Resilient Peer-to-Peer Multicast from the Ground Up

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

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, promising to make possible large scale multi-party applications such as audio and video conference, research collaboration and content distribution.

A number of research projects have recently proposed an end-system approach to multicast [11, 3, 24, 10, 21, 18, 28], partially in response to the deployment issues of IP Multicast [12, 13]. In this middleware [1] or application-layer approach, peers are organized as an overlay topology for data delivery, with each connection in the overlay mapped to a unicast path between two peers in the underlying Internet. All multicast related functionality is implemented at the peers instead of at routers, and the goal of the multicast protocol is to construct and maintain an efficient overlay for data transmission.

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 [5]. 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. Measurement studies of widely used peer-to-peer (P2P) systems have reported *median session times* [1] ranging from an hour to a minute [8, 15, 22]. **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 techniques: (1) *co-leaders* and, (2) *triggered negative acknowledgments (NACKs)*, Nemo's design emphasizes conceptual simplicity and minimum dependencies [2], thus achieving, in a cost-effective manner, performance characteristics resilient to the natural instability of its target environment. Simulation-based and wide-area experimentations show that Nemo can achieve high delivery ratios (up to 99.98%) and low end-to-end latency similar to those of comparable protocols, while significantly reducing the cost in terms of duplicate packets (reductions $> 85\%$) and control related traffic, making the proposed algorithm a more scalable solution to the problem.

The remainder of this paper describes our approach in more details (Section 2), and present early experimental results in Section 3. We briefly discuss related work in Section 4 and conclude in Section 5.

2. Nemo's Approach

Nemo follows the *implicit approach* [3, 10, 21, 28] 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. We here provide a summarized description of Nemo. For complete details, we direct the reader to the associated technical report [6].

The set of communicating peers are organized into clusters based on network proximity, [2] where every peer is 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 the *degree*. Each of these clusters selects a *leader* [3] that becomes a member of the immediate superior layer. In part to avoid the dependency on a single node, every cluster leader recruits a number of co-leaders 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. Hence peers can lead more than one cluster in successive layers of this logical hierarchy. [4]

Co-leaders improve the resilience of the multicast group by avoiding dependencies on single nodes and providing al-

[1] *Session time* is defined as the time between when a peer joins and leaves the network.

[2] Other factors such as bandwidth [25, 11] and expected peer lifetime [8] could be easily incorporated.

[3] The leader is the peer in the center of the cluster, in terms of end-to-end latency.

[4] This is common to both Nemo and Nice [3] as well as Zigzag [24]; the degree bounds have been chosen to help reduce oscillation in clusters.

NCA'04, August 30 - September 1, 2004, Cambridge, MA, USA.

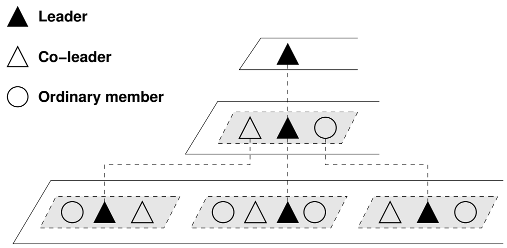


Figure 1: Nemo's logical organization. The shape illustrates only the role of a peer within a cluster: a leader of a cluster at a given layer can act as leader, co-leader, or an ordinary member at the next higher layer.

ternative 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.

A new peer joins the multicast group by querying a well-known special end-system, 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.

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 can choose any of the crew members in the cluster's leader group as destination.

Figure 2 illustrates the data forwarding algorithm using the logical topology from Figure 1. Each row corresponds to one time step. At time t_0 a publisher forwards the packet to its cluster leader, which in turn, sends it to all cluster members and the leader of the next higher layer (t_1). At time t_2, this leader forwards the packet to all its cluster members, i.e. the members of its lowest layer and the members of the second lowest layer. In the last step, the leader of the cluster on the left forwards the packet to its members. While we have employed leaders for this example, Nemo uses co-leaders in similar manner for forwarding.

To illustrate Nemo's resilience to peer failures, Figure 3 shows an example of the forwarding algorithm in action. 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 leaders can still forward their share of messages through the tree. Like other protocols aiming at high resilience [20, 4], Nemo relies on sequence numbers and triggered NACKs to detect lost packets.

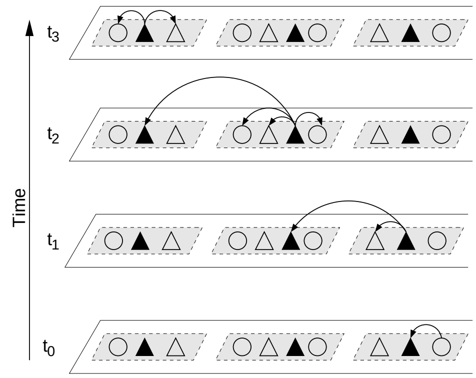


Figure 2: Basic data forwarding in Nemo. One time step per row.

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 a given time period.

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 [11], Nice [3] and Nice-PRM [4] – both in terms of application performance and protocol overhead. Application performance is captured by delivery ratio and end-to-end latency, while overhead is evaluated in terms of number of duplicate packets.

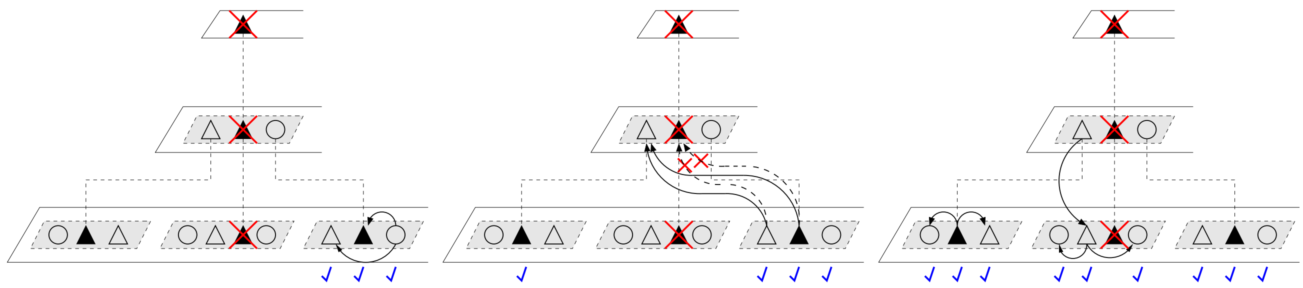
For each of the three alternative protocols, the values for the available parameters were obtained from the corresponding literature [11, 3, 4].

We used two different failure rates. The high failure rate employed a mean time to failure (MTTF) of 5 minutes, and a mean time to repair (MTTR) of 2 minutes. The low failure rate used a MTTF of 60 minutes and a MTTR of 10 minutes. For details on the protocols implementation and on the experimental setup, we direct the reader to the associated technical report [6].

All experiments were run with a payload of 100 bytes. We opted for this relatively small packet size to avoid saturation effects in PlanetLab. For simulations, we assume infinite bandwidth per link and only model link delay, thus packet size is secondary. We employ a buffer size of 32 packets and a rate of 10 packets per second. This corresponds to a 3.2-second buffer, which is a realistic scenario for applications such as multimedia streaming.

3.1. Simulation Results

For all simulation results, each data point is the mean of 25 independent runs.



(a) Publisher forwards to crew.

(b) Crew forwards to next higher layer.

(c) Co-leader forwards to all clusters.

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.

Figure 4 illustrates the delivery ratio during the group-membership-change phase. The first graph shows the delivery ratio of Narada, followed by Nice, Nice-PRM(3,0.01) and Nemo. As the figure illustrates, Nemo has a higher delivery ratio than the alternative protocols. The alternate data paths in Nemo explain these improvements, as they enable the early detection and retransmission of lost packets within the

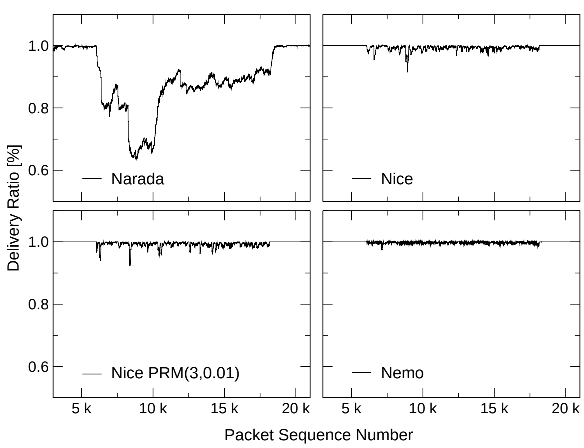


Figure 4: Delivery ratio (512 end hosts, high failure rate).

The summarized results for the two failure rates (high and low) are presented in Table 1 and 2. Nice-PRM(3,0.02), Nice-PRM(3,0.03) and Nemo achieve comparably high delivery ratio, significantly better than Nice and Narada.

The cost of a resilient multicast protocol can be measured in terms of duplicate packets per sequence number. The second set of columns in Table 1 and Table 2 show this overhead for the compared protocols. In both cases, Nemo's approach results in comparably high delivery ratios with significantly lower cost in terms of duplicates per sequence number.

Protocol	Delivery ratio		Duplicate packets		
	Mean	Std	Mean	Max	Std
Nemo	0.998	0.89E-3	3.16	3.77	0.29
Nice-PRM(3,0.01)	0.993	1.25E-3	12.47	14.43	1.04
Nice-PRM(3,0.02)	0.994	1.23E-3	18.20	19.75	0.77
Nice-PRM(3,0.03)	0.994	1.01E-3	24.22	28.14	1.82
Nice	0.992	1.95E-3	7.10	8.32	0.71
Narada	0.852	60.3E-3	0.00	0.00	0.00

Table 1: High-Failure Rate (512 end hosts).

Protocol	Delivery ratio		Duplicate packets		
	Mean	Std	Mean	Max	Std
Nemo	1.000	0.12E-3	0.34	0.59	0.09
Nice-PRM(3,0.01)	0.999	0.56E-3	6.42	6.98	0.38
Nice-PRM(3,0.02)	0.999	0.36E-3	12.00	13.43	0.66
Nice-PRM(3,0.03)	0.999	0.27E-3	16.74	18.42	0.65
Nice	0.999	0.52E-3	1.29	1.92	0.40
Narada	0.950	38.3E-3	0.00	0.00	0.00

Table 2: Low Failure Rate (512 end hosts).

Under both failure rates, 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 well with its delivery ratio.

Figure 5 shows the cumulative distribution function (CDF) of latency for all received packets with a 32-packet buffer. Nemo's co-leaders improve resilience by providing alternative paths for packet forwarding. These alternative paths, with delays higher than or equal to the optimal and only choice in Nice, could introduce some latency penalties for packet delivery. On the other hand, these same paths ensure the delivery of messages otherwise lost. Still, as Figure 5 shows, Nemo suffers no significant additional delays for packets delivered without retransmission (left side of the plot).

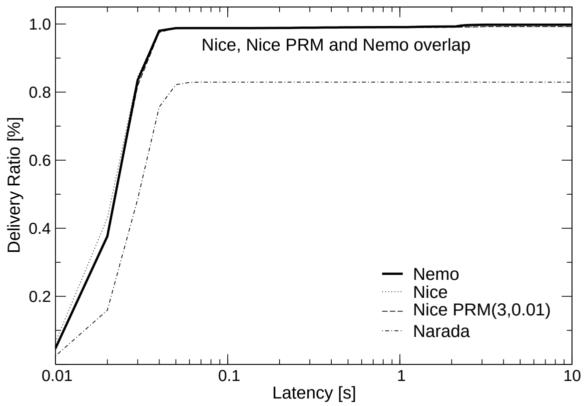


Figure 5: Latency CDF (512 end hosts, high failure rate).

3.2. Wide-Area Results

The results presented here are based on twenty-five runs, where each run is a set of one experiment per protocol, done at different times of the day. We present representative graphs for Nemo, Nice and Nice-PRM(3,0.02). For Nice-PRM we opted for a 2% forwarding probability since

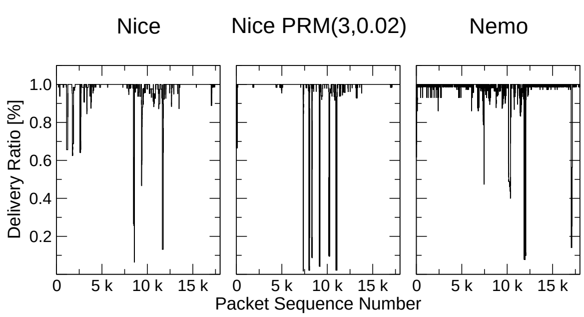


Figure 6: Delivery ratio (PlanetLab, ∼72 end hosts, high failure rate).

Figure 6 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.

Protocol	Delivery ratio		Duplicate packets		
	Mean	Std	Mean	Max	Std
Nemo	0.979	0.010	1.27	2.53	0.56
Nice-PRM(3,0.02)	0.953	0.024	2.02	3.00	0.57
Nice	0.939	0.032	1.06	1.83	0.47

Table 3: Wide-Area Results with High Failure Rate (PlanetLab, ∼72 end hosts).

In Table 3 we summarize the runs of the wide area experiment on PlanetLab. Nemo has a substantially higher delivery ratio than Nice-PRM, while incurring less duplicate packets.

The latency distribution not shown here confirm the data gathered through simulation, although the experienced latencies are slightly higher in the wide area experiment (due in part to user-level processing time). Nemo outperforms Nice on the latency of packets requiring retransmission, as Nemo's alternate data paths translate into earlier packet-loss detection and faster recovery.

4. Related Work

Banerjee et al. [3] 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 middleware 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.

A large number of research projects have addressed reliable and resilient multicast at the network layer [20, 26, 27, 19, 17, 14]. A comparative survey of these protocols is given in [16, 23]. Like many of them, Nemo relies on reactive techniques to recover from packet losses. STORM [26] 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 neighbors who have indicated [5] to cache them locally.

In the context of overlay multicast, a number of protocols has been proposed to improve resilience [4, 9, 24, 18]. ZigZag [24] is a single source P2P streaming protocol that achieves resilience by separating the control and data delivery trees at every level. That is, at each level one peer is made responsible for the organization of the sub-tree and a second one for 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 [4] 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 [9] and CoopNet [18] 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 [7].

[5] Peers distribute the local cache state with every data packet they send.

5. Conclusions

We have presented Nemo, a new overlay multicast protocol designed for high resiliency from the ground up. Through the introduction of co-leaders to minimize dependencies and the use of triggered NACKs to detect lost packets, Nemo is able to achieve high delivery ratios under high stress at a lower cost in terms of duplicate messages than alternative protocols and without penalties in terms of additional delays. We have demonstrated the effectiveness of this approach through a comparative study using simulation and wide-area experimentation, under different stress scenarios.

Acknowledgments

We would like to thank K. Schwan and P. Dinda, who kindly loaned us their equipment for some of our experiments. We are also grateful to J. Casler, Y. Qiao, J. Skicewicz, A. Sundararaj and D. Lu for their helpful comments on early drafts of this paper.

References

[1] B. Aiken, J. Strassner, B. Carpenter, I. Foster, C. Lynch, J. Mambretti, R. Moor, and B. Teitelbaum. Network policy and services: A report of a workshop on middleware. Request for Comments 2768, IETF, February 2000.

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

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

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

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

[6] S. Birrer and F. E. Bustamante. Nemo - resilient peer-to-peer multicast without the cost. Tech. Report NWU-CS-04-36, Northwestern U., April 2004.

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

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

[9] 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.

[10] M. Castro, A. Rowstron, A.-M. Kermarrec, and P. Druschel. SCRIBE: A large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication*, 20(8), 2002.

[11] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communication*, 20(8), October 2002.

[12] S. E. Deering. Multicast routing in internetworks and extended LANs. In *Proc. of ACM SIGCOMM*, August 1988.

[13] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, 14(1), January/February 2000.

[14] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6), December 1997.

[15] 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.

[16] B. N. Levine and J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *Multimedia Systems Journal*, 6(5), August 1998.

[17] B. N. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves. The case for reliable concurrent multicasting using shared ack trees. In *ACM Multimedia*, November 1996.

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

[19] C. Papadopoulos, G. M. Parulkar, and G. Varghese. An error control scheme for large-scale multicast applications. In *Proc. of IEEE INFOCOM*, March 1998.

[20] S. Paul, K. K. Sabnani, J. C.-H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communication*, 15(3), April 1997.

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

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

[23] 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*, 15(3), April 1997.

[24] 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.

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

[26] X. R. Xu, A. C. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proc. of NOSSDAV*, May 1997.

[27] R. Yavatkar, J. Griffoen, and M. Sudan. A reliable dissemination protocol for interactive collaborative applications. In *ACM Multimedia*, November 1995.

[28] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proc. of NOSSDAV*, June 2001.