NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
NWU-CS-04-36
May 26th, 2004

Nemo – Resilient Peer-to-Peer Multicast without the Cost

Stefan Birrer and Fabián E. Bustamante

Abstract

One of the most important challenges of peer-to-peer multicast protocols is the ability to efficiently deal with the high degree of transiency inherent to their environment. As multicast functionality is pushed to autonomous, unpredictable peers, significant performance losses can result from group membership changes and the higher failure rates of end-hosts when compared to routers. Achieving high delivery ratios without sacrificing end-to-end latencies or incurring additional costs has proven to be a challenging task.

This paper introduces Nemo, a novel peer-to-peer multicast protocol that aims at achieving this elusive goal. Based on two simple techniques: (1) *co-leaders* to minimize dependencies and, (2) *triggered negative acknowledgments (NACKs)* to detect lost packets, Nemo's design emphasizes conceptual simplicity and minimum dependencies, thus achieving performance characteristics capable of withstanding the natural instability of its target environment.

We present an extensive comparative evaluation of our protocol through simulation and wide-area experimentation. We compare the scalability and performance of Nemo with that of three alternative protocols: Narada, Nice and Nice-PRM. Our results show that Nemo can achieve delivery ratios *(up to 99.9%)* similar to those of comparable protocols under high failure rates, but at a fraction of their cost in terms of duplicate packets *(reductions > 90%)* and control-related traffic *(reductions > 20%)*.

Keywords: Peer-to-peer, overlay, multicast, resilience, co-leaders, wide-area evaluation.

Nemo – Resilient Peer-to-Peer Multicast without the Cost

Stefan Birrer and Fabián E. Bustamante
Department of Computer Science
Northwestern University, Evanston IL 60201, USA,
{sbirrer,fabianb}@cs.northwestern.edu

June 22, 2004

Abstract

One of the most important challenges of peer-to-peer multicast protocols is the ability to efficiently deal with the high degree of transiency inherent to their environment. As multicast functionality is pushed to autonomous, unpredictable peers, significant performance losses can result from group membership changes and the higher failure rates of end-hosts when compared to routers. Achieving high delivery ratios without sacrificing end-to-end latencies or incurring additional costs has proven to be a challenging task.

This paper introduces Nemo, a novel peer-to-peer multicast protocol that aims at achieving this elusive goal. Based on two simple techniques: (1) *co-leaders* to minimize dependencies and, (2) *triggered negative acknowledgments (NACKs)* to detect lost packets, Nemo's design emphasizes conceptual simplicity and minimum dependencies, thus achieving performance characteristics capable of withstanding the natural instability of its target environment.

We present an extensive comparative evaluation of our protocol through simulation and wide-area experimentation. We compare the scalability and performance of Nemo with that of three alternative protocols: Narada, Nice and Nice-PRM. Our results show that Nemo can achieve delivery ratios *(up to 99.9%)* similar to those of comparable protocols under high failure rates, but at a fraction of their cost in terms of duplicate packets *(reductions $>$ 90%)* and control-related traffic *(reductions $>$ 20%)*.

1 Introduction

Multicast is an efficient mechanism to support group communication. 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. More of a decade after first being proposed, however, IP Multicast [17] is not yet widely available due in part to a number of both technical and non-technical issues [18].

Recently, a number of researchers have proposed an alternate, peer-to-peer architecture for supporting group communication applications over the Internet [16, 23, 20, 32, 13, 4, 12, 33, 44, 29, 36]. In this application-layer approach, participating peers organize themselves into an overlay topology

for data delivery. The topology is an overlay in the sense that each edge corresponds to a unicast path between two end systems or peers in the underlying Internet. All multicast related functionality is implemented at the peers instead of at the routers, and the goal of the multicast protocol is to construct and maintain an efficient overlay for data transmission.

Despite the undeniable advantages of this approach, a number of issues need to be addressed if it is to become a practical alternative to IP Multicast [16, 7]. First, application-layer multicast can result in higher stress on the network, as it is impossible to completely prevent multiple overlay edges from traversing the same physical link [16, 4]. In addition, communication between peers may involve visiting other peers, possibly resulting in higher latencies. Finally, as multicast functionality is pushed to autonomous, unpredictable peers, significant performance loss can result from the end hosts' higher degree of transiency (a.k.a. *churn*) when compared to routers.

Efficiently handling the inherent high degree of churn on peer populations may well be the primary challenge for these P2P architectures [7]. A good indication of churn 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 an hour to a minute [10, 22, 34, 14]. Achieving high delivery ratios under these conditions, without incurring additional costs or sacrificing end-to-end latencies has proven to be a difficult task.

This paper makes four main contributions. First, we introduce Nemo, a novel peer-to-peer multicast protocol that aims at achieving this elusive goal. Based on two techniques: (1) *co-leaders* and, (2) *triggered negative acknowledgments (NACKs)*, Nemo's design emphasizes conceptual simplicity and minimum dependencies [2], achieving in a cost-effective manner performance characteristics capable of withstanding the natural instability of its target environment. Nemo's approach is aimed at applications, such as real-time audio and video streaming, that can benefit from high delivery ratios without requiring perfect reliability, a model previously termed *resilient multicast* [39, 5]. In this class of applications, receivers' playback quality improves if delivery ratio can be increased within specific latency bounds.

Second, we present simulation-based and wide-area experimentations showing that Nemo can achieve high delivery ratios *(up to 99.9%)* and low end-to-end latency similar to those of comparable protocols under high failures rates, while significantly reducing costs in terms of duplicate packets *(reductions $>$ 90%)* and control related traffic *(reductions $>$ 20%)*, making the proposed algorithm a more scalable solution to the problem.

Third, Nemo addresses the resilience problem of tree-based multicast through the introduction of co-leaders, alternating leaders that help avoid dependencies on single nodes, providing alternative paths for data forwarding. We investigate the performance implications of different number of co-leaders per cluster and report our findings.

Finally, we describe the use of periodic probabilistic operations for overlay maintenance and discuss their effectiveness in dealing with highly dynamic environments.

The remainder of this paper is structured as follows. Section 2 outlines Nemo's approach and presents its design and operational details. We have performed an extensive evaluation of our pro-

tocol through simulation and wide-area experimentation. We describe the experimental setups in Sec. 3 and report our results in Sec. 4. Section 5 discusses related work and Sec. 6 concludes.

2 Nemo's Approach

All peer-to-peer or application layer multicast protocols organize the participating peers in two topologies: a control topology for group membership related tasks and a delivery tree for data forwarding. Available protocols can be classified according to the sequence they adopt for their construction [3, 16]. Nemo follows the *implicit approach* [4, 12, 33, 44, 36] to building an overlay for multicasting: participating peers are organized in a control topology and the data delivery network is implicitly defined based on a set of forwarding rules which we will describe in the following paragraphs.

Nemo organizes the set of communication peers into clusters, where every peer is a member of a cluster at the lowest layer. Each of these clusters selects a *leader* that becomes a member of the immediate superior layer. The process is thus repeated, with all peers in a layer being grouped into clusters from where leaders are selected to participate in the next higher layer. Hence peers can lead more than one cluster in successive layers of this logical hierarchy.

The formation of clusters at each level of the hierarchy is currently based on network proximity, although other factors such as bandwidth [37, 15] and expected peer lifetime [10] could be easily incorporated. Clusters vary in size between k and $3k - 1$, where k is a constant known as *degree*. [1]

Our work focuses on improving the resilience of peer-to-peer overlay multicast systems. Before discussing the details of our approach, in the following paragraphs we explain the dynamics of our basic tree-based protocol, such as the joining and departure of peers, as well as Nemo's probabilistic approach to overlay maintenance.

2.1 Member Join and Departure

A new peer joins the multicast group by querying a well-known node, the rendezvous point, for the IDs of the members on the top layer. Starting there and in an iterative manner, the incoming peer continues: (i) requesting the list of members at the current layer from the cluster's leader, (ii) selecting from among them who to contact next based on the result from a given cost function, and (iii) moving into the next layer. When the new peer finds the leader with minimal cost at the bottom layer, it joins the associated cluster.

Members can leave Nemo in announced (graceful) or unanounced manner. Since a common member has no responsibilities towards other peers, it can simply leave the group after informing its cluster's leader. On the other hand, a leader must first elect replacement leaders for all clusters it owns; it can then leave its top layer after informing its cluster's leader.

[1] This is common to both Nemo and Nice [4]; the bounds have been chosen to help reduce the degree of oscillation in clusters.

To detect unannounced leaves, Nemo relies on heartbeats exchanged among the cluster's peers. Unreported members are given a fixed time interval, or *grace period*, before being considered dead. Once a member is determined dead, a repair algorithm is initiated. If the failed peer happens to be a leader, the tree itself must be fixed: the members of the victim's cluster must elect the replacement leader from among themselves.

To deal with dynamic changes in the underlying network, every peer periodically checks the leaders of the next higher layers and switches clusters if another leader is closer than the current one (thresholds are used to prevent oscillation). Additionally, in a continuous process of refinement, every leader checks its highest owned cluster for better suited leaders and transfer leadership if such a peer exists.

2.2 Splitting and Merging Clusters

Due to membership changes, clusters may grow/shrink beyond the cardinality bounds defined by the clusters' degree; such clusters must be dealt with to guarantee the hierarchical properties of the protocol. Undersized clusters are merged with others while oversized ones are split into two new ones. Both split and merge operations are carried on by the cluster's leader.

If the size of a cluster falls bellow its lower bound, the cluster leader redirects all cluster members to the leader of a neighbor cluster and, once the move has completed, it removes itself from the (all) next higher layers.

When a cluster's cardinality exceeds its upper bound, the cluster leader must split it into two new clusters, each of size at least $3k/2$. The two new clusters are defined so to minimize the total sum of path latencies (i.e. the cluster's diameter). For this we employ a genetic algorithm [21] that, although it may not yield as good an optimization as alternative approaches (such as exhaustive search), it produces near optimal results at a fraction of their cost (the algorithm runs in cn^2 time, with a small constant c, where n is the size of the cluster). Finally, instead of selecting a leader for each of the newly created clusters, the current leader automatically becomes the leader for one of the emerging clusters, thus reducing the control overhead of leader demotion/promotion.

Cluster split and merge operations can be triggered by each arrival/departure event or invoked through a periodic cluster-check operation. While the triggered approach strictly guarantees cluster sizes, it can result in a significant performance overhead under high frequency membership changes. Part of the problem stems from the time it takes for a new structure to be established (due to, among other factors, propagation delays). During this period of time, joining nodes may face difficulties finding their appropriate cluster. We opt for a periodic cluster-check operation as this limits the rate of cluster changes and their associated costs. On the other hand, as a result of our periodic approach, the cardinality of some clusters may temporarily lay outside the bounds stated by their degree.

2.3 A Probabilistic Approach to Overlay Maintenance

Proactive recovery, where a system tries to react immediately to membership changes, adds additional stress to an already-stressed network [34]. Nemo relies on a set of periodic algorithms for overlay maintenance in order to avoid congestion collapse, but adopts a probabilistic approach to reduce the load on a possible stressed system.

In Nemo, some of the most costly maintenance operations, such as splitting, merging and refinement, are only executed with some probability or, alternatively, deferred to the next interval. We refer to them as *periodic probabilistic operations*. In the presence of high churn, many of these operations can not only be deferred, but completely avoided as follow-up changes may revert a previously triggering situation. For example, a stream of member departures/failures may leave a cluster undersized, triggering a relatively expensive merge operation. If deferred, a series of subsequent member joins may push the undersized cluster's cardinality beyond its merge-triggering bound.

In another example of the probabilistic approach to overlay maintenance, crew lists are distributed by the leader with some probability at the beginning of an epoch. The probability and the epoch length are chosen such that the mean time of crew list exchanges follows an exponential distribution with a specified mean time. Changes to the crew list always force an update operation. This guarantees that all peers store recent crew lists with high probability.

2.4 Planning for Node Failure

Tree-based overlay multicast protocols have proven to be highly scalable and efficient in terms of physical link stress, state and control overhead, and end-to-end latency [23, 4, 14]. As other tree-based structures, however, these proposed protocols have an inherent problem of resilience from their dependence on the reliability of non-leaf nodes. The high-degree of transiency of end-systems has been pointed out as one of the main challenges for these architectures [7].

Nemo addresses the resilience issue of tree-based systems through the introduction of co-leaders. Every cluster leader recruits a number of co-leaders with whom it forms the *crew*. Crew lists are periodically distributed to the cluster's members. Co-leaders improve the resilience of the multicast group by avoiding dependencies on single nodes and providing alternative paths for data forwarding. In addition, crew members share the load from message forwarding, thus improving scalability. Figure 1 illustrates the logical organization of Nemo.

2.5 Data Forwarding

Nemo's data delivery topology is implicitly defined by the set of packet-forwarding rules adopted. A peer sends a message to one of the leaders for its layer. Leaders (the leader and its co-leaders) forward any received message to all other peers in their clusters and up to the next higher layer. A node in charge of forwarding a packet to a given cluster must select the destination peer among all crew members in the cluster's leader group. The algorithm is summarized in Fig. 2.

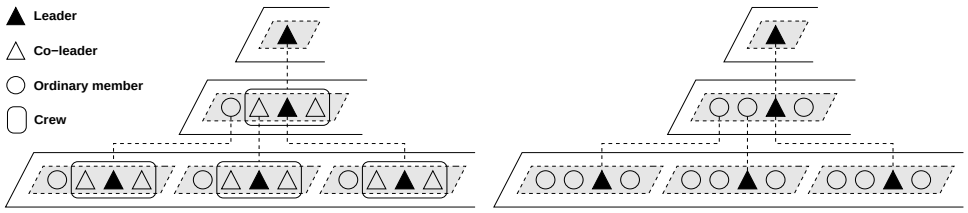


Figure 1: Nemo's logical organization on the left side compared to the logical organization of a basic tree-based overlay (e.g. Nice) on the right side. The shape illustrates only the role of a peer within a cluster: a leader of a cluster at a given layer can act as co-leader or ordinary member at the next higher layer.

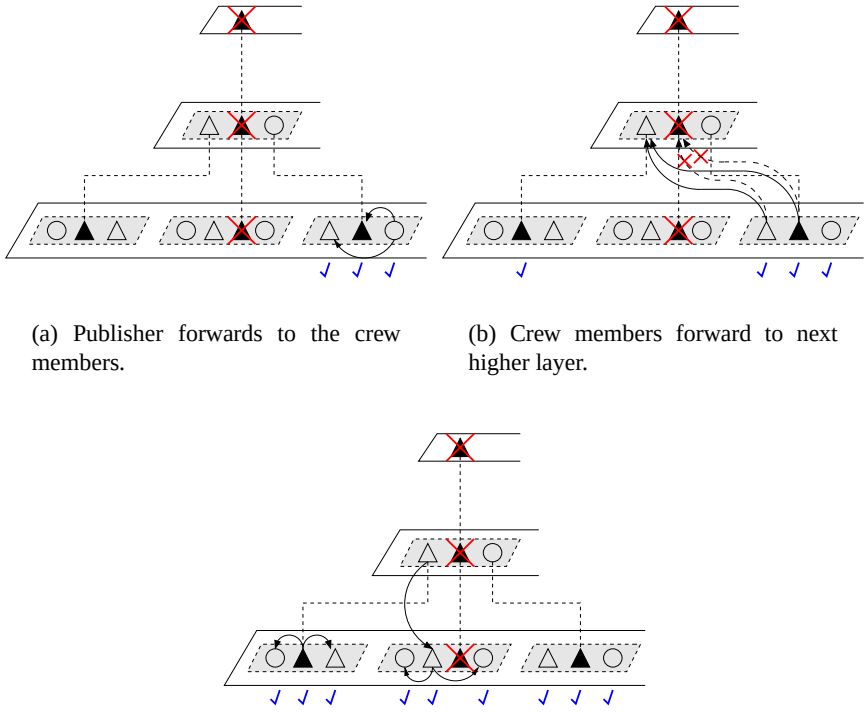
```
FORWARD-DATA(msg)
 1  R ← ∅
 2  if leader ∉ msg.sender_crew
 3    then R ← R ∪ leader
 4  for each child in children
 5  do if child ∉ msg.sender_crew
 6        then R ← R ∪ child
 7  SEND(msg, R, sender_crew ← crewOf(self))
 8  if isCrewMember(self) and leader ∉ msg.sender_crew
 9    then R ← ∅
10         R ← R ∪ super_leader
11         for each neighbor in neighbors
12         do R ← R ∪ neighbor
13         SEND(msg, R, sender_crew ← crewOf(leader))
```

Figure 2: Data Forwarding Algorithm: SEND transmits a packet to a list of nodes, selecting the real destination among the crew members associated with the given destination.

Figure 3 shows an example of the forwarding algorithm in action and illustrates Nemo's resilience under different failure scenarios. The forwarding responsibility is evenly shared among the leaders by alternating the message recipient among them. In case of a failed crew member, the remaining (co-)leaders can still forward their share of messages in the tree. Lost messages are detected and recovered, as described in the following section, thus achieving high data delivery ratios even in the presence of crew members' failures.

2.6 Data Retransmission

As other protocols aiming at high resilience [31, 5], Nemo relies on sequence numbers and



(a) Publisher forwards to the crew members.

(b) Crew members forward to next higher layer.

(c) Co-leader forwards to all other clusters, including its own.

Figure 3: Data forwarding in Nemo with a failed node: All nodes are able to receive the forwarded data despite a node failure. Note how a sender alternates the packet destination among the crew members.

triggered NACKs to detect lost packets.

Every peer piggybacks a bit-mask with each data packet indicating the previously received packets. In addition, each peer maintains a cache of received packets and a list of missing ones. Once a gap (relative to a peer's upstream neighbors) is detected in the packet flow, the absent packets are considered missing after some fixed period of time. This time is selected to reduce the effect of jitter and processing delays. [2] For each missing packet, a NACK is sent to a peer caching it, and a different timeout is set for retransmission.

The peer requesting the retransmission may not be the only one missing the packet. Ideally, the requesting peer should forward the recovered packet to all other known peers (and only to those)

[2] In our current implementation this time is set to 2 `RTT` to the farthest crew member.

```
RECOVER-DATA(msg)
1  d ← DOWNWARD ∪ UPWARD
2  for each packet in recent_packets
3  do if isSameStream(packet, msg)
4        then if isChild(packet.sender_crew)
5                then d ← d ∩ ¬ UPWARD
6             if isLeader(packet.sender_crew)
7                then d ← d ∩ ¬ DOWNWARD
8  FORWARD(msg, direction ← d)
```

Figure 4: Data Recovery Algorithm: The function FORWARD forwards a packet as specified by a logical direction (UPWARD and/or DOWNWARD).

who need it. To achieve this we have designed an efficient algorithm that, without incurring additional control traffic, helps us improve latency in delivery for retransmitted packets while reducing the number of duplicates. The algorithm takes advantage of the fact that, over a reasonably small window of time, a peer sees the packets from one source flowing in one logical direction.

After a peer has detected a lost packet, it initiates the recovery protocol and chooses the direction in which to forward the recovered packet, if at all. No forwarding would be required if all other peers have successfully received the packet. Otherwise, the forwarding direction will be decided based on the flow direction of the follow-up packets: *upward*/*downward* if the source is logically below/above the recovering peer. The algorithm helps reduce the number of duplicate packets generated indirectly by packet losses. A sketch of the algorithm is provided in Fig. 4.

3 Evaluation

We analyze the performance of Nemo using detailed simulation and wide-area experimentation. We compare Nemo's performance to that of three other protocols – Narada [16], Nice [4] and Nice-PRM [5] – both in terms of application performance and protocol overhead.

It is worth noting that the implementation of all evaluated protocols is shared between the wide-area and the simulation experiments. We achieve this by abstracting the protocols' logic from the environment-dependent functionality through well-defined interfaces. Thus, for any given protocol, the difference in the code-base of the wide-area and the simulator implementations is limited to how the communication between two peers is realized.

The remainder of this section describes the performance metrics employed, discusses implementation details of the compared protocols and describes our evaluation setup. Section 4 presents the results for both simulation and wide-area experiments.

3.1 Performance Metrics

We evaluate the effectiveness of the different protocols in terms of performance improvements to the application and protocol's overhead. Application performance is captured by delivery ratio, end-to-end latency and connectivity, while overhead is evaluated in terms of number of duplicate packets per sequence number and control-related traffic.

- *Delivery Ratio:* Ratio of subscribers which have received a packet within a fixed time window. Disabled receivers are not accounted for.
- *Latency:* End-to-end delay (including retransmission time) from the source to the receivers, as seen by the application. This includes path latencies along the overlay hops, as well as queueing delay and processing overhead at peers along the path.
- *Connectivity:* Percentage of nodes that have received at least one packet in a 10-sec. interval. That is, time is divided in 10-sec. intervals and connectivity is determined for each of those intervals.
- *Duplicate Packets:* Number of duplicate packets for all receivers counted per sequence number, reflecting an unnecessary burden on the network. Packets arrived outside of the delivery window are accounted for as duplicates, since the receiver already assumed them as lost.
- *Control-Related Traffic:* Total control traffic in the system, in MB per second, during the failure interval. We measure the total traffic during the observation interval by accounting packets at the router leve.

3.2 Details on Protocol Implementations

For each of the three alternative protocols, the values for the available parameters were obtained from the corresponding literature.

For Narada [16], the number of directly connected peers (fanout) is set to six and may approach 12 for a short period of time. The distance vectors, the set of shortest-paths latencies to all other peers, are exchanged in 10-sec. intervals. The timeout for detecting dead members is set to 60 sec., and the one for mesh partition repairs to 50 sec.

For Nice [4], heartbeats are sent at 10-sec. intervals. The cluster degree, k, is set to 3. The grace period for dead neighbor detection is set to 15 sec.

Nice-PRM is implemented as described in [5, 6]. We used PRM-(3,0.01), PRM-(3,0.02) and PRM-(3,0.03) with three random peers chosen by each node, and with one, two, and three percent forwarding probability. Discover messages to locate random overlay nodes are sent with 5-sec. intervals and a time-to-live (TTL) of 5 hops.

For Nemo, the cluster degree and the crew size are set to three. The grace period is set to 15 sec. and the mean time between crew list exchanges is set to 30 sec.

For the wide-area implementation, we employ UDP with retransmissions: ten attempts for heartbeats and five for all other control traffic. Data communication does not employ retransmission.

Our implementations of the alternative protocols closely follow the descriptions from the literature, and have been validated by contrasting our results with the published values. However, there are a number of improvements to the common algorithms, such as the use of periodic probabilistic operations, that while part of Nemo were made available to all protocols in our evaluations. The benefits from these algorithms help explain the performance improvements of the different protocols with respect to their original publications [16, 4, 5]. We have opted for this approach to isolate the contribution of PRM and co-leaders to the overall resilience of the multicast protocols.

For Nice, Nice-PRM and Nemo, we check the clusters every second, but limit the minimal time between maintenance operations. Assuming that the triggering conditions are satisfied (an undersized cluster, for example) and that there has not been a maintenance operation within the last 5 sec., a merge operation is executed with 1% probability, a split operation with 100% probability, and a refinement operation with 1% probability. When the cluster's cardinality falls bellow its lower bound, we reduce the probability of execution of the refinement operation to 0.1% in an additional attempt to avoid an expensive merge.

3.3 Experimental Setup

We performed our evaluations through detailed simulation using a locally written, packet-level, event-based simulator and wide-area experimentation on PlanetLab, an internationally-deployed test bed.

We ran our simulations using Inet [24], Transit-Stub and AS Waxman [43] topologies. In this paper we present simulation results based on Inet topologies with 3,072 and 8,192 nodes and a multicast group of 128, 256, 512, 1024, 2048 and 4096 members. [3] Members are randomly attached to routers, and a random delay of between 1 and 4 ms is assigned to every link.

Each simulation experiment lasts for 40 minutes (simulation time). All peers join the multicast group by contacting the rendezvous point at uniformly distributed, random times within the first 200 sec. of the simulation. A warm-up time of 300 sec. is omitted from the figures. Starting at 600 sec. and lasting for about 1200 sec., each simulation has a phase with rapid membership changes. We exercise each protocol under two different failure rates during this phase. Under a high-failure rate, nodes fail independently at a time sampled from an exponential distribution (with mean, *Mean Time To Failure*, equal to 5 min.) to rejoin shortly after (time sampled from an exponential distribution with mean, *Mean Time To Repair*, equal to 2 min.) [5]. The two means are chosen asymmetrically to allow, on average, 5/7 of all members to be up during this phase. We also run the same set of simulations with a lower failure rate given by a MTTF of 60 min. and a MTTR of 10 min. [38].

[3] Comparable results were obtained with the different topologies and group sizes. For the complete set of results please see [8].

For the wide-area experiments we restrict our comparison to Nice, Nice-PRM and Nemo with between 60 and 200 members distributed across ∼90 sites. The number of members per site varies from 1 to ∼12, with most sites having two members. As in our simulation experiments, we inject failures at the low and high failure rate (MTTF = 60 and 5 min. and MTTR = 10 and 2 min., respectively). Each protocol runs during 30 min. with a 10 min. failure injection. The order of the three protocols is randomly chosen.

For the simulation, we generate a failure event sequence based on the above MTTF and MTTR. The same sequence is used for all protocols and all runs. In the wide-area experiment, the failure events are drawn from a exponential distributed random number generator with mean equal to MTTF and MTTR.

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 [28] and leads to sufficient accuracy for our application.

In all experiments, we model a single source multicast stream to a group. The source sends constant bit rate (CBR) traffic of 100 B payload at a rate of 10 packets per second. The buffer size is set to 32 packets, which corresponds to the usage of a 3.2-second buffer, a realistic scenario for applications such as multimedia streaming.

4 Experimental Results

This section presents and discusses our results from both simulation and wide-area evaluations. Simulation-based experiments allow us to analyze the scaling properties of each approach and to better understand their behavior under controlled, reproducible settings. Wide-area evaluations help us understand how the different protocols behave in dynamic and unpredictable Internet environments, and give us an idea of the end-to-end performance experienced by applications.

Before discussing the simulation and wide-area experimental results, we digress briefly to evaluate the trade-offs of different crew sizes for Nemo.

4.1 Determining the Right Crew Size

Nemo introduces the concept of co-leaders and crews in order to improve the resilience of the multicast group. Leaders from every cluster recruit co-leaders to form their crew and distribute the crew composition to the cluster's members. Co-leaders help improve the resilience of the multicast group by avoiding dependencies on single nodes and providing alternative paths for data forwarding. With shared forwarding responsibility, crew members also share network load, making larger crews more attractive for high (and fair) load distribution.

The size of a crew is upper-bounded by the cluster size, which is defined by its degree k and may range from k to $3k - 1$ nodes. In Banerjee et al. [4], the authors used 3 as a reasonable choice for the cluster degree. The cluster size is the guaranteed number of peers in a cluster, thus a crew

Table 1: Crew Size (512 end hosts, high failure rate).

Nemo's Crew Size	Delivery ratio		Connectivity		Duplicate		
	Mean	Std	Mean	Std	Mean	Max	Std
Nemo-C1	0.992	1.95E-3	0.9975	1.02E-3	7.10	8.32	0.71
Nemo-C2	0.997	0.81E-3	0.9996	0.19E-3	3.01	3.69	0.31
Nemo-C3	0.998	0.89E-3	0.9998	0.21E-3	3.16	3.77	0.29
Nemo-C4	0.998	0.52E-3	0.9998	0.21E-3	3.67	4.18	0.27

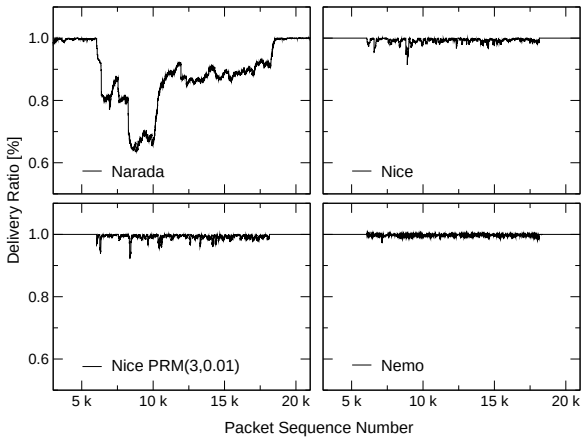


Figure 5: Delivery ratio (512 end hosts, high failure rate).

size larger than the degree cannot be enforced in all clusters.

In trying to determine the ideal crew size, we experimented with different values through simulation. Table 1 compares different alternatives in terms of delivery ratio, number of duplicates and degree of connectivity obtained from twenty-five simulations per crew size. As is shown in the table, an increase of the crew size gives a performance advantage with near equal or less overhead in terms of duplicate packets. This advantage becomes less pronounced as the crew size exceeds the cluster degree.

The results reported in the remainder of this section are all based on a crew size of 3, since this seems to offer a good balance in terms of delivery ratio, connectivity and rate of duplicates.

4.2 Simulation Results

For our simulation-based evaluation we employ SPANS, a locally written, packet-level simulator and a variety of generated topologies. Here we report results of twenty-five runs per protocol obtained with Inet topologies.

Table 2: High-Failure Rate (512 end hosts).

Protocol	Delivery ratio		Connectivity		Duplicate packets			
	Mean	Std	Mean	Std	Mean		Max	Std
Nemo	0.998	0.89E-3	0.9998	0.21E-3	3.16	(45%)	3.77	0.29
Nice-PRM(3,0.01)	0.993	1.25E-3	0.9989	0.28E-3	12.47	(175%)	14.43	1.04
Nice-PRM(3,0.02)	0.994	1.23E-3	0.9993	0.22E-3	18.20	(256%)	19.75	0.77
Nice-PRM(3,0.03)	0.994	1.01E-3	0.9993	0.17E-3	24.22	(341%)	28.14	1.82
Nice	0.992	1.95E-3	0.9975	1.02E-3	7.10	(100%)	8.32	0.71
Narada	0.852	60.3E-3	0.8534	57.9E-3	0.00	(0%)	0.00	0.00

Table 3: Low Failure Rate (512 end hosts).

Protocol	Delivery ratio		Connectivity		Duplicate packets			
	Mean	Std	Mean	Std	Mean		Max	Std
Nemo	1.000	0.12E-3	1.0000	0.02E-3	0.34	(26%)	0.59	0.09
Nice-PRM(3,0.01)	0.999	0.56E-3	0.9999	0.12E-3	6.42	(498%)	6.98	0.38
Nice-PRM(3,0.02)	0.999	0.36E-3	0.9999	0.07E-3	12.00	(930%)	13.43	0.66
Nice-PRM(3,0.03)	0.999	0.27E-3	0.9999	0.05E-3	16.74	(1298%)	18.42	0.65
Nice	0.999	0.52E-3	0.9998	0.16E-3	1.29	(100%)	1.92	0.40
Narada	0.950	38.3E-3	0.9548	36.2E-3	0.00	(0%)	0.00	0.00

Figure 5 shows the average delivery ratios per sequence number of all runs, for the four protocols evaluated, under high failure rate. The first graph shows the delivery ratio of Narada, followed by Nice, Nice PRM(3,0.01) and Nemo. Nemo's high delivery ratio is due to the alternate data paths offered by crew members, as they enable the early detection and retransmission of lost packets within the allowed timeout interval. The delivery ratio of Narada suffers most from the special characteristic of the failure event sequence we used for this simulation.

Tables 2 and 3 summarize the results for both high and low failure rates, respectively. Nemo, Nice-PRM(3,0.02) and Nice-PRM(3,0.03) achieve comparably high delivery ratio, significantly better than Nice and Narada.

The second set of columns in Tables 2 and 3 show the connectivity achieved by the compared protocols. The reported connectivity results presented are computed for a time interval of 10 sec., which is less than the specified grace period. Ideally, we aim at 100% connectivity. In both cases, Nemo offers the highest connectivity and the difference is more significant under high failure rate.

The cost of a resilient multicast protocol can be measured in terms of duplicate packets. The third set of columns in Tables 2 and 3 show this overhead for the compared protocols.

Nemo features the highest delivery ration of 99.8% under high failure rate. Nice and Nice-PRM achieve a delivery ratio of between 99.2% and 99.4%. Previous work [5] accounted link losses in simulation and reported a similar delivery ratio for Nice-PRM, but a substantial lower delivery ratio for Nice without retransmission. Our results clearly show that without losses at the network layer, the effectiveness of PRM is only minor for a protocol such as Nice. In terms of overhead, Nemo's approach results in the lowest number of duplicates (3.16 per sequence number) even outperforming

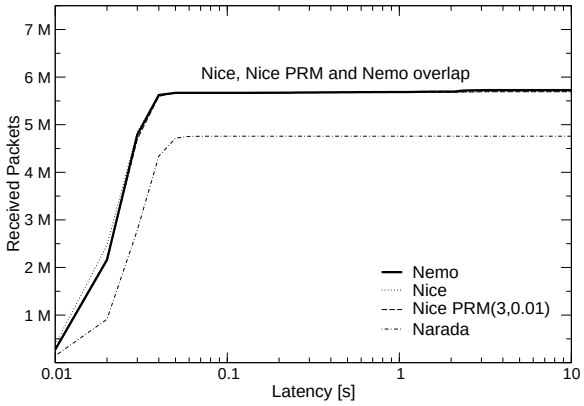


Figure 6: Latency CDF (512 end hosts, high failure rate).

Nice (7.10). Nice-PRM has a significantly higher cost than both Nice and Nemo with up to 24.22 duplicates per sequence number, partially due to its random forwarding component.

Under low failure rate Nemo achieves nearly perfect delivery of 100%, while incurring an overhead of just 0.34 duplicates per sequence number. Similar to the case with high-failure rate, Nice and Nice-PRM achieve a slightly lower delivery ratio (99.9%), but at substantially higher costs, 1.29 and 16.74 duplicates per sequence number respectively. Nemo's multiple crew members and its retransmission approach help explain these improvements, as the former results in alternate paths that increase the effectiveness of NACKs and the latter ensures delivery to other peers (potentially) missing the packet, thus reducing the total amount of retransmissions.

Under both failure rates, Nemo's approach results in comparably high delivery ratios at significantly lower cost in terms of duplicates. Nice-PRM incurs a higher number of duplicate packets than Nemo and Nice as a result of PRM's proactive randomized forwarding. As can be seen from the alternative PRM configurations, the number of duplicate packets correlates with its delivery ratio.

Nemo's higher resilience with low overhead comes at no cost in terms of latency, as can be observed in Fig. 6. The graph shows the cumulative distribution function (CDF) of latency for all received packets under high failure rate and a buffer of 32 packets, which corresponds to a 3.2-second delay. Nemo introduces alternative paths to improve resilience. These alternative paths, with delays higher than or equal to the optimal and only choice in Nice, might introduce some latency penalties for packet delivery. As can be observed in the graph, the advantage of delivering more packets outweighs the potential latency cost, and Nemo suffers no noticeable additional delays for packets delivered without retransmission as seen on the left side of the plot.

Table 4 summarizes the results for different group sizes. [4] Nemo's approach scales linearly

[4]Due to Narada's $O(n^2)$ memory consumption for simulations, we were not able to obtain data for Narada with group sizes larger than 1024 peers. A Narada peer stores information for every other peer in the group, whereas Nemo, Nice

Table 4: Scalability (8192 routers, high failure rate). Nice-PRM is configure with 3 random peers and 2% probability (*PRM(3,0.02*)).

Size	Delivery ratio				Duplicate packets				Control Traffic [MBps]			
	Nemo	Nice-PRM	Nice	Narada	Nemo	Nice-PRM	Nice	Narada	Nemo	Nice-PRM	Nice	Narada
128	0.999	0.995	0.996	0.957	0.43	4.79	1.24	0.00	0.12	0.18	0.12	0.74
256	1.000	0.997	0.991	0.863	0.41	7.63	2.04	0.00	0.25	0.32	0.24	3.12
512	0.999	0.998	0.995	0.882	1.30	14.70	4.70	0.00	0.52	0.69	0.48	13.07
1024	0.999	0.997	0.994	0.848	2.78	28.44	9.33	0.00	1.03	1.37	0.95	54.44
2048	0.999	0.995	0.995	–	5.70	63.01	15.82	–	2.03	2.69	1.99	–
4096	0.999	0.996	0.993	–	10.95	120.40	34.05	–	4.24	5.36	3.78	–

with the group size while maintaining a very high delivery ratio. The cost per receiver is constant for Nemo, Nice and Nice-PRM, independent of the group size, showing the scalability of the hierarchical approach. Narada suffers from its extremely high control traffic for larger group sizes. The control traffic given in the table corresponds to the total control traffic in the system during the failure interval, with only $5/7$ of the total population up and running. With a group size of 1024 peers, Nemo has 1.41 KBps (KBytes per second) overhead per node, lower than Nice-PRM's control overhead of 1.87 KBps. Nice has the lowest control overhead of 1.29 KBps, while Narada incurs a control traffic of 74.4 KBps per node. Note that control traffic omits duplicate packets, which are accounted for separately, but includes all parts of control traffic (such as header sizes), which might explain the slight overhead cost when contrasted with previously published results [4].

4.3 Wide-Area Results

The results presented here are based on twenty-five runs, each a set of one experiment per protocol. At every step we schedule, in a random order, one experiment for each of the protocols evaluated. During the experiments the controller only sent control traffic to inject failures on the peers over the network. The results were stored locally at the peers and fetched by the controller at the end of the experiment before the evaluation of the next protocol started. Once one step was completed, the next step was initiated until twenty-five runs each were complete. Given this procedure, we took several measurements at different times of the day.

In this section we present representative graphs for Nemo, Nice and Nice-PRM(3,0.02). We chose 2% forwarding probability for Nice-PRM, since this offers the best tradeoff between high delivery ratio and low number of duplicate packets.

Figure 7 shows the delivery ratio of one run each; the packet losses observed during the warm-up interval are due to the non-deterministic influence of the environment. The graphs confirm the

and Nice-PRM peers only store $O(\log n)$ data entries.

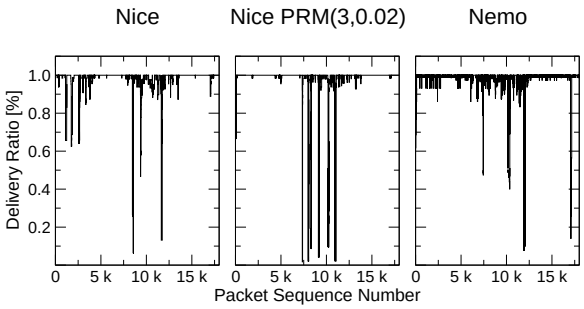


Figure 7: Delivery ratio (PlanetLab, ∼72 end hosts, high failure rate).

Table 5: Wide-Area Results (PlanetLab, ∼72 end hosts). The statistics include the packets with sequence numbers from 6,000 to 12,000. Nice-PRM is configure with 3 random peers and 2% probability (*PRM(3,0.02)*).

Protocol	High Failure Rate						Low Failure Rate					
	Delivery ratio		Duplicate packets				Delivery ratio		Duplicate packets			
	Mean	Std	Mean		Max	Std	Mean	Std	Mean		Max	Std
Nemo	0.979	0.010	1.27	(120%)	2.53	0.56	0.996	0.003	0.52	(173%)	1.10	0.27
Nice-PRM	0.953	0.024	2.02	(191%)	3.00	0.57	0.997	0.003	1.37	(457%)	1.68	0.15
Nice	0.939	0.032	1.06	(100%)	1.83	0.47	0.991	0.011	0.30	(100%)	1.22	0.23

previously obtained simulation results.

In Table 5 we summarize the runs of the wide-area experiments on PlanetLab with a group size of about 72 peers. The improvements that PRM brings to Nice's resilience become clear in this more realistic setting (with paths experiencing losses, for example). Still, Nemo outperforms both Nice (93.9%) and Nice-PRM (95.3%) with a delivery ratio of 97.9% under high failure rates. In terms of duplicates per sequence number, Nemo's cost of 1.27 is substantially lower than that of Nice-PRM (2.02) and comparable to that of Nice. Clearly, the number of duplicates incurred by a protocol is positively correlated with its delivery ratio (a lower delivery ratio results in a lower rate of duplicates for any protocol). When taken this into consideration, Nemo's relative savings in terms of duplicates packets under high failure rate become more significant. Under low failure rate the three protocol offer similar delivery ratio, although Nemo achieves this with a substantially lower rate of duplicate packets (0.52) when compared with Nice-PRM (1.37).

We show the latency distribution achieved in one of the wide-area experiment in Fig. 8. The data sets correspond to the plots shown in Fig. 7. The three graphs confirm the data gathered through simulation (although the experienced latencies are slightly higher in the wide-area experiment). Nemo outperforms Nice in the latency of packets requiring retransmission, as Nemo's alternate data paths translate into earlier packet-loss detection and faster recovery. We see that the slope of the plot starting at 0.2 seconds latency is steeper to the right for Nemo, due in part to recovered packets.

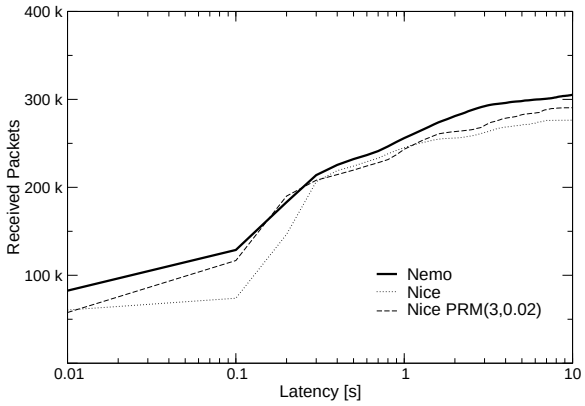


Figure 8: Latency CDF (PlanetLab, ∼72 end hosts, high failure rate).

Table 6: Wide-Area Results (PlanetLab, ∼130 end hosts). The statistics include the packets with sequence numbers from 6,000 to 12,000. Nice-PRM is configure with 3 random peers and 2% probability (*PRM(3,0.02*)).

Protocol	High Failure Rate						Low Failure Rate					
	Delivery ratio		Duplicate packets				Delivery ratio		Duplicate packets			
	Mean	Std	Mean		Max	Std	Mean	Std	Mean		Max	Std
Nemo	0.964	0.029	2.19	(172%)	3.62	0.99	0.982	0.010	2.27	(214%)	6.57	1.36
Nice-PRM	0.933	0.024	3.77	(297%)	6.40	1.50	0.981	0.019	3.44	(325%)	6.04	1.07
Nice	0.914	0.070	1.27	(100%)	2.72	0.80	0.974	0.030	1.06	(100%)	2.15	0.44

In Table 6 we summarize the runs of the wide-area experiments on PlanetLab with a larger group size of about 130 peers. Compared to the results with the smaller group size, we see that the delivery ratio is lower for all three protocols. Nemo has the highest delivery ratio under low and high failures. The cost in terms of duplicate packets is larger than we would expect by looking at the smaller group size results. This is in part due to PlanetLab's changing characteristics. First, we use a larger and different set of nodes for the second group size experiments. Second, PlanetLab is a shared environment, where experiments compete for resources.

5 Related Work

All peer-to-peer or application-layer multicast protocols organize the participating peers in two topologies: a control topology for group membership related tasks, and a delivery tree for data forwarding. Available protocols can be classified according to the sequence adopted for their con-

struction [3, 16]. In a tree-first approach [20, 23, 32], peers directly construct the data delivery tree by selecting their parents from among known peers. Additional links are later added to define, in combination with the data delivery tree, the control topology. With a mesh-first approach [16, 13], peers build a more densely connected graph (mesh) over which (reverse) shortest path spanning trees, rooted at any peer, can be constructed. Protocols adopting an implicit approach [4, 12, 33, 44] create only a control topology among the participant peers. Their data delivery topology is implicitly determined by the defined set of packet-forwarding rules.

Banerjee et al. [4] introduce Nice and demonstrate the effectiveness of overlay multicast across large scale networks. The authors also present the first look at the robustness of alternative overlay multicast protocols under group membership changes. Nemo adopts the same implicit approach, and its design draws a number of ideas from Nice such as its hierarchical control topology. Nemo introduces co-leaders to improve the resilience of the overlay and adopts a periodic probabilistic approach to reduce/avoid the cost of membership operations.

A large number of research projects have addressed reliable and resilient multicast at the network layer [31, 39, 41, 30, 26, 19]. A comparative survey of these protocols is given in [25, 35]. Like many of them, Nemo relies on reactive techniques to recover from packet losses. STORM [40] uses hierarchical NACKs for recovery: NACKs are sent to parents (obtained from a parent list) until the packet is successfully recovered or deemed obsolete. In the case of Nemo, NACKs are used only to request missing packets from neighboring peers who are known to cache them locally.

In the context of overlay multicast, a number of protocols have been proposed aiming at high resilience [5, 11, 36, 29]. ZigZag [36] is a single source P2P streaming protocol. Resilience is achieved by separating the control and data delivery trees at every level, with one peer being held responsible for the organization of the sub-tree and a second one dealing with data forwarding. In the presence of failures, both peers share repair responsibilities. In Nemo, the forwarding responsibility of a peer is shared among its crew members and its repair algorithm is fully distributed among cluster members. PRM [5] uses randomized forwarding and NACK-based retransmission to improve resilience. In contrast, Nemo relies on the concept of a *crew* and opts only for deterministic techniques for data forwarding. SplitStream [11] and CoopNet [29] improve resilience by building several disjoint trees. In addition, CoopNet adopts a centralized organization protocol and relies on Multiple Description Coding (MDC) to achieve data redundancy. Nemo is a decentralized peer-to-peer multicast protocol which offers redundancy in the delivery path with only a single control topology through the use of leaders and co-leaders. We are exploring the use of data redundancy using forward error correction (FEC) encoding [9].

RON [1] introduces alternate paths to recover from path outages and periods of degraded performance. It routes around failures by using intermediate hops. Nemo adopts a similar approach, as its forwarding algorithm chooses among several alternate paths for each packet. oStream [42] takes advantage of the strong buffering capabilities of end hosts to provide asynchronous streaming multicast. Nemo relies on these same capabilities of peers in order to support the local retransmission of lost packets.

In the context of structured peer-to-peer overlay networks, [27] proposes to dynamically adjust

protocol parameters, such as heartbeat intervals and grace periods, based on the operating conditions. Similar to Nemo, it tries to reduce the maintenance cost without failures while still providing high resilience. Nemo's approach differs in the way that it uses a static low cost algorithm, which handles an increased level of churn with comparatively low cost. Nemo's refinement algorithm could potentially benefit from their technique by adjusting the refinement interval based on the experienced membership change rate and the measured dynamics of the underlying physical network, thus reducing the total control overhead.

6 Conclusions

We have described Nemo, an overlay multicast protocol capable of attaining high delivery ratio without its associated costs. Through the introduction of co-leaders and the use of triggered NACKs, Nemo achieves its resilience goals, without sacrificing end-to-end latency or incurring additional costs in terms of duplicate packets or control-related traffic. We have demonstrated the effectiveness of our approach through simulation and wide-area experimentation under different stress scenarios.

Nemo's probabilistic approach has proved to be effective in reducing the negative effects of high levels of churn, while still providing an asymptotic convergence to an optimally stable state. Wide-Area experimentations show that Nemo achieves high delivery ratio ($> 96.4\%$) with a small buffer size (3.2 seconds) in a real-world environment with $\sim$130 peers and a high churn rate (> 37 joins and failures per minute), while maintaining comparatively low overhead.

Acknowledgments

We would like to thank Karsten Schwan and Peter Dinda, who kindly loaned us their equipment for some of our experiments. We are also grateful to Jeanine Casler, Yi Qiao, Jason Skicewicz, Ananth Sundararaj, Dong Lu and Ashish Gupta for their helpful comments on early drafts of this paper.

References

[1] ANDERSEN, D., BALAKRISHNAN, H., KAASHOEK, F., AND MORRIS, R. Resilient overlay networks. In *Proc. of the 18th ACM SOSP* (October 2001).

[2] ANDERSON, T., SHENKER, S., SOTICA, I., AND WETHERALL, D. Design guidelines for robust Internet protocols. In *Proc. of HotNets-I* (October 2002).

[3] BANERJEE, S., AND BHATTACHARJEE, B. A comparative study of application layer multicast protocols, 2002. Submitted for review.

[4] BANERJEE, S., BHATTACHARJEE, B., AND KOMMAREDDY, C. Scalable application layer multicast. In *Proc. of ACM SIGCOMM* (August 2002).

[5] BANERJEE, S., LEE, S., BHATTACHARJEE, B., AND SRINIVASAN, A. Resilient multicast using overlays. In *Proc. of ACM SIGMETRICS* (June 2003).

[6] BANERJEE, S., LEE, S., BRAUD, R., BHATTACHARJEE, B., AND SRINIVASAN, A. Scalable resilient media streaming. Tech. Report UMIACS TR 2003-51, U. of Maryland, 2003.

[7] BAWA, M., DESHPANDE, H., AND GARCIA-MOLINA, H. Transience of peers & streaming media. In *Proc. of HotNets-I* (October 2002).

[8] BIRRER, S., AND BUSTAMANTE, F. E. Resilient overlay multicast from ground up. Tech. Report NWU-CS-03-22, Northwestern U., July 2003.

[9] BLAHUT, R. E. *Theory and Practice of Error Control Codes*. Addison Wesley, 1994.

[10] BUSTAMANTE, F. E., AND QIAO, Y. Friendships that last: Peer lifespan and its role in P2P protocols. In *Proc. of IWCW* (October 2003).

[11] CASTRO, M., DRUSCHEL, P., KERMARREC, A.-M., NANDI, A., ROWSTRON, A., AND SINGH, A. Splitstream: High-bandwidth multicast in cooperative environments. In *Proc. of the 19th ACM SOSP* (October 2003).

[12] CASTRO, M., ROWSTRON, A., KERMARREC, A.-M., AND DRUSCHEL, P. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication 20*, 8 (2002).

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

[14] CHU, Y.-H., GANJAM, A., NG, T. S. E., RAO, S. G., SRIPANIDKULCHAI, K., ZHAN, J., AND ZHANG, H. Early experience with an Internet broadcast system based on overlay multicast. In *Proc. of USENIX ATC* (June 2004).

[15] CHU, Y.-H., RAO, S. G., SESHAN, S., AND ZHANG, H. Enabling conferencing applications on the Internet using an overlay multicast architecture. In *Proc. of ACM SIGCOMM* (August 2001).

[16] CHU, Y.-H., RAO, S. G., AND ZHANG, H. A case for end system multicast. In *Proc. of ACM SIGMETRICS* (June 2000).

[17] DEERING, S. E. Multicast routing in internetworks and extended LANs. In *Proc. of ACM SIGCOMM* (August 1988).

[18] DIOT, C., LEVINE, B. N., LYLES, B., KASSEM, H., AND BALENSIEFEN, D. Deployment issues for the ip multicast service and architecture. *IEEE Network 14*, 1 (January/February 2000).

[19] FLOYD, S., JACOBSON, V., LIU, C.-G., MCCANNE, S., AND ZHANG, L. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking 5*, 6 (December 1997).

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

[21] GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, January 1989.

[22] GUMMADI, K. P., DUNN, R. J., SAROIU, S., GRIBBLE, S. D., LEVY, H. M., AND ZAHORJAN, J. Measurement, modeling and analysis of a peer-to-peer file-sharing workload. In *Proc. of ACM SOSP* (December 2003).

[23] JANNOTTI, J., GIFFORD, D. K., JOHNSON, K. L., KAASHOEK, M. F., AND O'TOOLE JR, J. W. Overcast: Reliable multicasting with and overlay network. In *Proc. of the 4th USENIX OSDI* (October 2000).

[24] JIN, C., CHEN, Q., AND JAMIN, S. Inet: Internet topology generator. Technical Report CSE-TR-433-00, U. of Michigan, Ann Arbor, MI, 2000.

[25] LEVINE, B. N., AND GARCIA-LUNA-ACEVES, J. A comparison of reliable multicast protocols. *Multimedia Systems Journal 6*, 5 (August 1998).

[26] LEVINE, B. N., LAVO, D. B., AND GARCIA-LUNA-ACEVES, J. J. The case for reliable concurrent multicasting using shared ack trees. In *ACM Multimedia* (November 1996).

[27] MAHAJAN, R., CASTRO, M., AND ROWSTRON, A. Controlling the cost of reliability in peer-to-peer overlays. In *Proc. of IPTPS* (February 2003).

[28] MILLS, D. L. Improving algorithms for synchronizing computer network clocks. In *Proc. of ACM SIGCOMM* (August 1994).

[29] PADMANABHAN, V. N., WANG, H. J., AND CHOU, P. A. Resilient peer-to-peer streaming. In *Proc. of IEEE ICNP* (2003).

[30] PAPADOPOULOS, C., PARULKAR, G. M., AND VARGHESE, G. An error control scheme for large-scale multicast applications. In *Proc. of IEEE INFOCOM* (March 1998).

[31] PAUL, S., SABNANI, K. K., LIN, J. C.-H., AND BHATTACHARYYA, S. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communication 15*, 3 (April 1997).

[32] PENDARAKIS, D., SHI, S., VERMA, D., AND WALDVOGEL, M. ALMI: An application level multicast infrastructure. In *Proc. of USENIX USITS* (March 2001).

[33] RATNASAMY, S., HANDLEY, M., KARP, R., AND SHENKER, S. Application-level multicast using content-addressable networks. In *Proc. of NGC* (November 2001).

[34] RHEA, S., GEELS, D., ROSCOE, T., AND KUBIATOWICZ, J. Handling churn in a DHT. In *Proc. of USENIX ATC* (December 2004).

[35] TOWSLEY, D., KUROSE, J. F., AND PINGALI, S. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. *IEEE Journal on Selected Areas in Communication 15*, 3 (April 1997).

[36] TRAN, D. A., HUA, K. A., AND DO, T. ZIGZAG: An efficient peer-to-peer scheme for media streaming. In *Proc. of IEEE INFOCOM* (April 2003).

[37] WANG, Z., AND CROWCROFT, J. Bandwidth-delay based routing algorithms. In *Proc. of IEEE GlobeCom* (November 1995).

[38] XU, J., KALBARCZYK, Z., AND IYER, R. K. Networked Windows NT system field failure data analysis. In *Proc. of PRDC* (December 1999).

[39] XU, X., MYERS, A., ZHANG, H., AND YAVATKAR, R. Resilient multicast support for continuous-media applications. In *Proc. of NOSSDAV* (May 1997).

[40] XU, X. R., MYERS, A. C., ZHANG, H., AND YAVATKAR, R. Resilient multicast support for continuous-media applications. In *Proc. of NOSSDAV* (May 1997).

[41] YAVATKAR, R., GRIFFOEN, J., AND SUDAN, M. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia* (November 1995).

[42] YI CUI, B. L., AND NAHRSTEDT, K. oStream: Asynchronous streaming multicast in application-layer overlay networks. *IEEE Journal on Selected Areas in Communication 22*, 1 (January 2004).

[43] ZEGURA, E. W., CALVERT, K., AND BHATTACHARJEE, S. How to model an internetwork. In *Proc. of IEEE INFOCOM* (March 1996).

[44] ZHUANG, S. Q., ZHAO, B. Y., JOSEPH, A. D., KATZ, R. H., AND KUBIATOWICZ, J. D. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. of NOSSDAV* (June 2001).