegree is the packet-forwarding capacity, as a fraction of a basic stream rate, contributed by a node. 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 VII summarizes the maximum outdegree and the number of non-contributors in each of the delivery topologies, clearly illustrating the advantages of path diversity and multiple-tree 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 VIII illustrates the average delivery latency of multiple trees. We see that Magellan-Nice with four trees reduces the average delivery latency by more than 20% when compared to Nice.

D. Scalability

Clearly, large group sizes pose a significant challenge to resilient overlay multicast protocols. 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. While the non-resilient protocols perform adequately at smaller scales, the resilient ones show clear improvements over their non-resilient counterparts for increasing group sizes. 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.

Table IX illustrates the scalability of some representative non-resilient and resilient performance-centric protocols. The improved scalability of the resilient variants is particularly obvious with real-world bandwidth traces [45]. Bandwidth constraints limit the protocols' ability to recover from packet losses and or membership changes, potentially negatively impacting its performance. This can be already observed with 512 peers where all protocols deliver most of the packets, but some show significant increases on delivery latency. We see that Nice is significantly impacted by large queuing delays that, at larger scales, turn into losses as the links become overloaded.

In addition to increasing resilience, multipath data forwarding helps reduce bottlenecks in bandwidth constraint scenarios. Multipath techniques (PRM, Nemo, Magellan) consequently offer reduced delivery latency under load and suffer from fewer congestion losses. With group size of 512 peers, we see that the multipath techniques exhibit significant lower latency than the single path (Nice) protocols. As scale and thus load at the bottlenecks increases, packet drops limit the overall delivery. This can be observed in the throughput and delivery ratio with 1024 peers. Whenever the delivery ratio drops significantly, latency follows as peers closer to the source are more likely to receive a forwarded data packet. This effect can be observed with Nice when going from a group of 512 peers to 1024 peers.

The amount of control-related traffic is a key factor in the scalability of any approach. Bandwidth used for control traffic reduces the potential goodput. It is thus of interest to keep this within reasonable bounds. Table IX shows the total control overhead in kilo bits per second (kbps) per peer in the system. It is interesting to note that for all the presented approaches, the fraction of control overhead remains nearly constant with increasing group size and never exceeds 18 kbps.

E. 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.

V. Related Work

To the best of our knowledge this is the first study of alternative resilient techniques for tree-based overlay multicast. A number research proposals have addressed reliable group communication at the network layer; two excellent comparative studies include [50], [51].

In [52] the authors describe and analytically compare a set of non-resilient overlay multicast protocols including DHT-based and tree-based techniques. Castro et al. [53] 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

TABLE IX

SCALABILITY WITH AND WITHOUT BANDWIDTH CONSTRAINTS. IN THE BANDWIDTH CONSTRAINED GNUTELLA (GNT) SCENARIO, WE MODEL THE INTERNET WITH INFINITE BANDWIDTH CAPACITY AND ASSIGN CAPACITIES TO END HOSTS FOLLOWING REAL-WORLD TRACES MEASURED IN THE POPULAR GNUTELLA NETWORK [45] (MTTF=30 MIN, MTTR=5 MIN).

Scale	Protocol	Throughput [Mbps]		Delivery Ratio		Delivery Latency [sec]		Control Overhead [kbps]	
		∞	GnT	∞	GnT	∞	GnT	∞	GnT
128	Nice	80.9	76.4	0.99	0.93	0.05	0.43	5.3	5.2
	Nice w/ NACK	79.8	72.8	0.97	0.89	0.07	0.88	7.1	6.6
	Nice PRM	81.8	79.3	1.00	0.97	0.06	0.46	14.5	15.2
	Nemo	82.2	80.4	1.00	0.98	0.06	0.47	5.7	7.5
	Magellan-Nice S2	82.2	80.4	1.00	0.98	0.11	0.70	13.4	17.1
256	Nice	161.0	139.1	0.98	0.85	0.05	0.71	5.5	5.2
	Nice w/ NACK	160.8	152.5	0.98	0.93	0.09	0.69	5.6	6.2
	Nice PRM	162.2	159.6	0.99	0.97	0.08	0.37	14.2	14.5
	Nemo	162.2	160.7	0.99	0.98	0.09	0.48	5.8	7.6
	Magellan-Nice S2	162.7	159.6	0.99	0.97	0.14	0.85	15.3	16.7
512	Nice	312.3	222.5	0.95	0.68	0.06	2.25	5.6	5.1
	Nice w/ NACK	303.9	227.9	0.93	0.70	0.08	2.87	6.0	7.0
	Nice PRM	318.3	310.2	0.97	0.95	0.11	0.51	14.3	14.9
	Nemo	318.0	304.8	0.97	0.93	0.12	0.92	6.1	8.6
	Magellan-Nice S2	315.2	308.7	0.96	0.94	0.24	0.94	16.6	17.0
1024	Nice	616.1	450.5	0.94	0.69	0.07	1.75	5.8	5.3
	Nice w/ NACK	626.7	227.9	0.96	0.70	0.08	0.78	5.9	7.1
	Nice PRM	624.1	614.6	0.95	0.94	0.16	0.55	14.5	15.2
	Nemo	628.8	614.0	0.96	0.94	0.12	0.73	6.2	8.0
	Magellan-Nice S2	628.3	604.2	0.96	0.92	0.22	1.28	16.7	17.7

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 [10].

The feasibility of streaming applications has recently drawn significant attention. Chu et al. [15] report on their experience in deploying an overlay service. Sripanidkulchai et al. [54] 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. Birrer et al. [36], Sung et al. [55] and Venakataraman et al. [31] proposed the adoption of multitrees to leveraging the heterogeneity of bandwidth availability among peers. Birrer et al. [56] and Bharambe et al. [34] analyze the impact of transiency and heterogeneous bandwidth constraints on DHT-based multicast protocols. Rhea et al. [23] 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. In their description of PRM, Banerjee et al. [57] present a detailed comparison of Nice, and Nice-PRM with an approach based on forward error correction [58]–[61] and argue that FEC-based approaches are not alone sufficient for resilient multicast, especially for domains such as streaming multicast where low delivery latencies are required. The churn resilience problem of early tree-based protocols has motivated a number of recently proposed protocols based on a mesh or data-driven approach to data dissemination [26], [27]. Magharei et al. [25] report on an interesting comparison of mesh/data-driven and tree-based streaming protocols.

VI. 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 overlay multicast systems [5] through simulation and wide-area experimentation in the PlanetLab testbed. The resilience and overhead of the different protocols, both

[5] 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*.

performance-centric and DHT-based, are evaluated under a continuous stream of failures at different rates obtained from the literature, and for different number of peers.

While each of the resilience techniques 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.

ACKNOWLEDGMENTS

We would like to thank the FreePastry group at Rice University for making their code available, especially to Alan Mislove for answering our many questions concerning FreePastry. We also thank Jeanine M. Casler, Yi Qiao, David Choffnes and the anonymous referees for their many comments on early drafts of this paper.

REFERENCES

[1] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A case for end system multicast," *IEEE Journal on Selected Areas in Communication*, vol. 20, no. 8, October 2002.

[2] 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.

[3] P. Francis, "Yoid: Extending the Internet multicast architecture," April 2000, http://www.aciri.org/yoid.

[4] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel, "ALMI: An application level multicast infrastructure," in *Proc. of USENIX USITS*, March 2001.

[5] Y. Chawathe, "Scattercast: an architecture for Internet broadcast distribution as an infrastructure service," Ph.D. Thesis, U. of California, Berkeley, CA, Fall 2000.

[6] S. Birrer and F. E. Bustamante, "Resilient peer-to-peer multicast without the cost," in *Proc. of MMCN*, January 2005.

[7] 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.

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

[9] 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*, vol. 20, no. 8, 2002.

[10] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-level multicast using content-addressable networks," in *Proc. of NGC*, November 2001.

[11] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient peer-to-peer streaming," in *Proc. of IEEE ICNP*, 2003.

[12] 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.

[13] M. Bawa, H. Deshpande, and H. Garcia-Molina, "Transience of peers & streaming media," in *Proc. of HotNets-I*, October 2002.

[14] F. E. Bustamante and Y. Qiao, "Friendships that last: Peer lifespan and its role in P2P protocols," in *Proc. of IWCW*, October 2003.

[15] 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.

[16] S. Sen and J. Wang, "Analyzing peer-to-peer traffic across large networks," in *ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[17] 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.

[18] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," in *Proc. of ACM SIGMETRICS*, June 2003.

[19] 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.

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

[21] A. C. Snoeren, K. Conley, and D. K. Gifford, "Mesh-based content routing using xml," in *Proc. of the 18th ACM SOSP*, October 2001.

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

[23] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a DHT," in *Proc. of USENIX ATC*, December 2004.

[24] R. Mahajan, M. Castro, and A. Rowstron, "Controlling the cost of reliability in peer-to-peer overlays," in *Proc. of IPTPS*, February 2003.

[25] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live P2P streaming approaches," in *Proc. of IEEE INFOCOM*, May 2007.

[26] N. Magharei and R. Rejaie, "PRIME: peer-to-peer receiver-driven mesh based streaming," in *Proc. of IEEE INFOCOM*, Anchorage, Alaska, May 2007.

[27] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: a data-driven overlay network for efficient live media streaming," in *Proc. of IEEE INFOCOM*, Miami, FL, March 2005.

[28] 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.

[29] B. Cohen, "BitTorrent," bitconjurer.org/BitTorrent/, 2001, file distribution.

[30] V. Pai, K. Tamilmani, V. Sambamurthy, K. Kumar, and A. Mohr, "Chainsaw: eliminating trees for overlay multicast," in *Proc. of IPTPS*, Ithaca, NY, February 2005.

[31] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured end system multicast," in *icnp*, Santa Barbara, CA, November 2006.

[32] A. Rowstron and P. Drushel, "Pastry: Scalable, distributed ob-

ject location and routing for large-scale peer-to-peer systems," in *IFIP/ACM Middleware*, November 2001.

[33] Y. K. Dalal and R. M. Metcalfe, "Reverse path forwarding of broadcast packets," *Communication of the ACM*, vol. 21, no. 12, pp. 1040–1048, 1978.

[34] 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.

[35] S. Birrer and F. E. Bustamante, "Resilience in overlay multicast protocols," in *Proc. of the IEEE/ACM MASCOTS*, September 2006.

[36] ——, "Magellan: Performance-based, cooperative multicast," in *Proc. of IWCW*, September 2005.

[37] M. Yang and Z. Fei, "A proactive approach to reconstructing overlay multicast trees," in *Proc. of IEEE INFOCOM*, March 2004.

[38] Z. Wang and J. Crowcroft, "Bandwidth-delay based routing algorithms," in *Proc. of IEEE GlobeCom*, November 1995.

[39] 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.

[40] PlanetLab Consortium, "PlanetLab," http://www.planet-lab.org.

[41] D. Lu and P. A. Dinda, "GridG: Generating realistic computational grids," *ACM Sigmetrics Performance Evaluation Review*, vol. 30, no. 4, pp. 33–41, March 2003.

[42] ——, "Synthesizing realistic computational grids," in *Proc. of Supercomputing*, November 2003.

[43] M. B. Doar, "A better model for generating test networks," in *Proc. of Globecom*, November 1996.

[44] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling internet topology," *IEEE Communications Magazine*, vol. 35, no. 6, pp. 160–163, June 1997.

[45] S. Saroiu, P. K. Gummadi, and S. D. Gribble, "A measurement study of peer-to-peer file sharing systems," in *Proc. of MMCN*, January 2002.

[46] J. Xu, Z. Kalbarczyk, and R. K. Iyer, "Networked Windows NT system field failure data analysis," in *Proc. of PRDC*, December 1999.

[47] A. Haeberlen, A. Mislove, A. Post, and P. Druschel, "Fallacies in evaluating decentralized systems," in *Proc. of IPTPS*, February 2006.

[48] D. L. Mills, "Improving algorithms for synchronizing computer network clocks," in *Proc. of ACM SIGCOMM*, August 1994.

[49] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *Proc. of ACM SIGCOMM*, August 2000.

[50] B. N. Levine and J. Garcia-Luna-Aceves, "A comparison of reliable multicast protocols," *Multimedia Systems Journal*, vol. 6, no. 5, August 1998.

[51] D. Towsley, J. F. Kurose, and S. Pingali, "A comparison of sender-initiated and receiver-initiated reliable multicast protocols," *IEEE Journal on Selected Areas in Communication*, vol. 15, no. 3, April 1997.

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

[53] 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.

[54] 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.

[55] Y.-W. Sung, M. Bishop, and S. Rao, "Enabling contribution awareness in an overlay broadcasting system," in *Proc. of ACM SIGCOMM*, September 2006.

[56] S. Birrer and F. E. Bustamante, "The feasibility of dht-based streaming multicast," in *Proc. of the IEEE/ACM MASCOTS*, Atlanta, GA, September 2005.

[57] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan, "Resilient multicast using overlays," *IEEE/ACM Transactions on Networking*, vol. 14, no. 2, pp. 237–248, April 2006.

[58] J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 349–361, August 1998.

[59] D. Rubenstein, S. Kasera, D. Towsley, and J. Kurose, "Improving reliable multicast using active parity encoding services (APES)," in *Proc. of IEEE INFOCOM*, 1999.

[60] J. Byers, M. Luby, and M. Mitzenmacher, "A digital fountain approach to asynchronous reliable multicast," *IEEE Journal on Selected Areas in Communication*, vol. 20, no. 8, pp. 1528–1540, October 2002.

[61] A. Mahanti, D. Eager, M. Vernon, and D. Sundaram-Stukel, "Scalable on-demand media streaming with packet loss recovery," in *Proc. of ACM SIGCOMM*, August 2001.

Stefan Birrer is a Ph.D. candidate in the Department of Electrical Engineering and Computer Science at Northwestern. His research interests span the intersection between distributed systems and networking. Birrer has a M.S. in computer science from Northwestern University and a B.S. in computer science from FH Aargau, Switzerland. He is the recipient of a Neokast Fellowship (2005-2007).

Fabián E. Bustamante is an assistant professor of computer science in the Department of Electrical Engineering and Computer Science at Northwestern. His research interests include distributed systems and networks, and operating systems support for massively distributed systems, including Internet and vehicular network services. Bustamante has an M.S. and Ph.D. in computer science from the Georgia Institute of Technology. Bustamante is the recipient of a National Science Foundation CAREER Award (2007). He is a member of the IEEE Computer Society, the ACM, and USENIX.