Hybrid Dissemination: Adding Determinism to Probabilistic Multicasting in Large-Scale P2P Systems

Spyros Voulgaris[1] and Maarten van Steen[2]

[1] Dept. of Computer Science, ETH Zurich, Switzerland,
spyros@inf.ethz.ch
[2] Dept. of Computer Science, Vrije Universiteit Amsterdam, The Netherlands,
steen@few.vu.nl

Abstract. Epidemic protocols have demonstrated remarkable scalability and robustness in disseminating information on internet-scale, dynamic P2P systems. However, popular instances of such protocols suffer from a number of significant drawbacks, such as increased message overhead in push-based systems, or low dissemination speed in pull-based ones.
In this paper we study push-based epidemic dissemination algorithms, in terms of hit ratio, communication overhead, dissemination speed, and resilience to failures and node churn. We devise a hybrid push-based dissemination algorithm, combining probabilistic with deterministic properties, which limits message overhead to an order of magnitude lower than that of the purely probabilistic dissemination model, while retaining strong probabilistic guarantees for complete dissemination of messages. Our extensive experimentation shows that our proposed algorithm outperforms that model both in static and dynamic network scenarios, as well as in the face of large-scale catastrophic failures. Moreover, the proposed algorithm distributes the dissemination load uniformly on all participating nodes.
Keywords: Epidemic/Gossip protocols, Information Dissemination, Peer-to-Peer

1 Introduction

Large-scale information dissemination constitutes fundamental functionality for a multitude of applications, ranging from file-sharing and web-casting to the massive distribution of software, security patches, and world-wide worm alert notifications. The emergence of new types of applications for large-scale decentralized systems drives the need for efficient, reliable, and scalable information dissemination frameworks.

Early attempts for information dissemination focused on network-layer solutions, leading to a number of IP Multicast protocols. These protocols rely on functionality embedded in routers, that enables the dynamic construction of spanning trees that reach all participating nodes, but generally provide no reliability guarantees. A number of solutions have been proposed on top of IP Multicast, such as SRM [6] and RMTP [13], to improve its reliability. Nevertheless, IP Multicast is not widely deployed on the Internet mainly due to extra complexity and state imposing on routers.

Application-layer multicast forms an alternative class of solutions that has emerged in the recent years. The main advantage of these solutions is that they are very generic, and, therefore, they can be directly deployed over today's network infrastructure.

Part of this work is funded by the European IST-FP6-15964 project AEOLUS (Algorithmic Principles for Building Efficient Overlay Computers).

There exist application-layer multicast protocols that provide reliability guarantees [8]. However, many of them do not scale well to a large number of nodes [17].

A class of application-layer multicast has recently emerged [3, 2, 21], based on the structure of DHTs such as Chord, Pastry, and Tapestry. What is common in these DHTs is that, in their respective overlays, each node is the root of a tree spanning the whole network. These spanning trees are used for message dissemination. Although systems of this class are nearly optimal with respect to message overhead, a single failure along a spanning tree can result in a whole branch missing a message. Failures are disregarded as a whole in [3], where the assumption of reliable communication is made. Scribe [2] provides by default best-effort delivery. Reliability is improved to some extent by imposing TCP connections among nodes, a rather heavy assumption for dynamic, large-scale P2P networks. Finally, Bayeux [21], a system mainly targeted at data streaming, improves on reliability by redundantly disseminating messages across different paths of a spanning tree. However, its design is exposed to scalability problems, as each request to join a group is routed to a single node managing that group.

Gossip-based protocols, such as Bimodal Multicast (*pbcast*) [1] and Directional Gossip [14] form an alternative to broadcasting approaches without sufficient redundancy. Each node forwards a message to a small random subset of the network, and so on. These protocols generally provide only *probabilistic* guarantees for message delivery. However, they are attractive because they are easy to deploy and resilient to node and link failures, due to redundant message deliveries. On the other hand, scalability can suffer if nodes are required to maintain full knowledge of the network, notably when node churn is at stake. Optimizations have been suggested in [1] to overcome such scalability issues.

Other gossiping protocols, such as *lpbcast* [4, 5] and [12, 7] provision for membership management too. In particular, [7] describes a hybrid dissemination system, that multicasts messages using a tree-based hierarchical structure, and locally switches to gossiping when a large number of failures is detected. These protocols drop the assumption of full knowledge of the network. Each node maintains a small view of the network, consisting of a few links to neighbors, which are used for dissemination. This makes them highly scalable. However, due to their probabilistic nature, a message may fail to reach the whole network even in a fail-free environment. To alleviate this, highly redundant message forwarding is employed.

Excessive redundancy of push-based approaches can be reduced while retaining a high hit ratio, by employing pull-based epidemic techniques: nodes periodically poll other nodes to pull messages they may have missed. However, the periodic nature of pull-based gossiping results in relatively long latency of message dissemination, significantly longer than reactive push-based approaches. We will not consider pull-based techniques in this paper.

Contributions

The contributions of this paper are three-fold. First, we study the algorithm proposed in [12] (which we call RANDCAST), we observe and quantify the excessive message overhead it imposes on the network, and explain why the class of flat, *probabilistic* *dissemination algorithms* requires high levels of redundancy to disseminate messages to the whole node population.

Second, we reason that imposing some level of *determinism* on probabilistic dissemination algorithms can substantially reduce the dependence on message redundancy, introducing the class of *hybrid (probabilistic/deterministic) dissemination algorithms*. Protocols of this class achieve deterministic dissemination to all nodes in fail-free environments. When failures occur, their reliability degrades gracefully with the number of failures.

Third, we propose RINGCAST, a novel hybrid dissemination algorithm, which achieves complete dissemination of messages (hit ratio 100%) with an order of magnitude lower message overhead compared to RANDCAST. Our extensive experimentation and side by side comparison of the two protocols, show that RINGCAST outperforms RANDCAST in terms of hit ratio, message redundancy, tolerance to node churn, and resilience to (even large-scale) node failures. Moreover, both algorithms distribute the dissemination load uniformly on all participating nodes.

2 Evaluating a Dissemination System

A number of issues are of concern when evaluating or comparing information dissemination systems. It is essential for the rest of this paper to list the metrics used to evaluate the effectiveness and usefulness of a dissemination system.

Hit ratio This is defined as the ratio of nodes that receive a message over the total node population. It rates the dissemination reliability. Ideally, a reliable dissemination system should always achieve a hit ratio of 100%. In our evaluation (Section 7) we present graphs of the complementary *miss ratio* metric, defined as: $MissRatio = 1 - HitRatio$.

Resilience to failures and churn For a dissemination system to be meaningful in a real-world dynamic network, it should operate reasonably well in the presence of node or link failures, and node churn. The operation under such conditions is evaluated by means of the hit ratio, described above.

Dissemination speed The time required for the dissemination of a particular message to complete. The faster a message is disseminated the better. Dissemination speed depends on two principal factors. First, the delay in forwarding messages (processing delay on nodes plus network latency). Second, the number of hops a message takes to reach the last node. In our evaluation we focus on the latter factor.

Message overhead The overall number of times a message is forwarded during its dissemination. For a message to reach N recipients, it should be forwarded a minimum of N times. In practice, however, messages are forwarded a number of redundant additional times, to sustain churn and failures. Message overhead rates a dissemination system with respect to preserving or wasting network resources.

Load distribution The distribution of load over nodes, in terms of messages received and messages forwarded. Ideally, load should be evenly distributed among participating nodes.

In this paper we are interested in reliable dissemination of messages originating at *any* node to *all* participating nodes. We do not focus on optimizing the dissemination

```
when node $P$ generates message $m$,
        or receives $m$ from node $Q$ do
    if $m$ not already seen then
        targets $\leftarrow$ selectGossipTargets($Q$)
        foreach $T \in$ targets do send($T$, $m$)
    endif
end
```

(a)

```
function selectGossipTargets($Q$)
    targets $\leftarrow$ view-$\{Q\}$
    return targets
end
```

(b)

Fig. 1. (a) The generic dissemination algorithm. (b) Gossip target selection for deterministic dissemination (flooding).

of messages with respect to any proximity metric or by building a spanning tree. Also, we do not consider positive or negative acknowledgements, or requests for retransmission of lost messages. Instead, we introduce redundancy in message dissemination and examine its relation to the level of reliability achieved. We investigate the power of epidemics at disseminating messages to all nodes, with a high probability.

3 Deterministic Dissemination

Consider a system consisting of N nodes, and a set of directed links among them. A *message* can originate at any of the participating nodes, and aims at reaching the whole network. A node that generates a new message or receives a message for the first time, forwards it across *all* its outgoing links. If a node receives a message for the second time, it simply ignores it. As an optimization, a message is never forwarded back to the node it was just received from. This basic algorithm is often referred to as *flooding*. Figure 1(a) shows the pseudocode for the dissemination algorithm.

The distinguishing characteristic of flooding is that one can deterministically control dissemination by imposing the appropriate overlay on the nodes. The underlying requirement to guarantee complete dissemination starting from any participating node, is to form a *strongly connected directed graph*[3] including *all* nodes. A multitude of overlays have been proposed for information dissemination by means of flooding, each one demonstrating a different behavior with respect to the metrics listed in the previous section.

Spanning trees or simply *trees* were among the first types of overlays proposed for flooding. Their strong point is that they are optimal with respect to the number of links maintained and, consequently, to the message overhead associated with dissemination. Indeed, in a network consisting of N nodes, the complete dissemination of a message over a tree involves exactly $N - 1$ point-to-point communications. Their main disadvantage, though, is that a single failure of any link or any non-leaf node disconnects the tree prohibiting messages from reaching all nodes. Also, maintaining a valid tree structure, ensuring the graph is connected and yet acyclic, is not a trivial task in the presence of failures. For these reasons, trees are not suitable for dynamic environments where failures can happen.

A special type of tree-based overlays for flooding is the *server-based* class (*star graphs*), where all nodes are connected by bidirectional links to a single node acting as a relay server. In these overlays all but the server node are leaf nodes, therefore their

[3] a directed graph in which there is a directed path between any ordered pair of nodes

failure has no effect on the remaining nodes, but the server becomes a single point of failure. In addition, such overlays demonstrate the worst possible load distribution, the server node being linearly loaded by the number of nodes and number of messages being disseminated, rendering it a non-scalable solution.

On the other end of the spectrum lie *cliques* (*complete graphs*). In such a setting, every node has a complete view of the network. A node broadcasts a message by sending it to every other node in the network. This provides maximum reliability, at the cost of high maintenance costs. Although messages always reach all nodes irrespectively of how many nodes have failed, maintaining this type of overlay is impractical. Maintaining a fully connected graph is expensive in networks larger than a few dozen nodes, notably when the membership changes continuously.

A class of flooding overlays deserving more attention is the one based on *Harary graphs*, introduced by Harary in [9], further studied by Jenkins and Demers [11], and applied by Lin et al. [15] in flooding. A Harary graph of connectivity t is a minimal link graph that is guaranteed to remain connected when up to $t - 1$ nodes or links fail. Its minimum cut, therefore, consists of t links. Moreover, in a Harary graph links are evenly distributed across nodes, each node having either t or $t + 1$ bidirectional links. An example Harary graph of connectivity two is a bidirectional ring, that we will use later in Section 5.1. Such overlays are very appealing for information dissemination in the presence of failures, as they are guaranteed to sustain up to a certain number of failures while imposing the minimum message overhead (for the corresponding reliability guarantees), and this overhead is evenly balanced across all nodes. The maintenance of such graphs, notably of higher connectivity t, can be a complicated and expensive task for large-scale, dynamically changing networks.

4 Probabilistic Dissemination

Acquiring reliability by imposing systematic structure on overlays is infeasible in dynamic networks of massive scale. In this section we take a look at an appealing alternative, *probabilistic dissemination* algorithms, which trade-in deterministic reliability guarantees in return of overlay construction and maintenance simplicity.

In these algorithms, dissemination is not guaranteed by means of a strategic topology, but by increased redundancy in message forwarding. The basic idea is that a node receiving a message forwards it to a number of *random* other nodes. It turns out that if that number is sufficiently high, messages reach all nodes with a high probability [12]. The choice of random nodes to forward messages to can be easily handled by a PEER SAMPLING SERVICE, as described in [10]. The main advantage of probabilistic dissemination algorithms is that they are very simple to implement and inherently tolerant to dynamic environments, at the cost of increased message overhead.

4.1 The RANDCAST Dissemination Algorithm

We consider a system consisting of N nodes. Each node runs the PEER SAMPLING SERVICE, providing it with a small, random, partial view of the network. A *message* can originate at any of the participating nodes, and aims at reaching the whole network. A node that generates a new message or receives a message for the first time, forwards it to (up to) F nodes, called the node's *gossip targets*, chosen randomly from its PEER

SAMPLING SERVICE view. F is a system-wide parameter, called the *fanout*. A message is never forwarded back to the node it was just received from. Figure 2 shows the pseudocode for the selection of gossip targets in the RANDCAST dissemination algorithm.

```
function selectGossipTargets($Q$)
    targets ← $F$ random nodes from view-$\{Q\}$
    return targets
end
```

Fig. 2. Gossip target selection for the RANDCAST dissemination algorithm.

Note that this algorithm is quite efficient at spreading a message to a considerable percentage of the nodes in the network very fast, specifically at exponential speed with base F: A new message progressively reaches F^0 (=1, the message generator), F^1, $F^2, \ldots$ other nodes. Consequently, a message spreads very fast even for small values of $F \geq 2$. As expected, dissemination slows down when the message is forwarded to nodes that have already received it. However, if the selection of nodes to forward a message to is uniformly random, this slowdown is expected to be negligible until the message has reached a substantial percentage of the network.

Despite its strength at spreading messages fast, RANDCAST is not as efficient at achieving *complete dissemination*, that is, to reach every single node in the network. It is by nature a probabilistic algorithm. Even in the absence of failures, it provides no hard guarantees that a message will reach *all* nodes. It is not hard to see why. By forwarding messages at random, a node has no guarantees that at least one of its incoming links will be chosen to forward the disseminated message. To alleviate this, abundant redundance should be introduced by means of a large fanout. However, this is not desirable, because message overhead increases proportionally to the fanout, as we will see in the evaluation in Section 7. The RANDCAST dissemination algorithm has been analyzed and evaluated by Kermarrec et al in [12].

In the following section we introduce a novel class of hybrid dissemination algorithms, combining deterministic and probabilistic dissemination. We also present a particular protocol of this class. We defer the evaluation of both protocols until Section 7, where they are compared side by side.

5 Hybrid Dissemination

As we discussed above, although probabilistic protocols are good at spreading messages fast even for small values of F, a large value of F is mandated to reach every single node in the network. This inefficiency can be tackled by introducing some *determinism* in the selection of gossip targets, ensuring any possible dissemination graph is connected and includes all nodes.

Hybrid dissemination protocols aim at combining probabilistic and deterministic behavior. To that end, they establish two types of links among nodes. Random links (*r-links*) contribute to their probabilistic behavior, and deterministic links (*d-links*) bring in determinism. R-links are simply links randomly selected, just like in purely probabilistic dissemination protocols. When presented with a message, a node forwards it across a few r-links. Consequently, messages initially spread to a large portion of the network at close to exponential speed.

However, a message being disseminated should reach every single node in the network. That is, it should be forwarded across at least one incoming link of each node. The basic idea is to establish a set of d-links, and have nodes deterministically forward messages across *all* their outgoing d-links, in addition to a few of their outgoing r-links. If the set of d-links forms an overlay compliant to the deterministic dissemination protocols' requirement, that is, it forms a strongly connected directed graph including all nodes, complete dissemination of messages is guaranteed. In such a graph, each node's indegree is at least 1. Moreover, if we ensure that the graph defined by the d-links has a minimal cut of t, then complete dissemination is guaranteed even in the presence of up to $t - 1$ faulty nodes.

Hybrid protocols effectively decouple the two fundamental goals in information dissemination. On one hand, spreading a message to a large percentage of the nodes fast, and on the other, reaching every single node. The probabilistic component carries out the bulk of the dissemination task, while the deterministic one takes care of the fine-grained details.

What makes hybrid dissemination protocols attractive, is that the set of d-links does not need to form a particularly sophisticated and hard-to-maintain structure. The sole requirement is that the set of d-links forms a strongly connected directed graph over all nodes. A simple structure satisfying this requirement is a ring. In the following section we explore how it can be used as a basis for a practical hybrid dissemination system.

5.1 The RINGCAST Dissemination Algorithm

We introduce RINGCAST, a novel *hybrid dissemination algorithm* that—even with a very low fanout—guarantees complete dissemination in a failure-free environment. In the presence of failures, its performance degrades gracefully, nevertheless still outperforming RANDCAST. Finally, when confronted with continuous churn, RINGCAST proves again more reliable than RANDCAST, excluding nodes that joined the system very recently (for which it performs worse).

As discussed above, hybrid dissemination algorithms maintain two types of links between nodes, namely r-links and d-links. R-links are random links, obtained by a membership management protocols such as the PEER SAMPLING SERVICE [10]. With respect to d-links, RINGCAST organizes nodes in a *global bidirectional ring* structure. A bidirectional ring constitutes a strongly connected graph, as required by deterministic dissemination protocols. Figure 3 illustrates an example RINGCAST overlay, where nodes form a bidirectional ring, and each one has a single outgoing r-link.

Just like in the dissemination protocols discussed earlier, a node that generates a new message or receives a message for the first time, forwards it to (up to) F nodes, where F is the system-wide fanout parameter. However, in the case of RINGCAST, a node always forwards a message to its two ring neighbors (sending it across its two outgoing d-links), and across $F - 2$ randomly selected r-links. If the message was received through one of the node's ring neighbors, the node forwards it to the other ring neighbor, and across $F - 1$ random r-links. Figure 5 shows the pseudocode for the selection of gossip targets in the RINGCAST dissemination algorithm.

Note that a bidirectional ring is a Harary graph of connectivity two, that is, its minimal cut is two. Consequently, although no single node failure can break the ring in two disjoined partitions prohibiting complete dissemination to the remaining nodes, such a

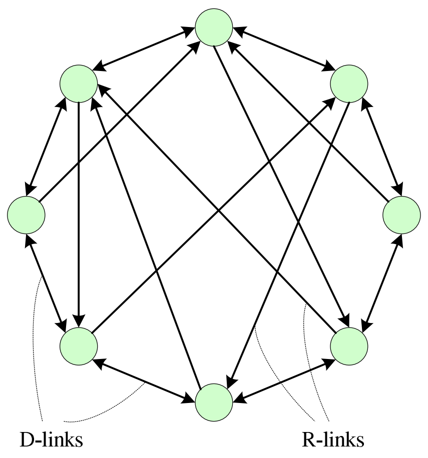


Fig. 3. Example of a RINGCAST overlay. Nodes are organized in a bidirectional ring (by means of the *d-links*), and each one has a number (in this case only one) outgoing random links (*r-links*).

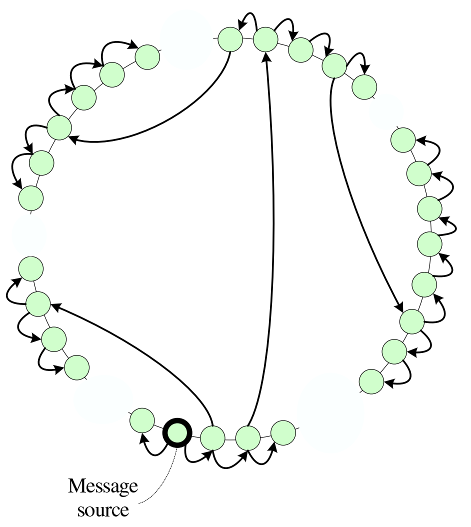


Fig. 4. Example of a message dissemination in a partitioned ring. For clarity, only a few of the followed r-links are shown.

```
function selectGossipTargets($Q$)
    targets ← {}
    if ringNeighbor1 $\neq Q$ then targets ← targets + {ringNeighbor1}
    if ringNeighbor2 $\neq Q$ then targets ← targets + {ringNeighbor2}
    targets ← targets + ($F$−targets.size) random nodes from (view−{$Q$})
    return targets
end
```

Fig. 5. Gossip target selection for the RINGCAST dissemination algorithm.

situation *will* occur if two non-adjacent nodes fail. In most cases, however, this is not a crucial problem for dissemination, as d-links are only one facet of the process. R-links can carry the message to arbitrary nodes, most often bridging the gap between two or more disjoined ring partitions. Effectively, it suffices if any *one* node of an isolated ring partition receives the message, as the message will propagate to the whole partition over the d-links. Figure 4 presents a complete dissemination scenario over a ring split in several partitions. As we will see in the evaluation in Section 7, RINGCAST achieves a high hit ratio (higher comparatively to RANDCAST) even in the presence of many failed nodes.

6 Building the RANDCAST and RINGCAST Overlays

The r-links and d-links are built using epidemic protocols too:

Random links (*R-links*) Several methods may be applied to randomly sample peers in an unstructured peer-to-peer overlay, e.g. by means of the PEER SAMPLING SERVICE [10]. In RINGCAST we use CYCLON [19], an epidemic protocol that is an instance of the PEER SAMPLING SERVICE, and that has shown to produce overlays that strongly resemble random graphs. Omitting certain details, in CYCLON each node maintains a small view of ℓ_{cyc} links to random other nodes. A node periodically gossips with another node, trading *some* of their links with each other. As a result, node views are periodically refreshed by links to random other nodes in the network. At any given moment, the current snapshot of the nodes along with their links resembles a random graph.

Deterministic ring links (*D-links*) Such links are maintained using a proximity-based topology construction epidemic protocol, here we use VICINITY [20]. The basic idea is that nodes maintain short views of the network of length ℓ_{vic}. They periodically gossip to random other nodes, exchanging their views. Upon epidemic view exchanges, a node keeps the ℓ_{vic} links to the closest peers according to a given proximity metric. This way, the neighbor set of each node gradually converges to the closest peers out of the whole node population. Here proximity refers to the distance between—arbitrarily chosen—*sequence IDs*, which determine the organization of nodes in a ring structure. The d-links of a node are the two peers with just higher and just lower sequence ID. Links to a few more peers with gradually higher and lower sequence IDs are not involved in the dissemination protocol, but are useful in maintaining the ring in dynamic conditions.

Note that both these protocols have a periodic nature. Each node initiates an epidemic view exchange (per protocol) once every T time units (nodes have independent, non-synchronized timers). We refer to T as the *cycle* of the protocol. This will be relevant in Section 7.3, where the churn rate is defined relative to the cycle length.

7 Evaluation

We evaluate the two protocols side by side in three scenarios. First, in a static and failure-free network. Second, in a static network right after a catastrophic failure, that is, after the sudden failure of a large number of nodes. Finally, in a dynamic network under continuous node churn. Evaluation was done with respect to the following criteria, as discussed in Section 2:

1. Hit ratio
2. Dissemination speed
3. Message overhead

We do not explicitly address load balancing, because both protocols are by nature distributing the load across all nodes evenly. A node receiving a message forwards it to F others, just like any other node.

Experiments were carried out using the PeerSim simulator [16]. We tested all scenarios by instantiating a network of 10,000 nodes. Each node was running CYCLON and, in the case of RINGCAST, VICINITY too, as described above, with view length 20 for each protocol ($\ell_{cyc} = \ell_{vic} = 20$). The view lengths are not crucial for the behavior of these algorithms ([20]). Nodes were initially supplied with a certain single contact in their CYCLON views, forming a star topology. VICINITY views were initially empty. After letting the network self-organize (for the record we let it run for 100 cycles, which were more than enough), we started disseminating messages from various nodes picked at random.

We assume a very simple dissemination model, that allows us to study the evolution of disseminations in terms of discrete rounds, that we call *hops*. The generation of a message is marked hop 0. At hop 1, the message reaches F neighbors of the origin node. At hop 2, it further reaches the neighbors' neighbors, and so on. This way, we can evaluate the progress of a dissemination by counting the number of messages sent and the number of new nodes notified per hop.

An implicit assumption underlying our dissemination model is that the processing delay and network latency between all pairs of nodes are the same. Although latencies vary in a real wide-area network, our assumption does not have an effect on the macroscopic behavior of dissemination with respect to the hit ratio. Dissemination relies on nodes forwarding the messages they receive. A node that receives a message for the first time, forwards it to the same number of neighbors picked with the same logic, irrespectively of the time this happens. Consider for instance two scenarios of RANDCAST, executing over the same static overlay (assume gossiping is currently stalled), starting from the same origin and each node picking the same gossip targets in both cases. If pair-wise latencies are different in the two scenarios, the order in which nodes are notified may change, but the exact same set of nodes will have been eventually notified. In the case of RINGCAST, the set of nodes notified may change, but the same macroscopic behavior is maintained.

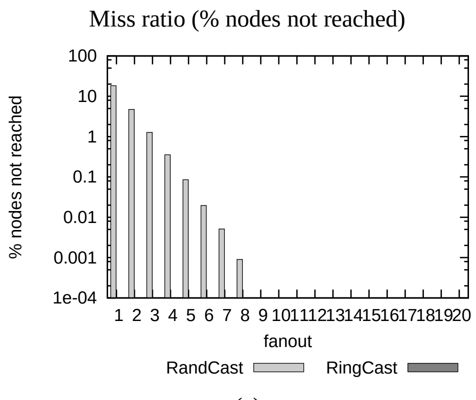
7.1 Evaluation in a Static Failure-free Environment

We first evaluate and compare the two protocols side by side by considering a failure-free static environment.

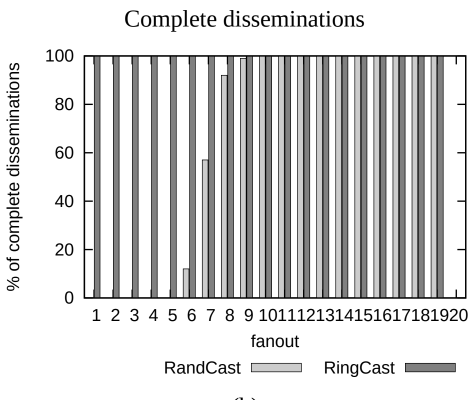
We instantiated a network of 10,000 nodes in PeerSim. Each node was running CYCLON and, in the case of RINGCAST, VICINITY too as described above, with view length 20 for each protocol. Nodes were initially supplied with a given single contact in their CYCLON views, forming a star topology. VICINITY views were initially empty. After letting the network self-organize for 100 cycles, we started posting messages and observing their dissemination.

We ran a number of experiments—not presented here—to investigate the effect of gossiping speed on dissemination. More precisely, we explored the relation between the gossiping period and message forwarding time, that is, the time is takes a node to process a message and forward it to a neighbor. We varied the message forwarding time from zero to several times the gossiping period. We recorded no effect whatsoever on the macroscopic behavior of disseminations. That is, although changing the message forwarding time results in different experiments, with different nodes being reached each time and in a different order, all macroscopic properties, such as the hit ratio, dissemination speed, and message overhead, are preserved. It is not hard to see why. With respect to VICINITY-managed d-links, they are not even altered by gossip exchanges once the optimal sets have been obtained. With respect to CYCLON-managed r-links, these are random links anyway, irrespectively of whether they are being updated fast or are currently fixed. Consequently, forwarding a message along a few of them has an equivalent effect regardless of whether gossiping runs at a high rate or is currently stalled.

Having verified this, we chose to disseminate messages over *fixed* overlays in all experiments presented in this section. This choice was primarily made to limit simulation execution to a reasonable time, considering the large number of experiments we carried out. So, in each experiment, after self-organizing for 100 cycles, the overlay was frozen and only then did disseminations start.



(a)



(b)

Fig. 6. Dissemination effectiveness as a function of the fanout, for a failure-free static network of 10K nodes. (a) Miss ratio averaged over 100 experiments; (b) Percentage of 100 experiments that resulted in complete dissemination.

For each value of F ranging from 1 to 20, we posted 100 messages from various nodes picked at random, resulting in a total of 2000 experiments for each protocol. Since the hit ratio approaches 100% even for small values of F, it is more meaningful to present the miss ratio instead, in logarithmic scale. Figure 6(a) presents the dissemination miss ratio averaged over 100 experiments for each value of F. RANDCAST and RINGCAST are represented by light and dark bars, respectively. The miss ratio for RANDCAST appears to be dropping exponentially as a function of the fanout F. Note that no dark bars appear in this graph, as the miss ratio for RINGCAST is zero for any choice of F. This comes as no surprise, as RINGCAST's operation guarantees complete dissemination in failure-free static networks.

Figure 6(b) shows the percentage of experiments that resulted in a complete dissemination, for each value of F. With respect to RANDCAST, it is interesting to see that the transit from 0% to 100% follows a rather steep curve. For instance, even with a fanout of 5, although the overall hit ratio was above 99.9% (Fig. 6(a)), none of the 100 experiments resulted in a complete dissemination. With a fanout of 7, more than half of the disseminations were complete, while by further increasing the fanout to 11 or higher we get only complete disseminations. As far as RINGCAST is concerned, this graph validates once again that disseminations are always complete, irrespectively of the chosen fanout.

Having seen to what extent messages eventually spread, we now take a closer look at the evolution of dissemination hop by hop. Figure 7 shows the progress of all 100 dissemination for each protocols, for four different fanouts. More specifically, it shows the number of nodes that have not yet been notified, as a function of the hops taken.

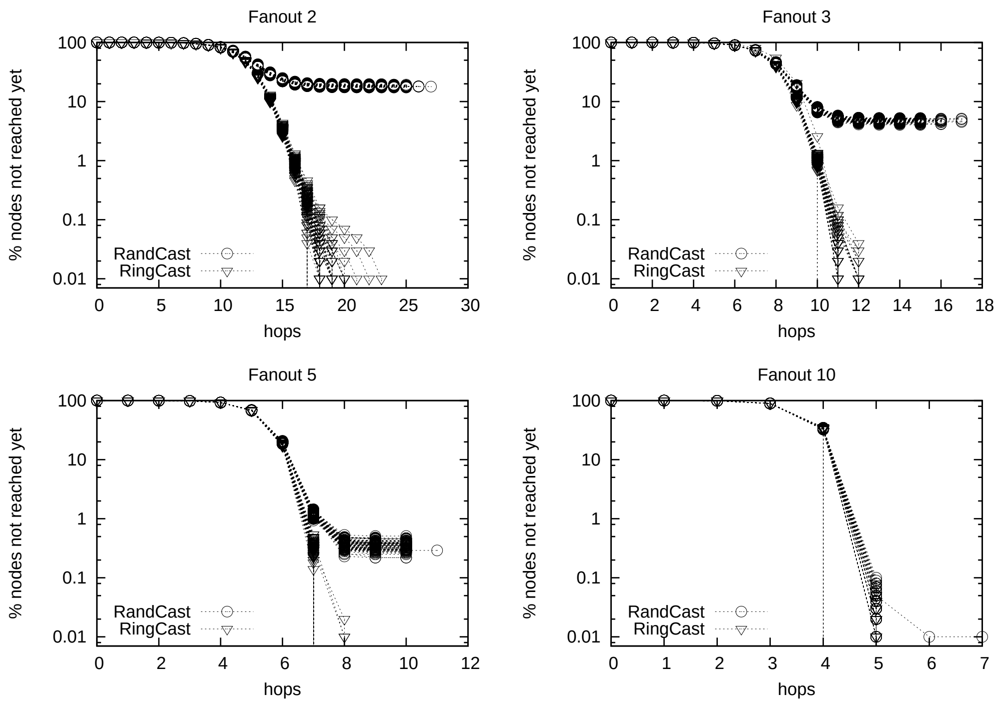
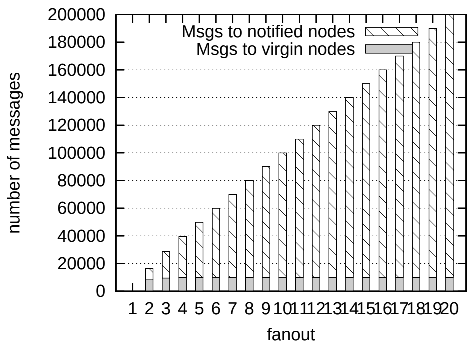


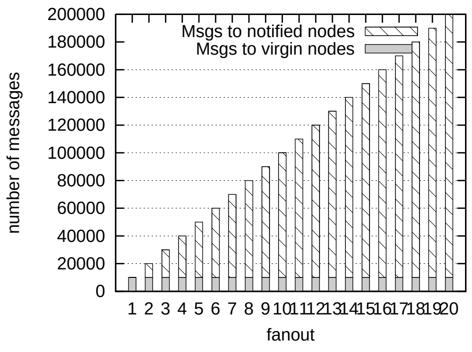
Fig. 7. Dissemination progress in a static failure-free network of 10K nodes. 100 experiments of each protocol are shown.

Four main observations can be made by examining these graphs. First, for a given fanout, all experiments of a protocol demonstrate very small variations in their progress with respect to the hit ratio and dissemination latency. This is important as it shows that by selecting the appropriate fanout value, we can tune a system's dissemination behavior to a good level of accuracy. Second, we notice a clear—expected—influence of the fanout on dissemination latency. The higher the fanout, the shorter a dissemination's duration. Third, we observe that the progress of disseminations for the two protocols is alike for a few initial hops, when the message has not yet reached a significant portion of the network. The protocols differentiate only after a substantial percentage of the nodes (i.e., at least 80%-90%) have been notified. This is a direct effect of the two protocols' operation. By forwarding messages at random, RANDCAST hardly reaches any more non-notified nodes, in an already saturated network. On the contrary, by also forwarding messages along the ring, RINGCAST exhaustively reaches out to every single node. Finally, we see that the higher the fanout the more similarly the two protocols disseminate messages. However, in all cases RINGCAST reaches the last node in fewer hops, demonstrating a lower dissemination latency.

The third metric we are interested in is message overhead. As we already mentioned in Section 4.1, message overhead increases proportionally to the fanout. Indeed, if a node forwards a newly received message to F other nodes and N_{hit} nodes are reached in a dissemination, the total number of messages sent is $F \times N_{hit}$. Figure 8 confirms this assessment. The shaded segments represent the number of messages reaching nodes for the first time (noted as "virgin" nodes). The striped segments represent the number of



RANDCAST



RINGCAST

Fig. 8. Total number of messages sent, divided in messages sent to not-yet-notified and already notified nodes.

redundant messages, that is, messages reaching already notified nodes, and therefore constitute a waste of network resources. As the network consists of $10K$ nodes, for a given fanout F a complete dissemination involves $F \times 10K$ total messages, out of which $10K$ are messages to "virgin" nodes, and the rest $(F - 1) \times 10K$ are redundant. The two graphs are practically identical except for low fanouts, for which RANDCAST disseminations do not reach all nodes. These graphs are illustrative with respect to the reason the fanout should be kept as low as possible.

7.2 Evaluation after Catastrophic Failure

For a system to be usable in a realistic environment, it has to cope with failures. In this section we explore the behavior of the two protocols in the face of catastrophic failures, that is, when a number of nodes suddenly break down.

We set up the experiments like the ones in the previous section, but before starting the disseminations we kill a randomly chosen portion of the nodes. That is to say, for each experiment we simulate a network of 10,000 nodes, let it self-organize for 100 cycles, and stall gossiping. We subsequently remove a randomly chosen set of the nodes and examine dissemination over the remaining ones.

Unlike failure-free static networks where ongoing gossiping has no influence on dissemination after some point (see Section 7.1), in the face of failures gossiping *does* have an effect, namely a positive one. Following a catastrophic failure, gossiping allows the network reorganize itself, removing links to dead nodes and reestablishing valid ring links. In our experiments gossiping was *not* allowed following the catastrophic failure, exploring the ability of a partially damaged overlay to disseminate messages without giving it the chance to self-heal. This was our deliberate choice, aiming at testing a catastrophic failure's worst-case influence on dissemination.

Figure 9 presents the dissemination effectiveness for both protocols after catastrophic failures killing 1%, 2%, 5%, and 10% of the nodes. Similarly to Figure 6 in the previous section, the graphs on the left show the miss ratio, and the ones on the right the percentage of disseminations that reached all nodes, as a function of the fanout F.

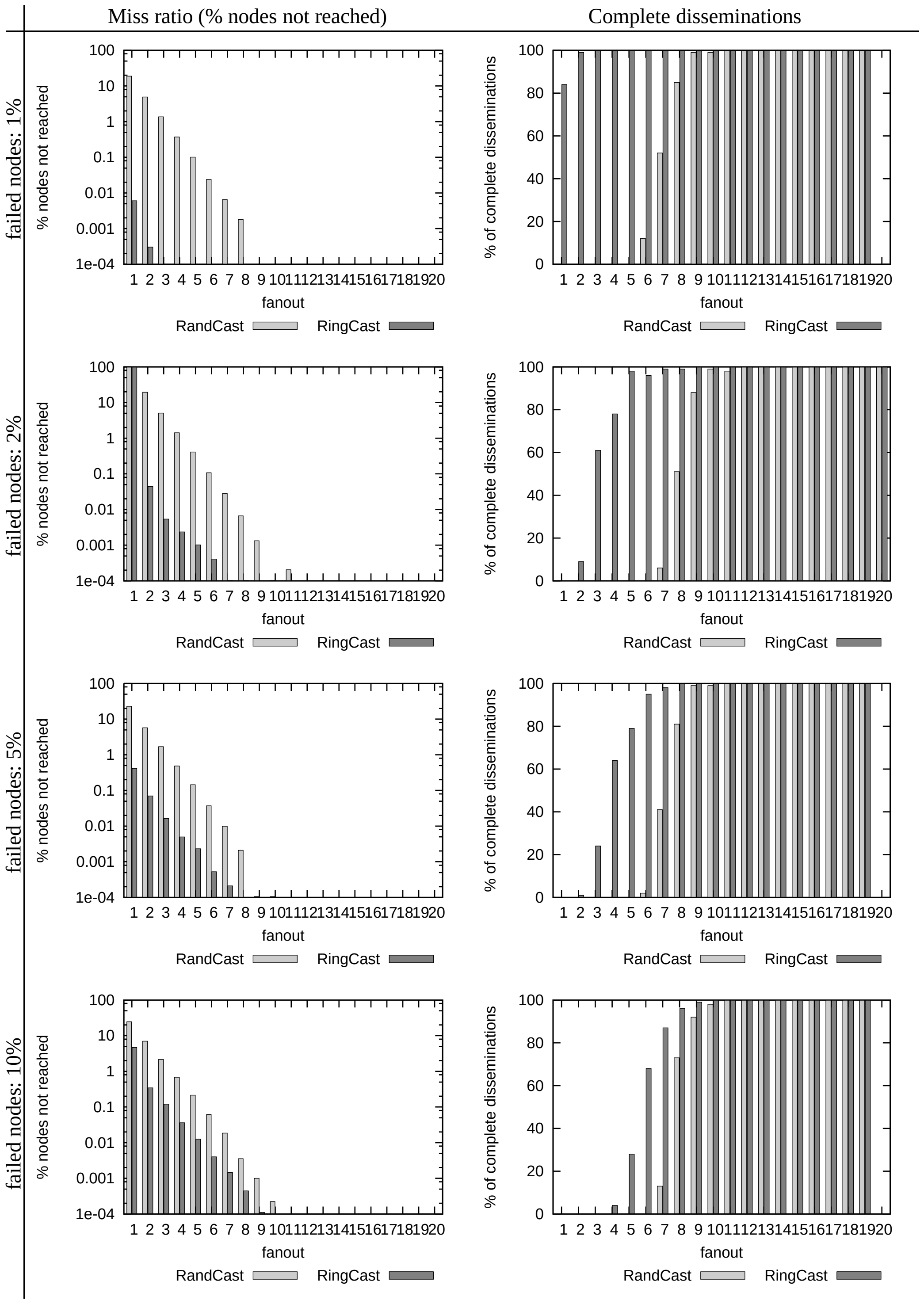


Fig. 9. Dissemination effectiveness as a function of the fanout for static network of 10K nodes, after catastrophic failures of 1%, 2%, 5%, and 10% of the nodes.

One can clearly see that RINGCAST is more effective at disseminating messages in all experiments. A closer look at these graphs shows that as the volume of the catastrophic failure grows larger, the difference between the two protocols' effectiveness decreases. However, even when 10% of the nodes are killed at once, RINGCAST demonstrates an order of magnitude lower miss ratio than RANDCAST. The lower miss ratio of RINGCAST reflects on the significantly higher percentage of complete disseminations for small fanouts.

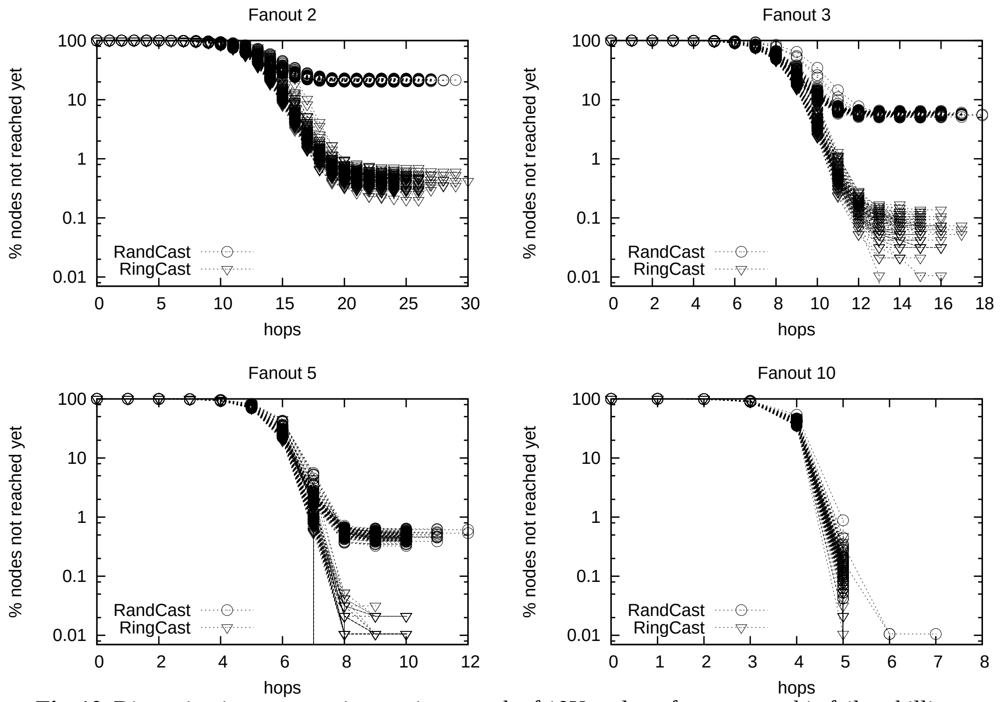


Fig. 10. Dissemination progress in a static network of 10K nodes, after catastrophic failure killing 500 nodes (5%). 100 experiments of each protocol are shown.

Figure 10 shows the evolution of disseminations after a catastrophic failure of 5% of the nodes, in accordance to Figure 7 in the previous section. Once again, the relation between the chosen fanout and dissemination latency is verified. We also see that the evolution of disseminations exhibits small variations for a given configuration, like in the case of a failure-free static network.

7.3 Evaluation under Churn

Apart from catastrophic failures, a system should also be able to deal with node churn, that is, continuous node arrivals and departures. In this section, we examine the behavior of the two protocols under churn.

We evaluate the two protocols against the artificial churn model described here. In each cycle a given percentage (known as the churn rate) of randomly selected nodes are removed, and the same number of new ones join the network. Note that this constitutes a worst case churn scenario, as removed nodes never come back, so dead links never become valid again, and new nodes have to join from scratch. We tested both protocols with a churn rate of 0.2%, which, given a gossiping period of 10 seconds, corresponds to the churn rate observed in the Gnutella traces by Saroiu et al [18].

Unlike experiments on static networks where a small number of cycles sufficed to warm up the respective overlays (Sections 7.1 and 7.2), experiments on dynamic networks required significantly more warm-up cycles. A network of 10,000 nodes was let gossip in the presence of continuous artificial churn, until every node had been removed and reinserted at least once. For all experiments this took several thousand cycles. Then the respective network was frozen, and the resulted overlay was tested with respect to dissemination effectiveness.

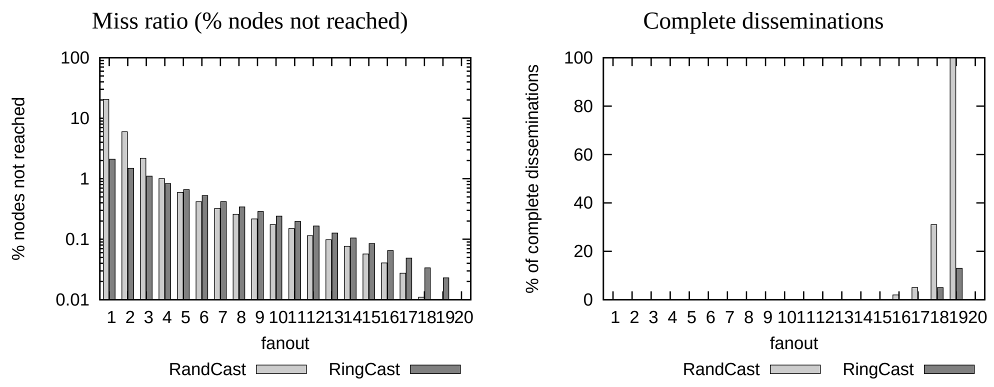


Fig. 11. Dissemination effectiveness as a function of the fanout, in the presence of node churn. In each cycle, a randomly selected 0.2% of the nodes was removed, and replaced by an equal number of newly joined nodes.

Figure 11 shows the miss ratio and the percentage of complete disseminations as a function of the fanout. Although RINGCAST results in a lower miss ratio than RANDCAST for low fanouts (2 to 5), it performs slightly worse for fanouts 6 or higher. It should also be noted that none of the protocols achieves any complete disseminations, except when maximizing the fanout, in which case RANDCAST appears to be performing better again.

By looking at these quantitative graphs alone, one could come to the conclusion that RINGCAST is not any better—if not worse—than RANDCAST when node churn is at stake. A closer, qualitative examination of *which* groups of nodes contribute to each protocol's miss ratio will prove otherwise. As we will see, RINGCAST's miss ratio is almost entirely due to its poor performance at reaching newly joined nodes, while it provides good dissemination guarantees to all older nodes.

Along these lines, we now investigate the relation between a node's *lifetime*, that is, the number of cycles since it joined the network, and its chance of receiving a disseminated message. Figure 12 presents the distribution of node lifetimes after the execution of several thousand cycles, when every node has been removed and reinserted at least once. In fact, Figure 12 plots the exact count of nodes having a given lifetime, aggregated over 100 experiments, in log-log scale. Given that the network consists of 10,000 nodes and the churn rate is 0.2%, at each cycle 20 random nodes are evicted and 20 new are added. Therefore, the number of nodes having a given lifetime cannot exceed 20. For all 100 experiments together, the number of nodes of a given lifetime ranges from 0 to 2000, hence the range of the vertical axis.

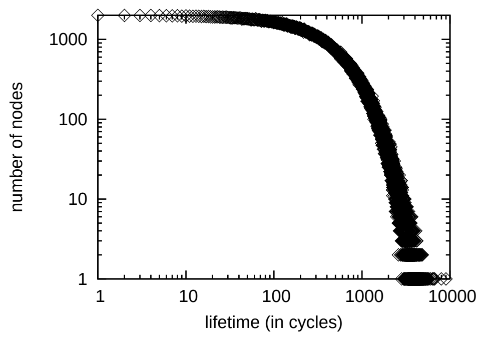
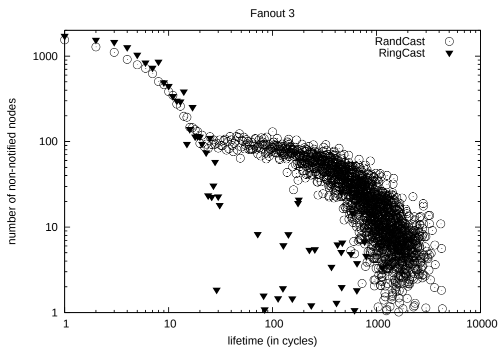


Fig. 12. Distribution of node lifetimes, summed over 100 experiments.

The distribution of lifetimes of nodes that *were not notified* during dissemination, is presented in Figure 13. The distributions for two fanouts are shown, 3 (top) and 6 (bottom). It is clear that in all cases newly joined nodes (i.e., ones that joined up to 20 or 30 cycles ago) experience significantly higher miss ratio than other, older nodes. RINGCAST, in particular, results in quite more misses (notice the log scale) than RANDCAST for these nodes. Nevertheless, for nodes that have been in the network for at least 20 or 30 cycles, it demonstrates a substantially lower miss ratio, almost negligible compared to that of RANDCAST. For instance, let us take a look at dissemination with fanout 6. Although RINGCAST appears to have a higher overall miss ratio than RANDCAST (Fig. 11), it hardly suffers any misses for nodes that joined at least 20-30 cycles earlier, contrary to RANDCAST. Its miss ratio is entirely attributed to misses in newly joined nodes.

The implication behind this observation is worth noting. RINGCAST proves to be a better dissemination tool, except for the first few cycles after a node's join. Once a warm-up period of a few cycles has elapsed, a node receives all disseminated messages with very high probability. For a gossiping period of 10 seconds and a view length $\ell_{cyc} = 20$, the warm-up phase amounts to a bit over 3 minutes. In applications where faster node joins is vital, new nodes can gossip at an arbitrarily higher rate for the first few cycles, to complete their warm-up phase correspondingly fast. However, this is a mere optimization and will not be considered further in this paper.

At this point, it is interesting to understand why new nodes experience more misses, and why this phenomenon is more intense in RINGCAST. Nodes are notified through their incoming links. Their probability of being notified is tightly related to how well they are known by other nodes. A new node joins the network with zero indegree,



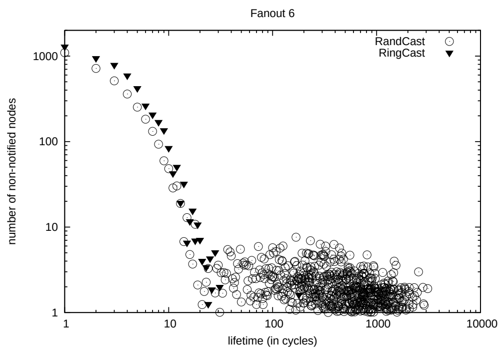


Fig. 13. Distribution of lifetimes of nodes that were not notified, summed over 100 experiments.

and gradually increases it. Until a node's indegree reaches the average indegree of the network, it has less chance to receive a message than older, better connected nodes. This shows clearly in the aforementioned graphs (Fig. 13).

More specifically, a new node's r-link indegree increases by one in each of its first few cycles, and takes approximately ℓ_{cyc} (here $\ell_{cyc} = 20$) cycles to stabilize to the average indegree of the network (which is ℓ_{cyc} too). This is a property of CYCLON, which manages r-links. So, for RANDCAST, which depends solely on CYCLON, we observe a steep decrease in misses for nodes of lifetimes 1 through 20, followed by an immediate stabilization thereafter. This is a direct effect of the join process in CYCLON, which takes approximately ℓ_{cyc} cycles to establish the average number of incoming links.

On the other hand, RINGCAST also depends on VICINITY to form the d-links (i.e., the edges of the ring). However, a node does not benefit from incoming VICINITY links until the appropriate incoming d-links are formed, that is, until it eventually becomes known by its two direct ring neighbors. Generally this does not happen instantly, but may require an undefined—yet small—number of cycles. Until then, a newly joined node relies only on its incoming r-links to receive messages. During that phase, it is clear that newly joined nodes have better chances to receive messages in RANDCAST, where messages are forwarded to F r-links, as opposed to only $F - 2$ r-links in RINGCAST. This explains why RINGCAST exhibits more misses than RANDCAST for nodes that joined roughly in the last 20 cycles (Fig. 13).

Note that the further curve in misses for lifetimes greater than 100 simply follows the lifetime distribution of the general node population (Fig. 12).

8 Conclusions and Discussion

We explored push-based epidemics for information dissemination in very large-scale systems, focusing on limiting redundant messages while retaining strong probabilistic delivery guarantees. We introduced a new class of push-based epidemic dissemination protocols, which combine probabilistic with deterministic features. The probabilistic component contributes in the exponential spreading of messages, while the deterministic component takes care of the "fine-grained job", making sure that a message reaches every single node. We proposed RINGCAST, a new protocol of this hybrid class, and by extensive experimentation in static, dynamic, and catastrophic failure scenarios performed better than RANDCAST, and at a significantly lower communication cost (message overhead).

Some applications may require higher reliability in dynamic environments. Recall from Section 3 that a bidirectional ring is a Harary graph of minimal cut two. One way to increase reliability, would be to design gossiping protocols that form Harary graphs of higher connectivity. Another, simpler way, is to organize nodes in multiple rings, assigning them a different random ID per ring. In both cases, reliability would be improved at the cost of increased gossip traffic.

Another potential optimization is proximity-based dissemination. Proximity can have many faces, e.g., geographic distance, domain name, network hops, etc. In the protocols examined in this paper, proximity is not taken into consideration. For instance, a message originating in the Netherlands could follow a path such as Netherlands → Australia → Switzerland → Canada → Greece → Uruguay → New Zealand. Obviously, such a path is far from optimal.

A straightforward way to partially deal with domain name proximity in RINGCAST, is to incorporate domain names in the VICINITY similarity function. In this version of RINGCAST, a node forms its ID by reversing its domain name (country domain first) and appending a randomly chosen number. I.e., the ID of a node at the `.inf.ethz.ch` domain of the ETH Zurich could be `ch.ethz.inf.1234`. Without any additional modifications, nodes naturally self-organize in a ring sorted by domain name, and domains sorted by country.

Finally, it should be noted that the protocols discussed in this paper are perfectly suitable for *topic-based publish/subscribe* too. In topic-based pub/sub, a number of *topics* are defined, and each event is associated with one of them. All events associated with a topic should be delivered to all nodes subscribed to that topic. The usage of dissemination protocols such as RANDCAST and RINGCAST for event dissemination is straightforward. Each topic forms its own, separate dissemination overlay. Subscribers join the overlay(s) of the topics of their interest. Finally, events are multicast by disseminating them in the appropriate dissemination overlay.

In this research we have explicitly not considered *pull-based* dissemination. We expect it to significantly improve the efficiency of the protocol in terms of reliability. However, additional issues have to be taken into account, such as the pull frequency, the duration for which nodes maintain old messages, the size of buffers on nodes, etc. Pull-based dissemination is left as future work, as it constitutes a natural extension of our current research.

References

1. Kenneth P. Birman, Mark Hayden, Oznur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Trans. Comp. Syst.*, 17(2):41–88, 1999.
2. Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. SCRIBE: A Large-scale and Decentralized Publish-Subscribe Infrastructure. *IEEE JSAC*, 20(8), October 2002.
3. Sameh El-Ansary, Luc Onana Alima, Per Brand, and Seif Haridi. Efficient broadcast in structured p2p networks. In *IPTPS*, pages 304–314, 2003.

4. Patrick Eugster, Sidath Handurukande, Rachid Guerraoui, Anne-Marie Kermarrec, and Petr Kouznetsov. Lightweight Probabilistic Broadcast. In *Int'l Conf. on Dependable Systems and Networks*. IEEE Computer Society, 2001.
5. Patrick Th. Eugster, Rachid Guerraoui, Sidath B. Handurukande, Anne-Marie Kermarrec, and Petr Kouznetsov. Lightweight probabilistic broadcast. *ACM Trans. Comp. Syst.*, 21(4):341–374, December 2003.
6. Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Trans. Netw.*, 5(6):784–803, 1997.
7. Indranil Gupta, Anne-Marie Kermarrec, and Ayalvadi J. Ganesh. Efficient and adaptive epidemic-style protocols for reliable and scalable multicast. *IEEE Transactions on Parallel and Distributed Systems*, 17(7):593–605, 2006.
8. Vassos Hadzilacos and Sam Toueg. *Fault-tolerant broadcasts and related problems*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993.
9. Frank Harary. The maximum connectivity of a graph. In *Proceedings of the National Academy of Sciences*, volume 48, pages 1142–1146, 1962.
10. Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. In *Fifth ACM/IFIP/USENIX International Middleware Conference*, pages 79–98, New York, NY, USA, October 2004. Springer-Verlag New York, Inc.
11. Kate Jenkins and Alan J. Demers. Logarithmic harary graphs. In *ICDCS Workshops*, pages 43–50, 2001.
12. Anne-Marie Kermarrec, Laurent Massoulié, and Ayalvadi J. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Trans. Par. Distr. Syst.*, 14(2):248–258, February 2003.
13. John C.-H. Lin and Sanjoy Paul. Rmtp: A reliable multicast transport protocol. In *INFOCOM*, pages 1414–1424, 1996.
14. Meng-Jang Lin and Keith Marzullo. Directional Gossip: Gossip in a Wide Area Network. In *European Dependable Computing Conference*, pages 364–379, 1999.
15. Meng-Jang Lin, Keith Marzullo, and Stefano Masini. Gossip versus Deterministic Flooding: Low Message Overhead and High Reliability for Broadcasting on Small Networks. In *14th Int'l Symp. Distributed Computing (DISC)*, pages 253–267. University of California at San Diego, 2000.
16. PeerSim, no date. http://peersim.sourceforge.net.
17. Rico Piantoni and Constantin Stancescu. Implementing the swiss exchange trading system. In *FTCS '97: Proceedings of the 27th International Symposium on Fault-Tolerant Computing (FTCS '97)*, page 309, Washington, DC, USA, 1997. IEEE Computer Society.
18. Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. Measuring and Analyzing the Characteristics of Napster and Gnutella Hosts. *Multimedia Systems Journal*, 9(2):170–184, August 2003.
19. Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, June 2005.
20. Spyros Voulgaris, Maarten van Steen, and Konrad Iwanicki. Proactive gossip-based management of semantic overlay networks. *Concurrency and Computation: Practice and Experience*, 2007. To appear.
21. Shelley Q. Zhuang, Ben Y. Zhao, Anthony D. Joseph, Randy H. Katz, and John D. Kubiatowicz. Bayeux: an architecture for scalable and fault-tolerant wide-area data dissemination. In *NOSSDAV '01: Proceedings of the 11th international workshop on Network and operating systems support for digital audio and video*, pages 11–20, New York, NY, USA, 2001. ACM Press.