

Efficient Epidemic-style Protocols for Reliable and Scalable Multicast

Indranil Gupta

Dept. of Computer Science, Cornell University
Ithaca, NY USA 14853
gupta@cs.cornell.edu

Anne-Marie Kermarrec, Ayalvadi J. Ganesh

Microsoft Research, 7 J J Thomson Ave.
Cambridge, UK CB3 0FB
{annemk, ajg}@microsoft.com

Abstract

Epidemic-style (gossip-based) techniques have recently emerged as a scalable class of protocols for peer-to-peer reliable multicast dissemination in large process groups. These protocols provide probabilistic guarantees on reliability and scalability. However, popular implementations of epidemic-style dissemination are reputed to suffer from two major drawbacks: (a) (Network Overhead) when deployed on a WAN-wide or VPN-wide scale they generate a large number of packets that transit across the boundaries of multiple network domains (e.g., LANs, subnets, ASs), causing an overload on core network elements such as bridges, routers, and associated links; (b) (Lack of Adaptivity) they impose the same load on process group members and the network even under reduced failure rates (viz., packet losses, process failures). In this paper, we report on the (first) comprehensive set of solutions to these problems. The solution is comprised of two protocols: (1) a Hierarchical Gossiping protocol, and (2) an Adaptive multicast Dissemination Framework that allows use of any gossiping primitive within it. These protocols work within a virtual peer-to-peer hierarchy called the Leaf Box Hierarchy. Processes can be allocated in a topologically aware manner to the leaf boxes of this structure, so that (1) and (2) produce low traffic across domain boundaries in the network. In the interests of space, this paper focuses on a detailed discussion and evaluation (through simulations) of only the Hierarchical Gossiping protocol. We present an overview of the Adaptive Dissemination protocol and its properties.

1. Introduction

The growth of the Internet and the emergence of application scenarios for large-scale peer-to-peer (p2p) systems is driving the need for scalable and reliable solutions to the multicast problem in distributed process group computing systems. Examples of such systems include publish-subscribe [7], distributed hash tables (DHTs) for file sharing and archival [20], distributed databases [5], distributed

failure detection [22], distributed resource location [21] and virtual synchrony [12]. These applications require a group multicast protocol that is a) reliable, even in the presence of network packet losses and process crashes, and b) scalable, in terms of the load imposed on the network and on participating processes, as the system size grows into thousands of processes.

Traditional solutions to the problem work well in small-scale settings, but suffer drastic reduction in reliability and performance at larger scales. Existing solutions, such as SRM [8] or RMTP [18], that augment best-effort IP multicast with negative or positive acknowledgments, hit a scalability limit in the range of only 50 to 100 participating group members [3, 17, 23], as the number of duplicate copies of the multicast that they generate grows linearly with the system size [3, 17], for any fixed packet loss rate. In addition, the lack of deployment of IP multicast limits applicability of these protocols.

Gossip-based (epidemic) protocols spread multicasts in a process group in a randomized peer-to-peer fashion much like the spread of rumor in society, or of a contagious disease in a population [2]. This results in *probabilistic* behavior of the reliability of multicast delivery at recipients, with per-process overhead and latency rising slowly (poly-logarithmically) with group size. Multicast throughput is resilient to system-wide noise arising from packet losses, and the presence of failed or perturbed processes. At higher failure and loss rates, the probabilistic delivery properties degrade gracefully [3, 5, 14]. Deterministic reliability can then be provided through a low-overhead recovery layer, inserted in the network stack between the epidemic dissemination and application layers. An example is virtually synchronous multicast [12].

Gossip-based protocols have been used in a variety of information dissemination applications [3, 5, 6, 12, 13, 21]. In a gossip protocol, each group member periodically picks a few other members at random and gossips to them about recently received multicast messages. Implementations differ in the length of gossip round and the weighting of members for gossip target selection. However, most conform to

a “Flat Gossiping” technique, where targets are chosen uniformly at random from the group membership [3, 5, 6].

Large networks, such as wide area networks (WAN) or corporate virtual private networks (VPN [1]) spanning several locations, are structured as a hierarchy of *domains*. For example, the Internet is structured hierarchically through domains such as ASs, Class A/B/C networks, subnets, local ethernet etc. Flat Gossiping is oblivious to network topology, and hence generates substantial network traffic into and out of these domains. This translates into significant *network overhead* on connecting core routers, bridges and links. Such behavior is undesirable even if the bandwidth, buffer space, and processing capability of core network elements scales to sustain Flat Gossiping traffic from a single process group, since it limits the deployment of multiple large process groups, as well as affects the behavior of non-gossip-based applications using the network simultaneously.

Previous work [3, 5, 6, 14, 22] has also embodied the belief that receiving multiple copies of a multicast in gossip-based protocols is acceptable overhead in exchange for reliability and scalability. Current gossip protocols continue to incur this overhead even if there are negligible message losses and process failures in the underlying network during the dissemination of a multicast. In other words, they lack *adaptivity*, in the sense of imposing *lower* overhead when there are fewer failures.

This paper proposes two new protocols for gossip-based multicast dissemination that address the issues of network overhead and adaptivity. These are a) a Hierarchical Gossiping algorithm, and b) an Adaptive multicast Dissemination framework which works in conjunction with any gossiping primitive. The protocols work within a virtual (or abstract) hierarchy for the process group, which we call the *Leaf Box Hierarchy*, and which is adapted from the Grid Box Hierarchy of [11]. The protocols exhibit probabilistic reliability similar to that of Flat Gossiping, and have slightly higher latency (by a factor that grows as the logarithm of group size). In exchange, when processes are mapped to leaf boxes in a topologically aware manner, they achieve an order of magnitude reduction in network overhead (the reduction factor is almost linear in group size). For brevity, we primarily discuss the Hierarchical Gossiping algorithm in this paper. The Adaptive Dissemination framework is investigated thoroughly in [10].

The paper is organized as follows. Section 2 describes the Flat Gossiping protocol and motivates the need for network overhead reduction and adaptivity. Section 3 presents algorithms for Leaf Box Hierarchy construction and membership maintenance. The Hierarchical Gossiping protocol and Adaptive Dissemination framework are presented in Sections 4 and 5. Section 6 presents an evaluation of the former protocol through simulations. Section 7 concludes.

2. Previous Work and Design Guidelines

Flat Gossiping: Gossiping protocols such as in [3, 5, 6] can be abstracted into a canonical protocol called “Flat Gossiping”, which disseminates a multicast as follows: each group member that receives the multicast gossips about it for $\log(N)$ rounds, where N is the (approximate) group size and a round is a fixed local time interval at the member. In each round, the group member selects b other members *uniformly at random* (flatly) from the group membership, and sends them a copy of the multicast message. Here, b is a constant. A gossip message is transmitted via a lightweight unreliable protocol such as UDP.

It can then be shown, as in [5, 14], that the probability of a given member receiving the multicast is $1 - \frac{1}{N^b} \cdot (1 + o(1))$. This is the probabilistic reliability achieved by the protocol. The number of rounds between origin and completion of gossip is $O(\log(N))$, which scales well as N increases. The protocol is fault-tolerant: its randomized nature “routes around” member failures and dropped messages.

Network Overhead: Large Internetworks and corporate-wide VPNs are structured hierarchically in domains¹. Figure 1 shows how Flat Gossiping can overload a router in a two-level topology. The same problem can occur in a multi-level topology, especially near the backbone.

Network overhead could be reduced by tailoring the choice of gossip targets to the specific underlying topology as in [5], but a more general strategy applicable to any topology is desirable. One such strategy is proposed in [22], where targets are probabilistically weighted so that, in each gossip round, only a constant number of gossip messages transit out of any given network domain. Another strategy is Directional Gossip [15], which calculates target weights dynamically. Hierarchical gossiping algorithms have also been proposed for managing distributed MIBs and content filtering in publish subscribe systems [7, 21]. However, the protocols in [7, 21], as well as Directional Gossip (the study in [15] is limited to small-scale WANs), are sensitive to member distributions across the topology and hence require careful tuning. Our weighted target selection is similar to that of [22], but our Hierarchical Gossiping protocol provides guarantees on reliability, latency and domain boundary load that hold for any distribution of members across the network topology.

Adaptivity: Adaptivity entails incurring low overhead at low failure rates, with a rise in this overhead only if the local or system-wide failure rate rises. Our approach to adaptivity works by initially attempting to disseminate the multicast via a virtual tree spanning the group members. This virtual tree is not static as in [3], but is constructed dynamically, on the fly (using random seeds) and locally. Each member uses only local information for this construction - its leaf box

¹Recall from Section 1 that we are using “domain” as an abstract name for ASs, subnets, ethernet, etc.

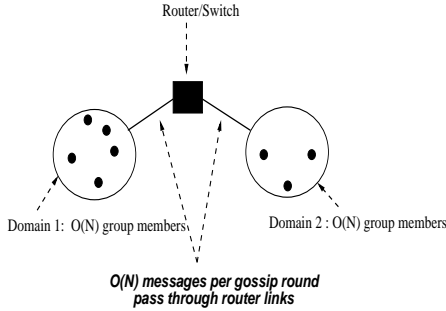


Figure 1. High Network Overhead of Flat Gossiping: In a WAN-wide process group distributed evenly across two domains, when there are $O(N)$ infected members in the group, $O(N)$ gossip messages in each round go across the domain boundaries (and hence the router). The domain boundary bandwidth usage grows linearly with group size.

address and local membership list. The tree is constructed in such a way that it mirrors the location of processes in the Leaf Box Hierarchy (and the network topology if the Leaf Box Hierarchy used a topologically aware mapping).

The protocol transitions to gossiping in subtrees of the Leaf Box Hierarchy where the global tree construction fails due to message losses and process failures. This localizes gossip to only the lossy regions of the Leaf Box Hierarchy.

Deterministically Constrained Flooding [16] can lower overhead when there are a small number of failures. However, as failure rates increase, the reliability of this protocol degrades worse than gossiping [16], while reliability of our Adaptive Dissemination Framework is always bounded from below by that of the gossiping primitive it uses.

Gossip protocols also require each group member to maintain a (small) *view* - a list of its knowledge about other group members. Views may be partial and inconsistent. The view needs to be small to minimize memory storage and target computation time during gossip rounds, yet it should be large enough as to not affect the reliability properties of gossiping, or non-partitionability of the group. It is shown in [6, 14] that a view size of $O(\log(N))$ suffices for Flat Gossiping to retain the same properties as with complete views. Our protocols require a view size of $O(\log^2(N))$.

3. The Leaf Box Hierarchy

The Leaf Box Hierarchy is defined by three parameters that are consistently known at all processes:

- (1) K , a small constant.
- (2) an estimate N of current group size. N is rounded off to the next power of K (say N'), and the number of leaf boxes in the hierarchy is then (N'/K) . Estimation of the value of N may be done either by individual members or by periodical dissemination within the group. Although this scheme defines a *range* of consistent values for N across the group members, for simplicity of exposition, the rest of the paper assumes that $N = N' = a$ power of K .

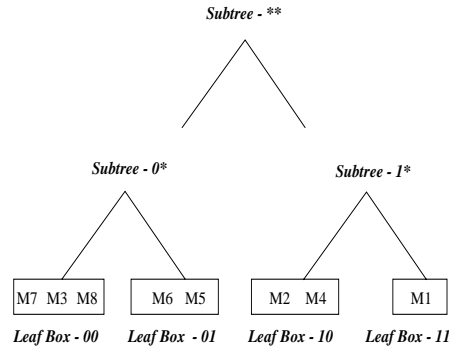


Figure 2. Example of a Leaf Box Hierarchy for a group of 8 members $\{M1 \dots M8\}$ with $K = 2$; there are $N/K = 4$ leaf boxes.

(3) a consistent map function H that maps members to one of N/K leaf boxes.

Each leaf box has a $(\log_K N - 1)$ -digit address in base K . *Subtrees of height j* ($0 \leq j \leq \log_K N - 1$) in the hierarchy contain the set of leaf boxes whose addresses match in the most significant $(\log_K N - j - 1)$ digits. M_i can calculate the leaf box address of any other group member M_l , in its view, by applying the consistent map function H^2 . For $1 \leq j \leq \log_K N - 1$, we denote the j^{th} *internal node ancestor* of a member M_i as the root of the height- j subtree that M_i lies in. Notice that the hierarchy is designed in a peer-to-peer fashion: internal nodes are not associated with any particular group member, but the entire subset of members lying in the leaf boxes of the subtree rooted at the internal node. This design decision avoids the overhead of reorganizing and maintaining the hierarchy on each individual process failure.

Figure 2 shows a Leaf Box Hierarchy for the process group of Figure 1, with parameters $K = 2, N = 8$ and a map function that we will describe shortly. Consider the process M_5 in this figure. M_5 lies in leaf box 01, which in turn lies within the Subtree-0*, which in turn lies within the Subtree-**. Leaf Box 01, Subtree-0* and Subtree-** are thus M_5 's height 0,1,2 -subtrees respectively. The internal nodes labeled Subtree-0* and Subtree-** are the first and second internal node ancestors of M_5 respectively.

Our design assumes consistent knowledge of the tuple (K, N, H) across the group. This is akin to the assumption in most peer-to-peer systems, e.g., Pastry, Chord [20], of the existence of consistent hash functions. The tuple (K, N, H) in the group may need to be changed after a while as the composition and distribution of the group changes. We assume, for simplicity of exposition, that the Leaf Box Hierarchy is not reorganized during the dissemination of a multicast. Section 6 investigates the effect of non-uniform map functions, such as skewed, random, overloading and underloading mappings. Our study indicates that Leaf Box Hier-

²Alternatively, $(M_i, H(M_i))$ may be disseminated within the group along with membership information.

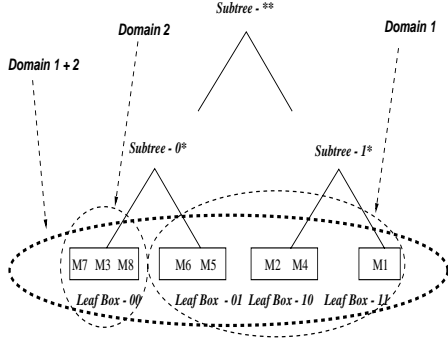


Figure 3. Example of Contiguous Mapping for the topology from Figure 1.

archy performs well even under most degraded conditions, and would require only infrequent reorganization, such as when the group size crosses a power of K or the distribution of members across leaf boxes is too skewed. Multicasts can continue to be disseminated during a reorganization by using the old hierarchy.

3.1. Choosing the Map Function H

Cryptographic hashes, e.g., MD-5, SHA-1, would offer no topological awareness. A geographical partitioning can be used to construct the Grid Box Hierarchy [11] for sensor networks. For a hierarchically structured Internetwork-type network such as a WAN or a corporate VPN, we propose a mapping scheme called the *Contiguous Mapping*, which assigns to each network domain a set of leaf boxes that are *contiguous* in the lexicographic space of leaf box addresses. The assignment is done top-down in the hierarchy of network domains, and a group member then belongs to a randomly chosen leaf box from among those assigned to the smallest domain to which it belongs. Figure 3 shows a possible Contiguous Mapping for the example process group in the topology of Figure 1. This scheme is preferable to associating each subtree with a single domain (e.g., as in the Astrolabe system [21]), since it can be made to fit any distribution of group members across domains.

The Contiguous map function H could be maintained in a distributed manner, or through a fault-tolerant committee of central servers with knowledge of the underlying network topology. The central servers that manage the VPN or the “introduction” operations of processes (i.e., joining and leaving the group), could be used for the latter. Our simulations (Section 6) show that non-uniform map functions do not degrade performance too drastically, so this committee would not need to be highly available. A discussion of the distributed algorithm is beyond the scope of this paper.

Accurate network topology discovery is difficult, and there is often only a loose correlation between the IP address of a host and its network location, except possibly in an administered VPN. In such a case, an *approximate* knowledge of the topology, where each “domain” corresponds to a coarse area rather than just a subnet, could be

used for the map function H . The Landmark Mechanism [19], for example, is fully decentralized and can provide such information. The more accurately the domains estimated by such an approximation correspond to actual network domains, the closer would be the match between the load on core network elements and the predictions of our analysis.

We reiterate that the formal description of protocols within the Leaf Box Hierarchy do not depend on the topological awareness of H . The topological awareness of H has an effect only on the domain boundary load behavior of the Hierarchical Gossiping protocol, with reliability, latency and per-process overhead remaining unaffected at a given group size and system-wide packet loss rate.

3.2. Composition of the View

Each member M_i maintains a view that consists of $\log_K N$ subviews, $View_{M_i}[0]$ through $View_{M_i}[\log_K N - 1]$. $View_{M_i}[j]$ consists of information (such as member identifiers) about at most $subviewsize(N)$ other distinct members *chosen uniformly at random* from among the members lying in the same height- $(\log_K(N) - j - 1)$ subtree as M_i . If there are fewer than $subviewsize(N)$ other members in the height- $(\log_K(N) - j - 1)$ subtree of M_i , M_i includes in $View_{M_i}[j]$ as many such other members as it knows about. From the definition, note that view elements of $View_{M_i}[j]$ might be repeated in any of the $View_{M_i}[\leq j]$ subviews. Views might also be partially inconsistent, i.e., contain members that have failed. Such elements time out and are deleted after a while.

Figure 4 shows the composition of the view at member M_5 in the Leaf Box Hierarchy example of Figure 2. Group member M_5 ’s view would consist of $\log_K(N) = 3$ subviews, each containing $subviewsize(N) = viewfactor * \log_K N$ members chosen uniformly at random from the subgroup of members within the appropriate same subtree as M_5 . Here, $viewfactor$ is a constant.

It is shown in [14] that knowledge of at least $viewfactor * \log_K N$ members in the group at each process suffices to provide a high probability of non-partitionability of the group, where $viewfactor > \log_K e$. Our view maintenance algorithm retains this property through the $View[0]$ sets in the group. Further, [6] shows that sampling the partial view by using a uniform random distribution on the membership, and maintaining it large enough to be able to pick unique gossip targets per multicast, would retain the probabilistic properties of Flat Gossiping. In Section 4.2, we show a similar property for Hierarchical Gossiping w.r.t. the above view composition.

Views in dynamic groups could be maintained using any of several dynamic membership algorithms from the literature, e.g., random probe-based [4], gossip-based heartbeat [22], or graph-based [9], and local views updated probabilistically with new or streaming membership updates. An

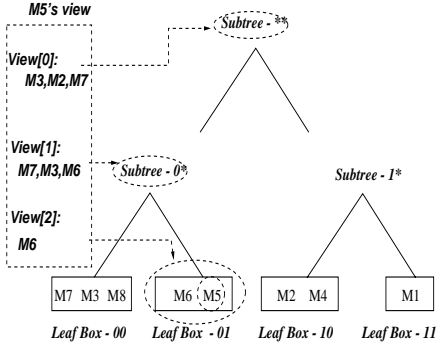


Figure 4. Example of View Maintenance: A typical group member (M_5 shown above) knows about ($view_factor * \log_K N$) other members in each of the Leaf Box Hierarchy subtrees it belongs to. Here $view_factor = 1$. Note that multiple subviews at a member might overlap (e.g., M_7 occurs in $View_{M_5}[0]$ and $View_{M_5}[1]$).

```

Ms:Hierarchical Gossip (Multicast m)
for logK N gossip rounds
  for i := 1 to b
    pick a subtree of height l with probability distribution
      pl =  $\frac{1}{K^{l+1}} * p(N, K)$ 
      where 0 ≤ l ≤ logK N - 1,
      p(N, K) =  $\left[ \sum_{j=0}^{\log_K(N)-1} \frac{1}{K^{j+1}} \right]^{-1}$ 
    pick a node Mtarget uniformly at random from ViewMs[l]
    Send m to Mtarget

```

Figure 5. The Hierarchical Gossiping Protocol.

investigation of interaction with such membership protocols is left to future work.

4. Hierarchical Gossiping Protocol

We discuss and analyze the Hierarchical Gossiping protocol. When combined with the Contiguous Mapping in the Leaf Box Hierarchy, this protocol tackles the network overhead problem. The protocol may also be used within the Adaptive Dissemination framework (Section 5).

4.1. Protocol Description

While Flat Gossiping selects targets uniformly at random, Hierarchical Gossiping prefers targets nearer in the Leaf Box Hierarchy. Each target at group member M_i is selected by (a) first choosing a level l ($0 \leq l \leq \log_K N - 1$) corresponding to M_i 's height- l subtree, and then (b) picking a member uniformly at random from $View_{M_i}[\log_K N - l - 1]$. The height-0 subtree is chosen with probability $\frac{1}{K} * p(N, K)$, the height-1 tree is chosen with probability $\frac{1}{K^2} * p(N, K)$, and so on until the height- $(\log_K N - 1)$ subtree is chosen with probability $\frac{1}{N} * p(N, K)$. Here, $p(N, K) = \left(\sum_{j=0}^{\log_K(N)-1} \frac{1}{K^{j+1}} \right)^{-1}$ is a normalizing factor.

The specification of the rest of the protocol is the same as the Flat Gossiping template described in Section 2. Upon receiving a multicast, a group member M_i gossips about it for $\log_K(N)$ gossip rounds, choosing b target members per round. These parameters and the gossip round duration are fixed as in Flat Gossiping. The pseudo code for the algorithm is presented in 5. Figure 6 shows the (relative) tar-

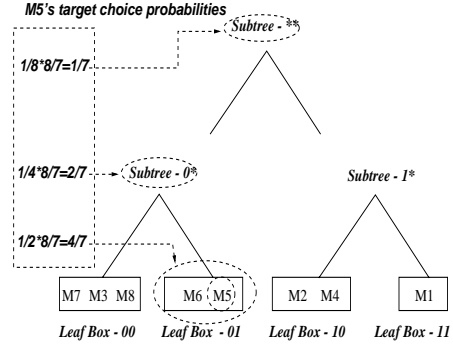


Figure 6. Hierarchical Gossiping at a process: M_5 's gossip-target probability distribution in the example Leaf Box Hierarchy.

get choice probability distribution for each gossip round at member M_5 in the Leaf Box Hierarchy of Figure 2.

4.2. Protocol Analysis

We now analyze the reliability and group member load characteristics of this algorithm. Let $K = 2$, and suppose there are N group members (N being a power of K) distributed uniformly among the leaf boxes by the map function H . These rigid assumptions will be relaxed in the simulations.

The normalization constant $p(N, K)$ is close to 1 for large N when $K = 2$. We study the simplified algorithm where each group member chooses a gossip target from the height- i subtree containing it with probability 2^{-i-1} , and remains silent with the residual probability $1 - p(N, 2) = 1/N$. The reliability analysis below pertains to this modified algorithm, which clearly has a smaller probability of success than the algorithm we actually use.

Define $f(N)$ to be the probability of a given member not receiving any copy of the multicast in a Leaf Box Hierarchy with N members (a height- $(\log_2(N) - 1)$ hierarchy). We now derive a recursion for $f(N)$.

Increasing the group size from N to $2N$ adds an extra level to the Leaf Box Hierarchy (making it a height- $(\log_2(2N) - 1)$ tree), as well as increasing the number of gossip rounds per member by one.

In a Leaf Box Hierarchy of height- $(\log_2(2N) - 1)$, a member M_i lies in the same height- $(\log_2(N) - 1)$ subtree as the sender with probability $\frac{1}{2}$ - such a member will *not* receive the multicast if it does not receive it through gossiping within the height- $(\log_2(N) - 1)$ subtree with N members, as well as from the extra gossip round added due to the number of group members being increased to $2N$. The probability of not receiving the gossip message in the first $\log_2(N)$ rounds is $f(N)$. The probability of receiving the message in the last round depends on the location of nodes that have received the message in the first $\log_2(N)$ rounds. We assume that (a) the number of nodes that have received the gossip message in the first $\log_2(N)$ rounds is equal to its expectation, which is $N(1 - f(N))$, and (b) there is a prob-

ability $\frac{1}{N}(1 - \frac{1}{2} \cdot \frac{1}{2N})$ that each of these nodes will choose M_i as its gossip target each time in the extra round.

The mean-value assumption embodied in (a) is common to most analyses of gossip protocols. The term $(1 - \frac{1}{2} \cdot \frac{1}{2N})$ in assumption (b) is the probability that a gossip remains within the height $\log_2(N) - 1$ subtree containing the sender and M_i ; conditional on this, we assume the message is targeted uniformly at each of the N members of this subtree. While this assumption is clearly false for messages originating from a specific node, we are only assuming it on average for messages originating from all nodes within this subtree. The performance results derived using this assumption seem to be borne out by simulations.

With these assumptions, the probability that M_i does not receive the gossip message in the last round from within its height $\log_2(N) - 1$ subtree is bounded above by

$$\left(1 - \frac{1}{N}\left(1 - \frac{1}{2} \cdot \frac{1}{2N}\right)\right)^{N(1-f(N))b} \leq \exp\left(-\frac{(1-f(N))b}{2}\right).$$

Combining this with the probability of not receiving the message in the first $\log_2(N)$ rounds, the probability of M_i not receiving the message is bounded above by $f(N) \exp(-(1-f(N))b/2)$.

A member M_i lies in a different height- $(\log_2(N) - 1)$ subtree than the sender with probability $\frac{1}{2}$. Such a member will not receive the multicast if either 1) no gossip message is sent from the sender's height- $(\log_2(N) - 1)$ subtree to M_i 's height- $(\log_2(N) - 1)$ subtree, or 2) if (1) does not hold, yet gossiping within this subtree fails to deliver the multicast to M_i , and M_i is not chosen as a gossip target by a node in the other subtree. (2) occurs with probability no more than $f(N)$. For (1) to occur, each of the $b \log_2(2N)$ messages sent by each of the $N(1-f(N))$ nodes that receive the gossip in the sender's subtree should remain within that subtree. Hence, (1) occurs with probability

$$\left(1 - \frac{1}{2} \cdot \frac{1}{2N}\right)^{bN(1-f(N)) \log_2(2N)} \leq \left(\frac{1}{2N}\right)^{b(1-f(N)) \log_2(e)/4}.$$

Thus, we have:

$$\begin{aligned} f(2N) &\leq \frac{1}{2}f(N) \exp\left(-\frac{(1-f(N))b}{2}\right) + \\ &\frac{1}{2} \left[f(N) + \left(\frac{1}{2N}\right)^{\frac{b(1-f(N)) \log_2(e)}{4}} \right] \\ &\leq f(N) \frac{1+e^{-\beta(N)}}{2} + \left(\frac{1}{2N}\right)^{\log_2(e) \cdot \beta(N)/2}, \end{aligned}$$

where $\beta(N) = (1-f(N))b/2$. Clearly, we can choose b so that $f(4) \leq 1/2$. Since $f(N)$ decreases as b increases, we assume without loss of generality that $b \geq 8$, so that $\beta(4) \geq 2$. Now, if $f(N) \leq 1/\sqrt{N}$ and $\beta(N) \geq 2$, we have from above that

$$f(2N) \leq \frac{1}{\sqrt{2N}} \left(\frac{1+e^{-2}}{\sqrt{2}} + \left(\frac{1}{2N}\right)^{\log_2 e - 1/2} \right).$$

Thus, $f(2N) \leq 1/\sqrt{2N}$ if $N \geq 16$. By induction, we have **Reliability:** $\beta(N) \geq 2$ and $f(N) \leq 1/\sqrt{N}$ for all $N \geq 16$. In particular, $f(N) \rightarrow 0$ as $N \rightarrow \infty$.

A (similar) analysis for $K > 2$ is excluded for brevity.

Per member load: Each member sends at most $b \log_K(N)$ copies of any multicast. The average number of copies of any multicast received by a group member is also no more than $b \log_K(N)$.

View size sufficiency: Choosing *viewfactor* as a constant that is large enough to ensure $subviewsize(N) = viewfactor \cdot \log(N) > b$ retains the reliability properties of Hierarchical Gossiping. This is true because each process knows about enough members (chosen uniformly at random from prospective targets) at each level of the hierarchy.

Network Overhead: We calculate the domain boundary load by first measuring the overhead on internal nodes. Visualize each gossip message as "traveling through" the edges of the Leaf Box Hierarchy from the source to destination member. Consider an internal node IN_h at the root of a height- h subtree in the Leaf Box Hierarchy. For each member M_i within the subtree rooted at this internal node, the probability that any given gossip message from M_i will pass through IN_h is: $p(N, K) \cdot \sum_{j=h}^{\log_K N - 1} \frac{1}{K^{j+1}} \cdot \left(1 - \frac{1}{K^{j-1}}\right) \leq \frac{1}{K^h \cdot (K-1)}$. Since the subtree with IN_h as root contains an average of K^{h+1} members, each of which chooses $b \log_K N$ gossip targets per multicast, the average number of copies of a given multicast that will pass *up* through the internal node IN_h is at most $b \cdot p(N, K) \frac{K}{K-1} \log_K(N)$. A symmetrical number of gossip messages passes *down* through IN_h . Thus, any internal node sees, in expectation, at most $2b \cdot p(N, K) \frac{K}{K-1} \log_K(N)$ copies of a given multicast.

Now consider a contiguous set of leaf boxes assigned to a domain in the Contiguous Mapping scheme. Consider the set of internal nodes that are non-common ancestors of the leftmost and rightmost leaf boxes in the set. A gossip message passing into or out of the domain (i.e., through its boundary) will also pass through at least one of these internal nodes. It is important to notice that *this fact is independent of the size of the domain, or the number of group members that it contains*. Since there can be at most $(2 \log_K N - 4)$ non-common internal node ancestors of the leftmost and rightmost leaf boxes in a domain, the average replication factor of a multicast across *any* domain boundary can be bounded from above by $2b \cdot p(N, K) \frac{K}{K-1} (\log_K N) (2 \log_K N - 4) = O(\log^2(N))$.

5. Adaptive Dissemination Protocol Overview

The Adaptive Dissemination framework consists of two subprotocols: *Tree Dissemination* and *Gossip*. Any gossip protocol, e.g., Flat Gossiping, Hierarchical Gossiping, etc., could be used as the *Gossip* primitive. Each message is either a TREE message or a GOSSIP message.

Each message also bears a hop number. The multicast sender first sends a TREE message with hop number

0 to itself. A group member M_i , on receiving a multicast through a TREE message m with hop number $m.h$ ($0 \leq m.h < \log_K(N) - 1$), attempts to forward this message to K randomly chosen child members, one in each of the child subtrees of the $(\log_K(N) - m.h - 1)^{th}$ internal node ancestor of M_i in the Leaf Box Hierarchy. To avoid a member receiving more than one TREE message, this type of message also bears the list of ancestor members that it has traversed so far - the size of this list can be shown to be constant in expectation and $(\log_K(N) - 1)$ at worst.

Tree Dissemination thus creates a dynamic multicast tree mirroring the Leaf Box Hierarchy. TREE messages are acknowledged by the immediate child members. A TREE message with hop number $(\log_K N - 1)$ or non-receipt of an acknowledgment initiates GOSSIP messages, through a *Gossip* primitive, in the relevant height- $(\log_K N - h - 1)$ subtree where the failure has occurred.

It is shown in [10] that Adaptive Dissemination gives at least as good probabilistic reliability as when *Gossip* is used solely. In absence of failures and message losses during the multicast dissemination, the protocol delivers at most a constant number (i.e., independent of group size) of copies of the multicast at each member. Network and per-member load rises with the rate of failures and message loss. With the Contiguous Mapping, the average number of TREE messages passing through any domain boundary is at most $O(\log(N))$ in expectation, and $O(\log^2(N))$ at worst.

6. Experimental Results

We present experimental studies comparing the scaling trends of Hierarchical Gossiping (“HierGssp”) with Flat Gossiping (“FlatGssp”). We use metrics of per-member load, reliability, domain boundary load, and latency. Section 6.1 reports on topology-independent simulations with uniform, skewed, overloading, underloading and random map functions. While these experiments ignore the network topology, they are nevertheless useful for measuring reliability, latency and domain boundary load in the leaf box hierarchy, which are all topology-independent for HierGssp. The only topology-dependent metric is the load on router links. Section 6.2 presents experiments with transit-stub network network topologies, and shows that HierGssp’s reduction of domain boundary load translates into a reduction of load on router links.

6.1. Topology Independent Simulations

The Leaf Box Hierarchy is constructed using a branching factor of $K = 2$. N is used to denote the actual group size (which is also the same as the group size estimate, except in (3) below). Protocol behavior under different map functions is investigated:

- (1) **The default uniform mapping:** exactly $K = 2$ members per leaf box; No suffix in plots;
- (2) **Skewed mappings:** modeled with a parameter called *mask*. Every 2^{mask} -th leaf box contains $2^{mask} + 1$ members. Other leaf boxes contain one member each. *mask* =

0 gives the default distribution, and higher values of *mask* model more skewed mappings. Denoted in plots with suffix: “*mask* =”;

(3) **Overloading (resp. underloading) mappings:** by using the default distribution, but on a Leaf Box Hierarchy with $\frac{N}{K^2}$ (resp. N) leaf boxes. Suffix in plots: “OL” (resp. “UL”);

(4) **Random mapping:** obtained by using the C library function `rand()` to determine members’ leaf box addresses. Suffix in plots: “rand”.

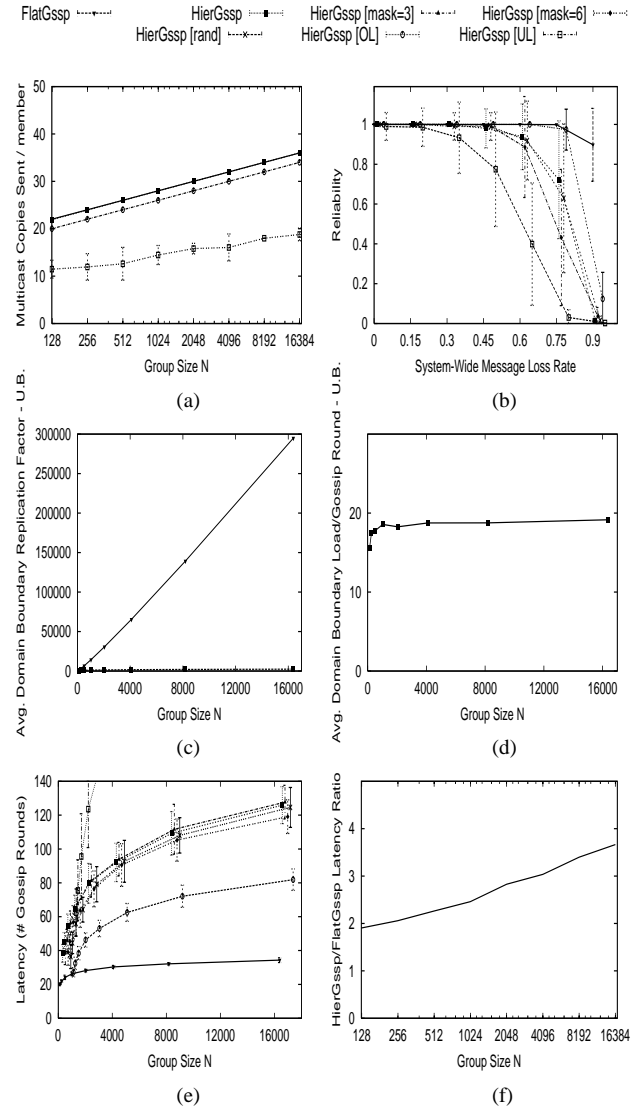


Figure 7. Topology Independent Simulations: Flat Gossip versus (New) Hierarchical Gossip. In (b), $N = 2047$. In the other plots, the system-wide message loss rate=0%. Overlapping data-points in (b) and (e) horizontally perturbed slightly for clarity. The plots are explained in Section 6.1.

(1) models performance with perfect knowledge of N and the network topology, e.g., soon after a Leaf Box Hierarchy reorganization. (2)-(4) model staleness of H and the group size estimate N in the tuple (K, N, H) (which

might occur due to process joins and failures since the last hierarchy reorganization).

Both gossiping protocols (Flat and Hierarchical) are configured as follows. Protocol rounds are synchronous at group members³. Each member, on receiving a multicast, gossips about it for $\log_K(N) + 4$ rounds - the extra 4 rounds are added for robust dissemination of gossip at small group sizes. During each round, a member selects $b = 2$ gossip targets using the (Flat or Hierarchical) distribution.

The membership view at each member M_i is constructed with *viewfactor* = 4 in the algorithm of Section 3.2. The view size is thus bounded by $4(\log_2(N))^2$ and increases slowly with group size. At $N = 16384$, the average measured view size is about 600, or less than 5 % of group size.

General Scaling and Comparative Trends: All averages in the plots of Figure 7 are measured over 50 random runs of the protocol, and error bars show one standard deviation. Figure 7(a) shows the baseline for the comparison; the measured average overhead per group member per multicast - this grows as the logarithm of group size. HierGssp, default and random and for different values of *mask*, have the same overhead. The UL and OL overheads are different; in OL since the number of gossip rounds falls with decreasing hierarchy height, and in UL due to lower reliability.

Figure 7(b) shows that FlatGssp tolerates up to 75 % system-wide message loss while HierGssp (default) starts to degrade at message loss rates of 30 %. This occurs because weighing target choices sends more copies (gossip messages) of a multicast to group members that have previously received a copy. However, loss rates beyond 30% indicate extreme congestion in the network, so HierGssp and FlatGssp have comparable reliability in practical operating ranges. Skewed mappings in HierGssp lead to similar behavior as the default, and are not shown. OL (resp. UL) gives a higher (resp. lower) reliability than the default as the Leaf Box Hierarchy has one level less (resp. more), and is thus flatter (resp. more hierarchical). The rand mapping exhibits high variability in reliability beyond a 20% loss rate. Member failures give similar results, and these are omitted for brevity.

Figure 7(c) compares the upper bound on the average replication factor of a multicast through any domain boundary. For FlatGssp, this corresponds to the worst case configuration of Figure 1. For (default) HierGssp, this is derived from a summation of replication factors at internal nodes (Section 4.2). The upper bound grows linearly with group size for FlatGssp (300000 at 16384 members), whereas it increases as the square of the logarithm of the group size for HierGssp (2410 at 16384 members). The upper bound on average domain boundary load (copies of a multicast/time

³This is a conservative assumption since a gossip is disseminated *faster* with asynchronous gossip rounds rather than with synchronous gossip rounds [22].

unit) passing through any domain boundary is the replication factor divided by dissemination latency. As group size rises, domain boundary load in FlatGssp increases linearly, while HierGssp (Figure 7(d)) quickly converges to a limit *independent of group size*. In this experiment, the limit appears to be around 20 gossips/round.

Figure 7 (e) shows that average latency until a HierGssp epidemic dies out is independent of mapping skewedness, and increases sublinearly with group size. The latency is higher than in FlatGssp, but Figure 7(f) shows that the ratio is low; it stays below 4 at 16384 members. While FlatGssp’s latency is logarithmic in group size, HierGssp’s latency appears to increase as the square of this logarithm. In general though, both protocols have low latencies - with a 1 s gossip round, HierGssp’s (resp. FlatGssp) latency at 16384 members is a little over 2 min (resp. 34 s). The OL mapping gives a lower latency than the default as target selection is “flatter” than default HierGssp. As expected, UL results in a very high latency.

Our conclusion is primarily that compared to FlatGssp, HierGssp (default) achieves an order of magnitude lower network overhead at the cost of a small decrease in fault-tolerance and a small increase in multicast dissemination latency. Further, the tradeoff space spanned by varying gossip round length is better in HierGssp than in FlatGssp. With increasing group size, the product of latency and domain boundary replication factor would be sublinear in the former, and at least linear in the latter. The variation of reliability and latency of HierGssp are not affected by mappings that are skewed or overloading (although they are with underloading and random mappings).

Non-uniform Leaf Box Mappings: Non-uniform mappings do however influence distribution of load on internal nodes of the hierarchy, and thus multicast replication factor across domain boundaries⁴. In Figure 8, we show the variation of internal node loads in a height-10 Leaf Box Hierarchy. A uniform mapping imposes similar load distributions on all internal nodes at all levels (Figure 8(a); levels are marked down from the root which is level 0). As the skewedness of the mapping rises, the number of messages passing through some internal nodes increases (Figures 8(b,c)). From the addresses of the overloaded internal nodes, we observe that these nodes are the first few direct ancestors of the leaf boxes that contain a large number of members. However, this increase in overhead is slow - even at a skew of *mask* = 6, where the largest leaf boxes contain 65 members, no internal node has a replication factor beyond 257; this compares with *mask* = 0 (Figure 8(a)) where the maximum replication factor at a level 9 node

⁴Recall from the last paragraph of Section 4.2 that the replication factor across a domain boundary can be bounded by summing the loads on the non-common internal node ancestors of the leftmost and rightmost subtree assigned to this domain.

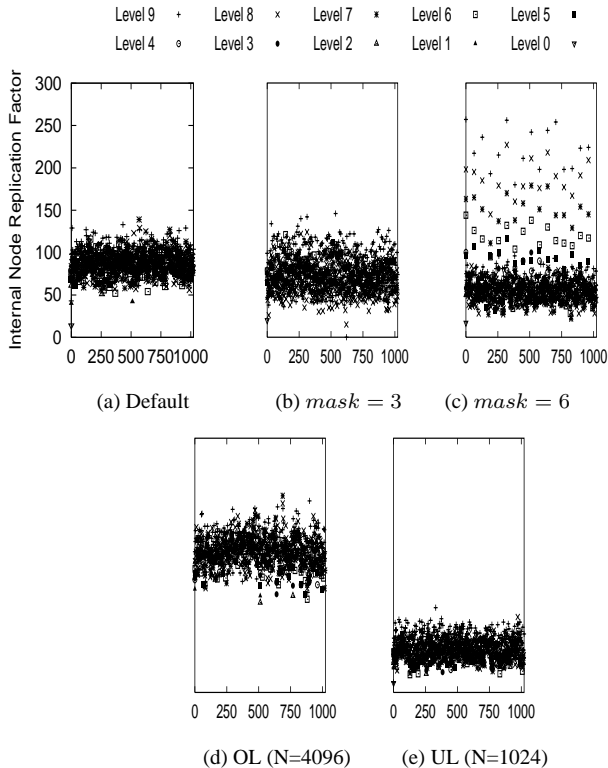


Figure 8. Topology Independent Simulations: Internal node load due to Hierarchical Gossip with different mapping schemes in a height-10 Leaf Box Hierarchy. The X-axis shows internal node addresses (obtained by zeroing out “*”’s). Level 0 is the root. $N = 2047$ in (a)-(c). The plots are explained in Section 6.1.

(i.e., root of a height-1 subtree) is 133, and at any internal node is 139. This occurs since the view composition replicates members across subviews. Figures 8(d,e) show that the UL (resp. OL) mapping, as expected, causes a halving (resp. doubling) of domain boundary load w.r.t. the default mapping. Plots for the random mapping depend on the exact distribution of members and are not shown; intuitively however, the *average* load on any given internal node over all possible random distributions will look the same as Figure 8(a). We conclude that domain boundary load would not increase significantly with mappings that are skewed, overloading or underloading.

Conclusions about Leaf Hierarchy reorganization: Uniform, skewed and OL mappings perform quite well under the metrics considered, while rand and UL mappings perform worse on reliability and latency, though not on domain boundary load. We conclude that the Leaf Box Hierarchy would require reorganization when the group size crosses a power of K (particularly when this size decreases) or when the skew crosses a (pre-specified) threshold.

6.2. Simulations on Transit-Stub Topologies

A reduction of domain boundary load may not necessarily translate into a reduction of overhead on core network elements such as routers. For example, a router associated with a boundary domain may have to route gossip messages

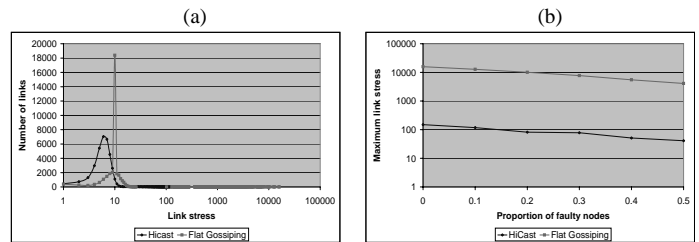


Figure 9. Transit-stub Topologies: Comparison of network link stress in a transit-stub topology in Hierarchical Gossiping (HiCast) and Flat Gossiping: (a) Distribution of link stress in a 16,000 sized group; (b) Maximum link stress in the presence of failures in a 16,000 sized group.

sent from other outside domains.

Hence, we compared the performance of C# implementations of the two multicast protocols in groups spread within transit-stub network topologies generated from the Georgia Tech model (GT-ITM) [24]. The two implementations are called “HiCast” (based on Hierarchical Gossiping), and Flat Gossiping, respectively. Two discrete event simulation scenarios were investigated - a 16,000 sized group in a network of 600 core routers nodes and 60,000 LAN nodes, and a 64,000 sized group in a network of 5050 core routers nodes and 505,000 LAN nodes, with nodes assigned to routers (and members to nodes, one per node) via a uniform distribution. Network routing delays were modeled through a 1 ms delay on LAN links and a 40.5 ms delay on core links. Shortest path routing was used. The maximal process group membership was static. Contiguous Mapping was used to map group members uniformly into the Leaf Box Hierarchy that was constructed with $K = 2$ and an accurate estimate of group size. Each multicast was gossiped with $b = 2$, for 10 rounds in Flat Gossiping (20 rounds in HiCast) in the 16,000 sized group, and for 12 rounds in Flat Gossiping (25 rounds in HiCast) in the 64,000 sized group.

We measured the “stress” on a network link - the number of copies of a multicast that pass through it. Figure 9(a) shows the distribution of stress across network links in the absence of failures while Figure 9(b) shows the variation of peak stress with failure rate. In spite of the higher load per process in HiCast compared to Flat Gossiping, HiCast imposes a load on core network links that is an order of magnitude lower than Flat Gossiping. Similar behavior was observed in the 64,000 sized group, verifying our hypothesis that HiCast lowers overhead on core network elements.

Figure 10 shows the variation of reliability with process failure rate and corroborates the data of Figure 7(b).

7. Conclusions

Flat epidemic-style algorithms for multicast dissemination offer good probabilistic guarantees on reliability and scalability. However, they lack adaptivity and impose high network overhead across domains in the network, which translates to an overloading of core bridges, routers and associated links. Other epidemic-style algorithms either solve

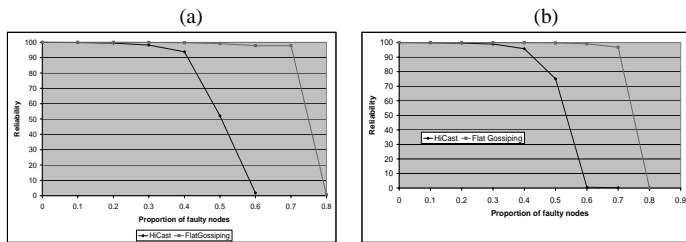


Figure 10. Transit-stub Topologies: Comparison of resilience to process failures in a transit-stub topology between Hierarchical Gossiping (HiCast) and Flat Gossiping in (a) a 16,000 sized group and (b) a 64,000 sized group.

this problem partially, or are not generally applicable. We have presented the (first) comprehensive set of solutions to avoid these drawbacks. The new algorithms use an abstract hierarchy defined on the process group, called the Leaf Box Hierarchy. We have detailed and analyzed the Hierarchical Gossiping algorithm, and presented an overview of an Adaptive multicast Dissemination Framework. The Leaf Box Hierarchy can be mapped on to the member distribution on any Internetwork-type topology by using the Contiguous Mapping, ensuring that the Hierarchical Gossiping and Adaptive Dissemination protocols impose low overhead on network elements such as core routers. Simulation results show that, compared to a canonical Flat Gossiping protocol, Hierarchical Gossiping imposes lower network overhead than, but suffers small decrease in reliability and small increase in latency. These algorithms require a view size that increases with the square of the logarithm of the number of processes participating in the group. Simulations also predict that the Leaf Box Hierarchy behaves well under uniform, skewed and overloaded mappings, and would require infrequent reorganization.

References

- [1] Secure Computing. White Paper: an overview of Virtual Private Networks (VPNs), Mar. 2000.
- [2] N. T. J. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Hafner Press, 2nd edition, 1975.
- [3] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal Multicast. *ACM Tr. Computer Systems*, 17(2):41–88, 1999.
- [4] A. Das, I. Gupta, and A. Motivala. SWIM: Scalable Weakly-consistent Infection-style process group Membership protocol. In *Proc. 2002 Intl. Conf. Dependable Systems and Networks (DSN '02)*, pages 303–312, 2002.
- [5] A. Demers, D. Greene, C. Hauser, W. Irish, and J. Larson. Epidemic algorithms for replicated database maintenance. In *Proc. 6th Annual ACM Symp. Principles of Distributed Computing (PODC '87)*, pages 1–12, 1987.
- [6] P. Eugster, R. Guerraoui, S. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight Probabilistic Broadcast. In *Proc. 2001 Intl. Conf. Dependable Systems and Networks (DSN '01)*, pages 443–452, 2001.
- [7] P. T. Eugster and R. Guerraoui. Probabilistic Multicast. In *Proc. 2002 Intl. Conf. Dependable Systems and Networks (DSN '02)*, pages 313–322, 2002.
- [8] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Tr. Networking*, 5(6):784–803, Dec. 1997.
- [9] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. SCAMP: peer-to-peer lightweight membership service for large-scale group communication. In *Proc. 3rd Intl. Workshop Networked Group Communication (NGC '01)*, pages 44–55. LNCS 2233, Springer, 2001.
- [10] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Adaptive and efficient epidemic-style protocols for reliable and scalable multicast. Technical Report, Microsoft Research, Cambridge, UK, 2001.
- [11] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proc. 2001 Intl. Conf. Dependable Systems and Networks (DSN '01)*, pages 433–442, 2001.
- [12] I. Gupta, R. van Renesse, and K. Birman. Fighting fire with fire: using randomized gossip to combat stochastic scalability limits. *Journ. Quality and Reliability Engg. Intl.*, 18:165–184, May/June, 2002.
- [13] A. Iamnitchi, M. Ripeanu, and I. Foster. Locating data in (small-world?) p2p scientific collaborations. In *Proc. 1st Intl. Workshop Peer-to-Peer Systems (IPTPS '02)*, Mar. 2002.
- [14] A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Reliable probabilistic communication in large-scale information dissemination systems. Technical Report MMSR-TR-2000-105, Microsoft Research Cambridge UK, 2000.
- [15] M.-J. Lin and K. Marzullo. Directional Gossip: Gossip in a Wide Area Network. In *Proc. European Dependable Computing Conference*, pages 364–379. LNCS 1667, Springer, 1999.
- [16] M.-J. Lin, K. Marzullo, and S. Masini. Gossip versus Deterministically Constrained Flooding on small networks. In *Proc. 14th Intl. Conf. Distributed Computing (DISC 2000)*, pages 253–267. LNCS 1914, Springer, 2000.
- [17] M. Lucas. *Efficient Data Distribution in Large-Scale Multicast Networks*. PhD thesis, U. Virginia, May 1998.
- [18] S. Paul, K. Sabnani, and S. Bhattacharya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journ. Selected Areas in Communications*, 15(3):405–421, 1997.
- [19] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proc. 21st IEEE INFOCOM*, New York, Jun. 2002.
- [20] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. ACM SIGCOMM Conf.*, pages 149–160, 2001.
- [21] R. van Renesse and K. Birman. Scalable Management and Data Mining using Astrolabe. In *Proc. 1st Intl. Workshop Peer-to-Peer Systems (IPTPS '02)*, Mar. 2002.
- [22] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Proc. Middleware '98*, pages 55–70. Springer, 1998.
- [23] Z. Xiao and K. Birman. A randomized error recovery algorithm for reliable multicast. In *Proc. 20th IEEE INFOCOM*, volume 1, pages 239–248, 2001.
- [24] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proc. 15th IEEE INFOCOM*, volume 2, pages 594–602, 1996.