Efficient and Adaptive Epidemic-style Protocols for Reliable and Scalable Multicast

Indranil Gupta* Anne-Marie Kermarrec† Ayalvadi J. Ganesh‡

indy@cs.uiuc.edu, Anne-marie.Kermarrec@irisa.fr, ajg@microsoft.com

Abstract

Epidemic-style (gossip-based) techniques have recently emerged as a class of scalable and reliable protocols for peer-to-peer multicast dissemination in large process groups. However, popular implementations of epidemic-style dissemination 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 describe two protocols to address these problems: (1) a Hierarchical Gossiping protocol, and (2) an Adaptive Dissemination framework (for multicasts) 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, and induce minimal overhead when there are no failures.

Keywords (from IEEE taxonomy): Distributed Systems, Reliability, Network Communication, Simulation.

Other Keywords: Multicast, Reliability, Epidemics, Gossip, Adaptivity, Topology-awareness.

1 Introduction

Context and Previous Work: The emergence of application scenarios for large-scale peer-to-peer (p2p) systems on the Internet is driving the need for scalable and reliable solutions to the multicast problem in distributed process groups. Examples of

*(corresponding author) Dept. of Computer Science, University of Illinois at Urbana-Champaign, IL USA 61820. Ph: +1 217 265 5517. Fax: +1 217 265 6494.

†IRISA/INRIA Campus Universitaire de Beualieu, 35042 Rennes Cedex, France. Ph: +33 2 9984 2598.

‡Microsoft Research, 7 J J Thomson Ave., Cambridge, UK CB3 0FB. Ph: +44 1223 479700.

such systems include publish-subscribe [9], distributed hash tables (DHTs) for file sharing and archival [25], replicated databases [7], distributed failure detection [27], distributed resource location [26] and virtual synchrony [14]. 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 load imposed on the network and participating processes, as system size grows into hundreds or even thousands of processes.

Network layer protocols such as SRM [10] or RMTP [22] augment best-effort IP multicast by using negative or positive acknowledgments to repair packet losses. However, the number of request and repair messages grows linearly in the group size, which limits their scalability [3, 19, 28]. In addition, the lack of deployment of IP multicast limits their applicability. These factors have stimulated interest in application-level multicast. Several application layer protocols, e.g., [1, 4], have been proposed.

This paper focuses on gossip-based (epidemic-style) protocols for application layer multicast. Gossip-based protocols spread multicasts in a process group in a randomized peer-to-peer fashion much like the spread of rumors in society, or of a contagious disease in a population. They have been used in a variety of information dissemination applications [3, 7, 8, 14, 15, 26]. The majority of gossip protocols proposed to date (see, e.g., [3, 7, 8]) can be abstracted into a canonical protocol called "Flat Gossip", which disseminates a multicast as follows: each group member that receives the multicast gossips about it for $\log(N)$ rounds, where N is the 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 (or from the members it knows about), and sends each of these targets a copy of the multicast message.

This protocol is reliable in a probabilistic sense: it can be shown, by adapting the analysis in [16], that the probability of a given member receiving the multicast is:

$1 - \frac{1}{N^b} \cdot (1 + o(1))$. The protocol is fault-tolerant as its randomized nature helps it "route around" member failures and dropped messages. For example, if the network drops a fraction f of messages at random, then the probability that any given member receives the multicast is $1 - \frac{1}{N^{(1-f)b}} \cdot (1 + o(1))$. 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 [14]. We do not study the deterministic extension here; it is however feasible, since the good probabilistic properties of our protocols would make the recovery layer very low-overhead.

Large networks, such as wide area networks (WAN) or corporate virtual private networks (VPN) 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 Ethernets 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 since it limits the deployment of multiple large process groups, as well as degrades non-gossip-based applications sharing the network.

Network overhead could be reduced by tailoring the choice of gossip targets to the specific underlying topology as in [7], but a more general strategy applicable to any topology is desirable. One such strategy is proposed in [27], 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 [17], which calculates target weights dynamically. Hierarchical gossiping algorithms have also been proposed for managing distributed MIBs and for content filtering in publish-subscribe systems [9, 26]. However, the protocols in [9, 26], as well as Directional Gossip, are sensitive to member distributions across the topology and hence require careful tuning.

Another reason gossip protocols impose a high network load is their high degree of redundancy: each node receives of the order of $\log(N)$ copies of each multicast message, on average. Previous work [3, 7, 8, 16, 27] has embodied the view that receiving multiple copies of a multicast in gossip-based protocols is acceptable overhead in exchange for reliability. Current gossip protocols continue to incur this overhead even if there are *negligible* message losses and process failures in the underlying network during the multicast. In other words, they lack adaptivity, in the sense of imposing lower overhead when there are fewer failures. Deterministically Constrained Flooding [18] is a technique that can lower overhead when there are a small number of failures. However, as failure rates increase, the reliability of this protocol degrades faster than that of gossip, whereas the reliability of the adaptive scheme that we propose is always bounded from below by that of the gossiping primitive it uses.

Contributions of this Paper: In this paper, we propose two new protocols for gossip-based multicast that address the issues of network overhead and adaptivity. These are a) a Hierarchical Gossiping algorithm, and b) an Adaptive Dissemination (Multicast) framework. Both these protocols work within a virtual hierarchy for the process group, called the *Leaf Box Hierarchy*. This is a generalization of the Grid Box Hierarchy of [13], and can be constructed in a topology-aware manner in order to achieve low network load.

The Hierarchical Gossiping algorithm [12] uses a weighted target selection similar in spirit to that of [27]. We derive probabilistic guarantees on reliability, latency and domain boundary load that hold for *any* distribution of members across the network topology. Simulations confirm that this achieves a substantial reduction in network overhead (load across network domain boundaries and on router links) compared to flat gossip schemes, while achieving similar probabilistic reliability and only slightly higher latency.

The Adaptive Dissemination framework can be combined with any gossiping primitive to achieve reliability comparable to the basic gossiping approach, with a reduction in overhead when the failure rates (of members and message deliveries) are low. It works by initially attempting to disseminate the multicast via a virtual tree spanning the group members. This virtual tree is constructed dynamically, on the fly (using random seeds) and locally. Each member uses only local information for this construction - its leaf box 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 uses 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 or process failures. This localizes gossip to only the lossy regions of the Leaf Box Hierarchy.

The paper is organized as follows. Section 2 presents algorithms for Leaf Box Hierarchy construction and membership maintenance. The Hierarchical Gossiping protocol and Adaptive Dissemination framework are presented in Sections 3 and 4. Section 5 presents an evaluation of the protocols through simulations. Section 6 concludes.

2 The Leaf Box Hierarchy

The Leaf Box Hierarchy is generalized from the Grid Box Hierarchy of [13], and is defined by three parameters that are assumed to be known consistently at all processes: a small constant, K, an estimate N of the current group size (rounded up to the next higher power of K), and a consistent map function H that maps each member into a leaf box. Estimation of N may be done either by individual members or by periodic dissemination within the group. The number of leaf boxes in the hierarchy is N/K and the function H

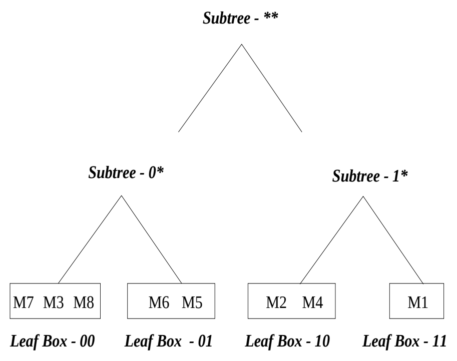


Figure 1: **Example of a Leaf Box Hierarchy** for a group of 8 members $\{M1 \ldots M8\}$ with $K = 2$; there are $N/K = 4$ leaf boxes. The process M_5 in this figure lies in leaf box 01, which lies within Subtree-0*, which in turn lies within Subtree-**. Leaf Box 01, Subtree-0* and Subtree-** are thus 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.

is used to map each member to one of these leaf boxes.

Each leaf box has a $(\log_K(N) - 1)$-digit address in base K. Subtrees of height j in the hierarchy contain the set of leaf boxes whose addresses match in the $(\log_K(N) - 1 - j)$ most significant digits; thus, a subtree of height 0 is a leaf box, and a subtree of height $(\log_K(N) - 1)$ is the whole group. A member 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. We define the j^{th} internal node ancestor of a member M_i as the root of the height-j subtree that M_i lies in.

Figure 1 shows a Leaf Box Hierarchy with parameters $K = 2$, $N = 8$ and a map function that we will describe soon. Note that the hierarchy is designed in a peer-to-peer fashion: internal nodes are not associated with any particular group member, but with the entire subset of members lying in the leaf boxes of the subtree rooted at the internal member. This design decision avoids the overhead of reorganizing and maintaining the hierarchy on each individual process failure.

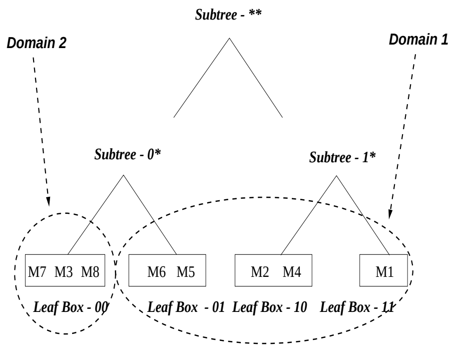


Figure 2: **Example of Contiguous Mapping** for the process group from Figure 1.

Using cryptographic hashes (e.g., MD-5 or SHA-1, utilized by peer to peer systems such as Chord [25] and Pastry [24]) for H offers no topological awareness. Instead, for a hierarchically structured network such as a WAN or a corporate VPN, we propose a mapping scheme called the **Contiguous Mapping**. This scheme 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. This scheme is preferable to associating each subtree with a single domain (e.g., as in Astrolabe [26]), since it can be made to fit any distribution of group members across domains. Figure 2 shows a possible Contiguous Mapping for the example process group in Figure 1.

Reorganizing (K, N, H)**:** Our design assumes consistent knowledge of the tuple (K, N, H) across the group. In a dynamic group where the composition and distribution of the group changes continuously, the map function and associated consistent value of N may need to be changed through a group-wide *reorganization*. Although a detailed study of reorganization is beyond our scope here, we (1) briefly detail the feasibility of such reorganization, and (2) show in Section 5 that protocols such as hierarchical gossiping degrade only slightly even if reorganization is infrequent, and (N, H) is outdated.

The reorganization can be carried out in either a centralized manner at the introducer member (which all new joiners contact), or in a distributed fashion. The group size N can be kept track of continuously, either at the introducer or through a distributed algorithm such as in [20]. Note that *only when this estimate crosses a threshold compared to the current N being used in the Leaf Box Hierarchy, does a reorganization need to be initiated.*

Although most peer to peer systems are dynamic, the number of nodes does not change much, even over a long period of time [2]. Thus, we expect that reorganization will be required only when the leaf boxes become unbalanced. Network locations of hosts can be found using schemes like Landmark [23], GNP [21], Vivaldi [5], and others, provide an approximate knowledge of the topology. When too many leaf boxes become overloaded, underloaded or skewed, a reorganization can be initiated [1].

Finally, we emphasize that the formal description of protocols within the Leaf Box Hierarchy do not depend on the topological awareness of H. Analytically, in a world where the latency and message loss probability on all member-to-member routes are assumed to be the same (this is an ideal assumption) the topological awareness of H has an effect only on the domain boundary load of the Hierarchical Gossiping protocol, but does not affect its latency, reliability, or per-member overhead.

Composition of the View: Multicast protocols require each group member to maintain a *view* - a list of other group members that it knows about. Views may be partial and inconsistent. The view needs to be small in order to minimize memory storage and target computation time during gossip rounds. Yet it should be large enough to prevent

[1]More accurate estimation and network location algorithms will lead to a better match between the predictions of our analysis and the actual load on core network elements.

partitioning of the group, and to ensure that the reliability of gossip is not degraded.

We now describe the view composition required for the gossip protocol to be described in the next section. Each member M_i maintains a view that consists of $\log_K N$ subviews. The j^{th} subview, $View_{M_i}[j]$, 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-($\log_K(N) - 1 - j$) subtree as M_i. If there are fewer than $subviewsize(N)$ other members that M_i knows about in this subtree, then it includes all such members in $View_{M_i}[j]$. Note from the definition that subviews are not required to be disjoint; the same member might be present in $View_{M_i}[j]$ for more than one value of j. Views might also be partially inconsistent, i.e., contain members that have failed. Such elements time out and are deleted after a while.

Figure 3 shows the composition of the view at member M_5 in the Leaf Box Hierarchy example of Figure 1. It was shown in [16] that, if $viewfactor > \log_K e$, then knowledge of at least $viewfactor * \log_K N$ uniformly randomly chosen group members at each process suffices to provide a high probability of non-partitionability of the group. Our view maintenance retains this property through the $View[0]$ sets in the group. Views in dynamic groups could be maintained using any of several membership algorithms, e.g., random probe-based [6], gossip-based heartbeating [27], or graph-based [11].

3 Hierarchical Gossiping Protocol

3.1 Protocol Description

The protocol is similar to Flat Gossip: upon receiving a multicast, a group member M_i gossips about it for $\log_K(N)$ rounds, choosing b targets per round, where b is fixed. The difference is that, while Flat Gossip selects targets uniformly at random, Hierarchical

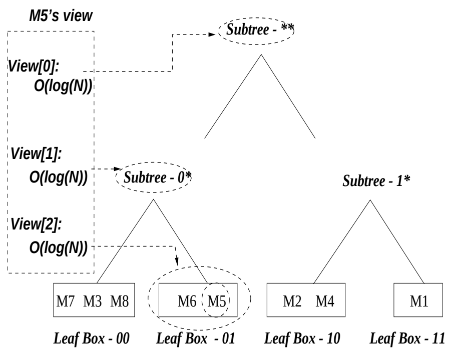


Figure 3: **Example of View Maintenance:** A typical group member (M_5 shown above) knows about ($viewfactor * log_K N$) other members in each of the subtrees it belongs to. Here $viewfactor = 1$. Group member M_5's view consists of $\log_K(N) = 3$ subviews, each containing $subviewsize(N) = viewfactor * log_K N$ members chosen uniformly at random from the subgroup of members within the appropriate same subtree as M_5. Here, $viewfactor$ is a constant. An example view at M_5 has $View[2] = \{M_6\}, View[1] = \{M_3, M_6, M_7\}, View[0] = \{M_1, M_2, M_7\}$. Note that multiple subviews at a member might overlap (e.g., M_7 belongs to $View_{M_5}[0]$ and $View_{M_5}[1]$).

Gossip prefers targets that are closer in the Leaf Box Hierarchy. As the contiguous mapping and the Leaf Box Hierarchy reflect network proximity, it results in a reduced load on core network elements.

Each gossip target at group member M_i is selected by first choosing a level j between 0 and $\log_K(N) - 1$, corresponding to M_i's height-j subtree, and then picking a member uniformly at random from $View_{M_i}[\log_K(N) - 1 - j]$, which represents M_i's knowledge of the members in this subtree. The height-0 subtree is chosen with probability $\frac{F(N,K)}{K}$, the height-1 subtree with probability $\frac{F(N,K)}{K^2}$, and so on until the height-($\log_K(N) - 1$) subtree is chosen with probability $\frac{F(N,K)}{N}$. Here, $F(N,K) = (\sum_{j=0}^{\log_K(N)-1} \frac{1}{K^{j+1}})^{-1} = \frac{(K-1)N}{N-1}$ is a normalizing constant. Figure 4 shows the (relative) target choice probability distribution for each gossip round at member M_5 in the Leaf Box Hierarchy of Figure 1.

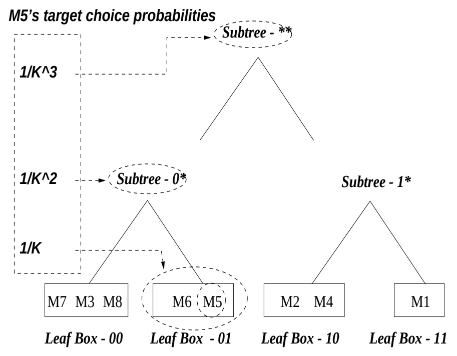


Figure 4: **Hierarchical Gossiping at a process:** M_5's gossip-target probability distribution in the example Leaf Box Hierarchy.

3.2 Protocol Analysis

We now analyze the reliability and group member load characteristics of this algorithm. Assume for simplicity that the number of group members N is a power of K and that the group members are distributed evenly among the leaf boxes by the map function H, i.e., there are exactly K members in each leaf box. These assumptions will be relaxed in the simulations.

Reliability: We model the spread of a gossip message originating at an arbitrary source using a random directed graph. The nodes of this graph denote group members. There are $b \log_K N$ arcs directed out of each node u. The endpoints of these arcs are chosen independently at random according to the following probability distribution: pick a number j between 0 and $\log_K(N) - 1$ with probability $K^{-j-1}F(N,K)$ (where $F(N,K) = \frac{(K-1)N}{N-1}$), and then pick a node v uniformly at random from among the K^{j+1} nodes in the same height-j subtree as u. The out-neighbors v of u (with arcs (u,v)) denote the nodes that u will gossip to if it receives the gossip message. The probability that the gossip reaches all group members is the probability that there is a directed path from the source to all other nodes in this random digraph.

We analyze the reliability of the Hierarchical Gossiping protocol. We first introduce some notation. Denote the source node by s. Let A_j denote the event that there is a directed path from s to all nodes in its height-j subtree using only arcs within this subtree. Let B_j denote the event that there is at least one arc directed from within this subtree to each of the other $K-1$ height-j subtrees sharing the same internal node ancestor at height $j+1$. Denote its complement by B_j^c. Let $\pi(j) = P(A_j)$, $\tilde{\pi}(j) = P(A_j | B_j)$ and $\hat{\pi}(j) = P(A_j | B_j^c)$. The probability that the gossip reaches all nodes is $\pi(\log_K(N) - 1)$. We want to show that this is close to 1 provided the number of gossip messages sent is sufficiently large.

Note that there is a directed path from s to every node in its height-$(j+1)$ subtree if there is such a path to every node in its height-j subtree, there is at least one arc crossing from this subtree to each of the $K-1$ subtrees sharing its internal node ancestor at height $j+1$, and that, starting from the endpoints of these arcs, there is a path to every node within each of these $K-1$ subtrees. Hence,

$$\pi(j+1) \geq \tilde{\pi}(j) P(B_j) \pi(j)^{K-1}. \tag{1}$$

But $\pi(j) = P(B_j)\tilde{\pi}(j) + P(B_j^c)\hat{\pi}(j)$, and so,

$$\tilde{\pi}(j) \geq P(B_j)\tilde{\pi}(j) \geq \pi(j) - P(B_j^c). \tag{2}$$

Theorem 1. *In the Hierarchical Gossiping protocol (described in Section 3.1), the constant b can be chosen sufficiently large (depending on K but not on N) that the probability of the gossip reaching all nodes is at least $1 - (3/2N)$.*

Proof: By the union bound, we have $P(B_j^c) \leq (K-1)P(\text{there is no arc from } T_j(0) \text{ to } T_j(1))$, where $T_j(0)$ is the height-j subtree containing s and $T_j(1)$ is another height-j subtree with the same height-$(j+1)$ ancestor. Now, there are $bK^{j+1} \log_K N$ arcs directed out of nodes in $T_j(0)$, each terminating in a node in $T_j(1)$ with probability $K^{-j-3}F(N,K)$, independent of the others. Hence,

$P(B_j^c) \leq (K-1)(1-K^{-j-3}F(N,K))^{bK^{j+1}\log_K N} \leq (K-1)\exp\Big(-bK^{-2}F(N,K)\frac{\log N}{\log K}\Big)$,

where we have used the inequality $(1-\frac{1}{x})^x \leq e^{-x}\ \forall\ x > 0$ to obtain the last inequality. Take $b \geq \frac{3K^2 \log K}{K-1} \geq \frac{3K^2 \log K}{F(N,K)}$. Then,

$$P(B_j^c) \leq (K-1)\exp(-3\log N) = \tfrac{K-1}{N^3} \leq \tfrac{1}{N^2},$$

since $N \geq K$. Combining this with (1) and (2), we get

$$\pi(j+1) \geq \Big(\pi(j) - \frac{1}{N^2}\Big)\Big(1-\frac{1}{N^2}\Big)\pi(j)^{K-1} \geq \pi(j)^K - \frac{2}{N^2}. \qquad (3)$$

Now $\pi(0)$ is the probability that the gossip reaches all $K-1$ nodes in the same leaf box as the source. Each of the $b\log_K N$ gossip messages originating from a node stays within the same leaf box with probability $K^{-1}F(N,K) \geq (K-1)/K$; conditional on staying within the leaf box, it is equally likely to be targeted at any of the $K-1$ other nodes. Thus, $\pi(0)$ is the probability that flat gossip in a group of size K reaches all nodes when the number of messages sent by each node is binomial with parameters $b\log_K N$ and $K^{-1}F(N,K)$. It can be shown (e.g., by adapting the arguments in [16]) that, by choosing b sufficiently large ($b = 6\log K$ will suffice), we can ensure that $\pi(0) \geq 1 - \frac{1}{N^2}$.

We now claim that, if b is chosen as above, then the inequality

$$\pi(j) \geq \pi(0)^{K^j} - \frac{2}{N^2}(1+K+\ldots+K^{j-1}) \qquad (4)$$

holds for all j between 1 and $\log_K(N)-1$. We shall show this by induction. The base case, $j=1$, is immediate from (3). Now, assuming that (4) holds for j, we have by (3) that

$$\begin{aligned}
\pi(j+1) &\geq \left[\pi(0)^{K^j} - \frac{2}{N^2}(1+K+\ldots+K^{j-1})\right]^K - \frac{2}{N^2} \\
&= \pi(0)^{K^{j+1}}\left[1 - \frac{2}{N^2}\pi(0)^{-K^j}(1+K+\ldots+K^{j-1})\right]^K - \frac{2}{N^2} \\
&\geq \pi(0)^{K^{j+1}} - \frac{2}{N^2}(K+K^2+\ldots+K^j) - \frac{2}{N^2},
\end{aligned}$$

which establishes the induction. To obtain the last inequality, we have used the fact that $(1-x)^K \geq 1-Kx$: to see this, note that $f(x) = (1-x)^K - 1 + Kx$ is convex for $K > 1$, and that it attains its minimum value of 0 at $x = 0$.

Set $j = \log_K(N) - 1$. We have $1+K+\ldots+K^{j-1} = \frac{K^j-1}{K-1} = \frac{N-K}{K(K-1)} \leq \frac{N}{2}$, since $K \geq 2$. Moreover, as b has been chosen so that $\pi(0) \geq 1 - \frac{1}{N^2}$, we have
$\pi(0)^{K^j} = \pi(0)^{N/K} \geq 1 - \frac{1}{NK} \geq 1 - \frac{1}{2N}$. Thus, by (4),

$$\pi(\log_K(N) - 1) \geq 1 - \frac{1}{2N} - \frac{2}{N^2}\frac{N}{2} = 1 - \frac{3}{2N}.$$

This completes the proof of the theorem.

Per member load: The theorem establishes that the number of gossip messages per member required to reliably transmit a single multicast message is at most $b\log_K(N)$; i.e., it grows only logarithmically in the group size, as with flat gossip. The constant in front of the logarithm is bigger, but we shall show that hierarchical gossip achieves a substantial reduction in network overhead in exchange for this small increase in per member load.

View size sufficiency: Choosing the constant $viewfactor$ to be larger than b ensures that $subviewsize(N) = viewfactor \cdot \log(N) > b\log N$. Then, each process knows about enough members (chosen uniformly at random from prospective targets) at each level of the hierarchy to implement the Hierarchical Gossiping protocol.

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 $= F(N,K) \sum_{j=h}^{\log_K(N)-1} \frac{1}{K^{j+1}} \frac{K-1}{K}$, which is bounded above by K^{-h}. 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 $bK \log_K(N)$. A symmetrical number of gossip messages passes *down* through IN_h. Thus, any internal node sees, in expectation, at most $2bK \log_K(N)$ copies of a given multicast.

Now consider a contiguous set of leaf boxes $L(D)$ assigned to a domain D in the Contiguous Mapping scheme. Say the lowest common ancestor is at height h. Denote the set of internal node ancestors of this domain, going up to the lowest common ancestor, by $IN(D)$. The set of ancestors at each height between 1 and h is contiguous (in a lexicographic ordering) and has a leftmost and rightmost member. Any message into or out of the domain has to pass through either the common ancestor at height h, or the leftmost or rightmost member of $IN(D)$ at some smaller height.[2] Thus, we can bound the number of messages crossing the domain boundary by the number of messages passing through the root or through the leftmost or rightmost node at some height between 1 and h of $IN(D)$. (Note that this relies crucially on the fact that a domain always consists

[2]For example, for Domain 1 in Figure 2, $L(D) = \{01, 10, 11\}$, while $IN(D) =$ {Subtree-0*, Subtree-1*, Subtree-**}. The claim is that any message into or out of this domain has to pass through one of these three internal nodes. In fact, in this instance, any such message has to pass through Subtree-0*. Alternatively, in the same figure, consider a domain with $L(D) = \{01, 10\}$. Then, $IN(D)$ again consists of Subtree-0*, Subtree-1* and Subtree-**. In this case, any message into or out of D has to pass through Subtree-0* or Subtree-1*.

of *contiguous* leaf boxes.) The number of such nodes is $2h - 1$. But $h \leq \log_K(N) - 1$, so the average replication factor of a multicast across *any* domain boundary is at most $2bK \log_K(N)(2 \log_K(N) - 3) \leq 4bK \log_K^2(N)$. Thus, the network overhead grows like $\log^2(N)$, whereas with flat gossip, it grows like $N \log N$ in the worst case.

4 Adaptive Dissemination Framework

The Adaptive Dissemination framework consists of two subprotocols: the *Tree Dissemination* and *Gossip* protocols. The Tree Dissemination sub-protocol uses the Leaf Box Hierarchy to dynamically construct a tree per multicast - this is done using TREE messages. Tree construction is completely decentralized, with each member using hop numbers in received TREE messages to decide, using its local partial views, which child members to forward the message to next (thus constructing the tree dynamically). When a process failure or message loss hinders further propagation of a multicast in some part of the hierarchy, a transition to the gossiping sub-protocol is automatically initiated, but is limited to the subtree(s) of the Leaf Box Hierarchy where the failure(s) occurred.

4.1 Protocol Description

Every Adaptive Dissemination message m bears exactly one multicast (batching of multiple multicasts into the same gossip message leads to a similar analysis), and the following additional information: (i) Type t of the message (t can be either TREE or GOSSIP); (ii) Hop number (hop count) h; and (iii) Ancestor list $Anc[1, a]$, if m is a TREE message. Anc is a list of identifiers of the members that the received multicast copy has passed through. We will show that this extra data in the message has size which is constant in expectation and logarithmic in the group size in the worst case.

For clarity of exposition, the description and analysis of the algorithm assumes that all process failures occur prior to the start of the dissemination. We explain later how to relax this assumption for a more general failure model, while retaining the adaptivity property. Message losses are allowed to occur during the execution of the protocol.

Tree Dissemination sub-protocol: The sender of the multicast starts by sending itself a TREE message with hop number $h = 0$. A group member M_i, on receiving a multicast through a TREE message with hop number $h < \log_K(N) - 1$, attempts to forward this message to K randomly chosen child members, one in each of the child subtrees of the internal node ancestor of M_i at height $\log_K(N) - h - 1$ in the Leaf Box Hierarchy. Member M_i cannot choose as a child any member that is already present in the ancestor list *Anc* of the TREE message it received. The hop number in the message is incremented every time it is forwarded. Every forwarded TREE message's ancestor (*Anc*) list is modified to include only those ancestors (including M_i) that lie in the subtree into which the message is being forwarded.

When a group member M_i receives a TREE message with hop number $h = \log_K(N) - 1$, this message needs to be forwarded within M_i's leaf box. This is done by using Flat Gossip within the leaf box.

Figure 5 shows an example run of this protocol. All TREE messages require to be acknowledged by the recipients. Non-receipt of an acknowledgment from a child member results in transitioning to a gossip protocol within the requisite subtree.

Transition to Gossip sub-protocol: Suppose a member M_i has received a TREE message m with hop count h. The tree dissemination protocol cannot make progress at M_i if: a) there is a child subtree of M_i's internal node ancestor at height $(\log_K(N) - h - 1)$ in which M_i does not know of any member that is not already present in *Anc*, or b) M_i does not receive an acknowledgment from some member to which it chooses to forward the TREE message. Case (a) could occur as views are not required to be complete. Case (b) could occur due to the chosen child member being faulty, or the forwarded TREE message

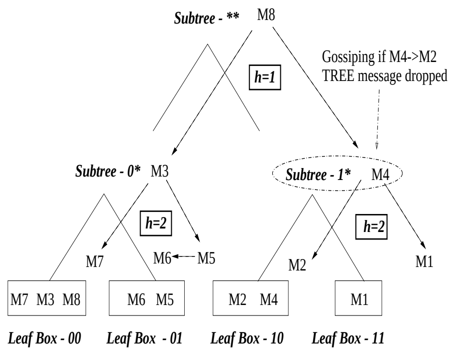


Figure 5: **Adaptive Dissemination of a multicast from member M_8 in the example from Figure 1.** The sender member (M_8) chooses two children, one from each of the child subtrees of the internal node ancestor at height $3-0-1=2$ in the Leaf Box Hierarchy (essentially the root internal node). M_8 chooses M_3 and M_4. When M_3 (respectively M_4) receives the TREE message from M_8, it notices the hop count of 1, and selects two children randomly, one from each of the child subtrees of its internal node ancestor at height $3-1-1=1$. These child subtrees happen to be leaf boxes - the members M_7 and M_5 (respectively M_2 and M_1) that receive the TREE message with hop count $=2=\log_K(N)-1$ resort to Flat Gossiping (for a fixed constant number of gossip rounds) to disseminate the multicast within their leaf boxes. If the acknowledgment from M_2 to M_4 were to be dropped, then M_4 would resort to spreading the multicast through gossip within Subtree-1* with a hop count $h=1$.

being dropped inside the network. If either (a) or (b) occurs at M_i, then M_i transitions to the gossip sub-protocol, but only within its height-($\log_K(N)-h-1$) subtree. It does so by spreading a GOSSIP message with hop count equal to h, using $View_{M_i}[h]$. (If $View_{M_i}[h]$ is empty, M_i will try $View_{M_i}[h-1]$ and so on.) A group member might gossip at multiple hop counts, when it receives GOSSIP messages with lower hop-count than it has seen yet.

Note that the algorithm is free to use any gossip primitive. We use Hierarchical Gossip in our simulations of Section 5.1. Also, the algorithm can be generalized to have each member tolerate up to f process failures or message losses (by trying up to f prospective child members for the TREE message) before transitioning to gossiping.

Figure 5 shows an example run of this protocol.

4.2 Protocol Analysis

(Per-member load): The number of Tree dissemination messages (of a given multicast) sent or received by any member is bounded by a constant, independent of group size. This is because the use of ancestor lists in Tree messages ensures that, if a member is responsible for disseminating a Tree message at some internal node, then it will not be responsible for the same message at any lower level of the tree defined by the Leaf Box Hierarchy. This implies receipt of at most 1, and transmission of at most K, Tree messages at any member. The number of acknowledgements received is also upper-bounded by K.

(Reliability): If the sender stays alive throughout the lifetime of a multicast, then it is clear that every leaf box either: a) receives at least one Tree message, or b) belongs to a subtree where gossip is initiated. Thus, the Adaptive Dissemination framework achieves at least as good reliability as the gossip primitive it uses.

(Message size): A Tree message with hop count h has at most h ancestors, and $h \leq \log_K(N) - 1$. Therefore, the ancestor list size is $(\log_K(N) - 1)$ at worst. To compute the expected list size, note that for each j between 0 and $h-1$, the j^{th} ancestor in this list was chosen uniformly at random from the tree of height $\log_K(N) - 1 - j$. Thus, the probability that it falls in the height-$(\log_K(N) - 1 - h)$ subtree that m is being forwarded to is K^{j-h}. Since the ancestor has to be included in Anc only if it falls within this subtree, the expected size of Anc is $= \sum_{j=0}^{h-1} K^{j-h} \leq \sum_{i=1}^{\infty} K^{-i} = \frac{1}{K-1}$. Thus, the expected message size is constant, irrespective of the group size.

(Network Overhead): Recall that each domain D is assigned a contiguous set of leaf boxes, $L(D)$. Say the lowest common internal node ancestor of $L(D)$ is at height h. We observed in the analysis of the Hierarchical Gossip protocol in Section 3.2 that every message crossing the domain boundary of D has to pass through one of $2h-1 \leq 2\log_K(N)-3$

internal nodes: either the lowest common ancestor, or the leftmost or rightmost internal node ancestor at some height j between 0 and $h-1$. We then used a bound on the load on each internal node to obtain a bound on the domain boundary load. We now proceed along similar lines.

A member M_i that receives a TREE message with hop count h places itself at its internal node ancestor of height $\log_K(N)-1-h$ and forwards a copy of the message to one randomly chosen node in each of the K subtrees of this internal node. The copies can be visualized as going up from M_i to the root of the tree at height $\log_K(N)-1-h$ and then down to the target nodes (there may be shorter paths to some of the target nodes, so this is a conservative estimate). Thus, each internal node sees at most $K+1$ messages (K going up and 1 going down) corresponding to every ancestor above it (including itself), up to the root of the tree. So, the number of messages passing through an internal node at height h is at most $(K+1)(\log_K(N)-h)$; for any internal node, it is at most $(K+1)(\log_K(N)-1)$. Since the number of internal nodes associated with a domain boundary is at most $2\log_K(N)-3$, the number of messages per multicast crossing a domain boundary is bounded by $2(K+1)\log_K^2(N)$ in the worst case.

Next, we compute the expected number of messages passing through an internal node IN_j at height j. Consider a TREE message initiated from IN_h, the internal node ancestor of IN_j at height h. The member M_i initiating this message is equally likely to be any of the K^{h+1} descendants of IN_h, hence it lies in the subtree rooted at IN_j with probability K^{j-h}. One copy of this TREE message is sent to a random member in each of the K subtrees rooted at IN_h; the probability that such a member lies in the subtree rooted at IN_j is $K^{j-(h-1)}$. Thus, the expected number of copies of the TREE message initiated at IN_h passing through IN_j is $= K \cdot K^{j-h} + 1 \cdot K^{j-(h-1)} = 2K^{j-h+1}$. Summing this over $h \geq j$, we find that the expected number of TREE messages passing through IN_j (for a single multicast message) is $= 2\sum_{h=j}^{\log_K(N)-1} K^{j-h+1} \leq 2K\sum_{i=0}^{\infty} K^{-i} = \frac{2K^2}{K-1} \leq 4K$. Since a domain boundary is associated with at most $2\log_K(N)-3$ internal nodes, the expected number of copies of each multicast message crossing a domain boundary is $\leq 8K\log_K(N)$.

In summary, the domain boundary load per multicast during the Tree Dissemination phase is $O(\log^2 N)$ in the worst case, and $O(\log N)$ in expectation, for any domain. Note that this result relies on using the Contiguous Mapping for assigning leaf boxes to domains.

4.3 On the Failure Model, and arbitrary values of N

We have assumed that all process failures occur prior to the start of the multicast dissemination. Relaxing this assumption to allow process failures during the protocol requires a group member M_i participating in the tree dissemination protocol to buffer a Tree message until receipt of a *second* acknowledgment from all the child members chosen for this multicast. Each member generates a second acknowledgment to its parent on either: a) receiving second acknowledgments from all its K children in the Tree Dissemination protocol, or b) initiating gossip within the requisite subtree. Non-receipt of the second acknowledgment from one of its children causes M_i to treat this as a failure of the child, and to initiate a gossip within the requisite subtree.

This modification to the protocol results in one extra message per multicast per group member and a worst case $\log_K(N)$ rounds increase in multicast dissemination latency. If the sender of the multicast stays non-faulty throughout the execution of the protocol, then the reliability analysis still holds true.

Finally, the analysis does not rely heavily on N being a factor of K, and the algorithms do not use this at all. Hence, the analysis results extend to arbitrary values of N too.

5 Experimental Results

This section first compares Hierarchical Gossip ("HierGssp") with Flat Gossip ("FlatGssp") in Section 5.1, then compares HierGssp with Adaptive Dissemination in Section 5.2. Section 5.3 studies a HierGssp variant running on transit-stub topologies.

5.1 HierGssp vs. FlatGssp: Topology-Independent Simulations

The Leaf Box Hierarchy is constructed using a branching factor of $K = 2$. We use N to denote the actual group size, which is the same as the group size estimate except in (3) below. In a dynamic group where H is reorganized rarely, the distribution of members across leaf boxes could become skewed, overloaded, or underloaded over a period of time. To study its effect, we investigate protocol behavior under the following map functions:

(1) **The default uniform mapping**: $K = 2$ members/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. $mask = 0$ is the default mapping; higher values of $mask$ are more skewed. Suffix in plots: $mask =$.

(3) **Overloading (resp. underloading) mapping**: uses the default distribution, but on a Leaf Box Hierarchy with $N/4$ (resp. N) leaf boxes, i.e. with 4 (resp. 1) members per leaf box. 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.

The mapping (1) models performance with perfect knowledge of N and the network topology, e.g., soon after a Leaf Box Hierarchy reorganization. Mappings (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 members[3]. Each member, on receiving a multicast,

[3]This is a conservative assumption since a gossip is disseminated *faster* with asynchronous gossip rounds rather than with synchronous gossip rounds [27].

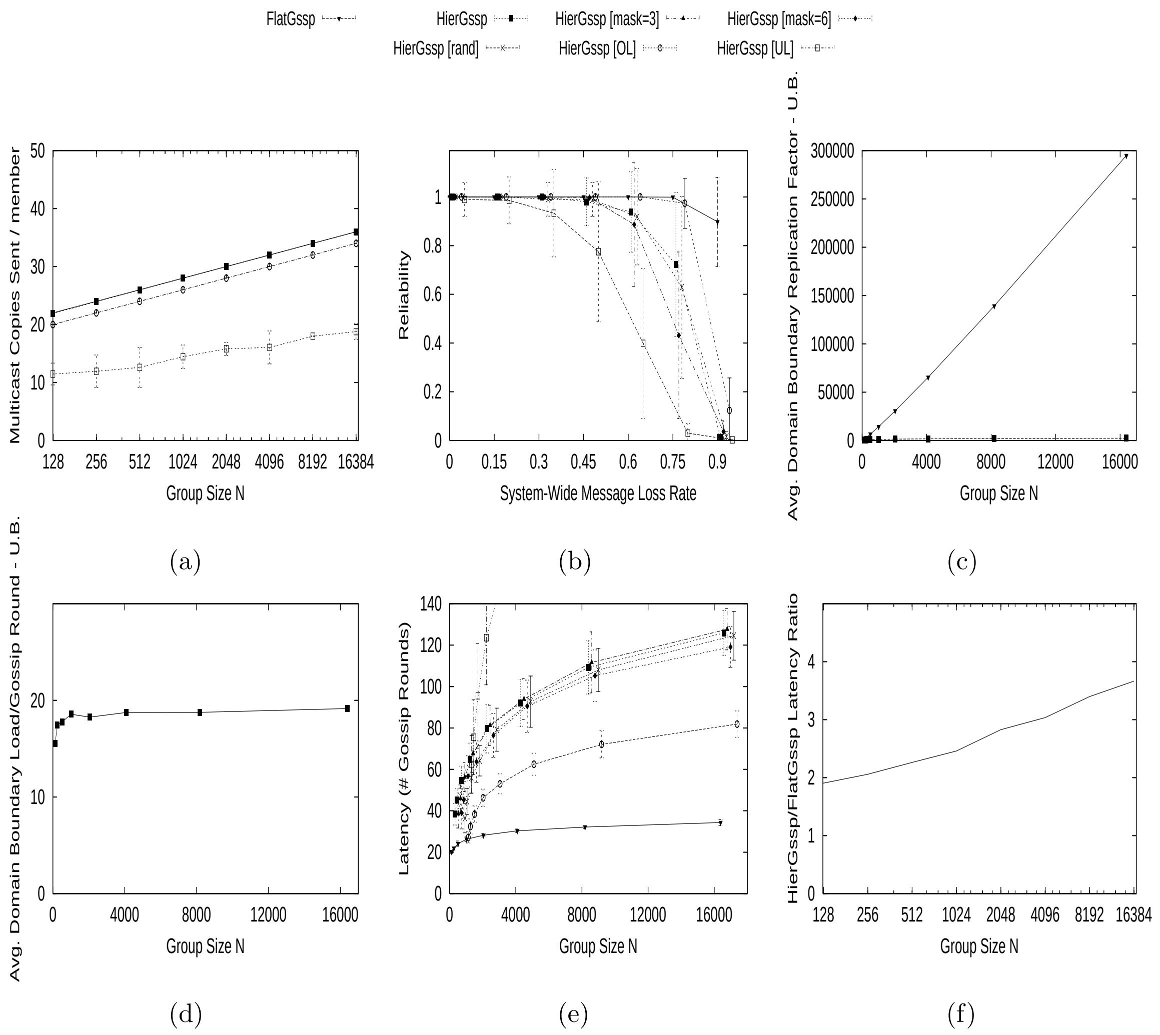


Figure 6: **Topology-Independent Simulations: Flat Gossip versus (New) Hierarchical Gossip.** In (b), $N = 2047$. In the other plots, the system-wide message loss rate=0%. Overlapping datapoints in (b) and (e) horizontally perturbed slightly for clarity. The plots are explained in Section 5.1.

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.

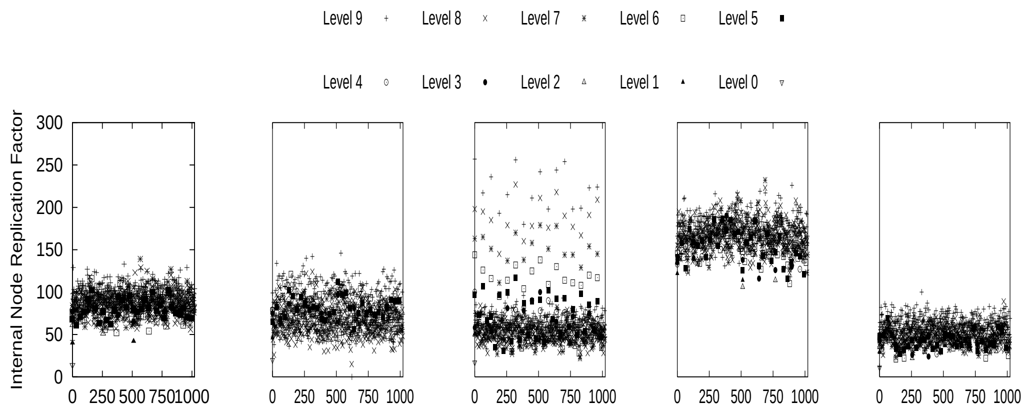
The membership view at each member M_i is constructed with $viewfactor = 4$ in the algorithm of Section 2. The view size is thus bounded by $4(\log_2(N))^2$ and increases slowly with group size. At $N = 16384$, the average measured view size is about 600, or less than 5% of group size. Each point in the plots is an average measured over 50 random runs of the protocol, with error bars showing one standard deviation.

General Scaling and Comparative Trends: The parameters are configured so that all but two variants have the same message overhead per group member per multicast. Figure 6(a) shows the measured values; all grow logarithmically. In HierGssp, the default, random and skewed mappings have the same overhead. The OL overhead is different due to fewer gossip rounds at smaller hierarchy height. UL is different due to lower reliability.

Figure 6(b) shows the reliability (fraction of members receiving a gossip message) as the system-wide message loss rate is increased. FlatGssp tolerates up to 75 % loss rate, while HierGssp (default) starts to degrade at message loss rates of 30 % due to the higher locality of gossip target selection in the latter. However, since loss rates beyond 30% indicate extreme congestion in the network, HierGssp and FlatGssp are comparable 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 degraded reliability beyond a 20% loss rate. Member failures gave similar results, and are omitted.

Figure 6(c) compares the upper bound on the average replication factor of a multicast through any domain boundary. For (default) HierGssp, this is derived from a summation of replication factors at internal nodes (Section 3.2). 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 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 in HierGssp (Figure 6(d)) it quickly converges to a limit *independent of group size.* In this experiment, the limit appears to be around 20 gossips/round.

Figure 6 (e) shows that average latency until a HierGssp epidemic dies out is independent of mapping skewedness, and increases very slowly with group size. The latency is higher than in FlatGssp, but Figure 6(f) shows that the ratio of latencies is small; it stays below 4 up to 16384 members. Both protocols have low latencies - with a 1-second gossip round, HierGssp's (resp. FlatGssp's) 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 higher latency.



(a) Default (b) $mask = 3$ (c) $mask = 6$ (d) OL(N=4096) (e) UL(N=1024)

Figure 7: **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 5.1.

The effect of staleness of the Leaf Box Hierarchy on domain boundary load is plotted in Figure 7. For a height-10 Leaf Box Hierarchy, a uniform mapping imposes similar load distributions on all internal nodes at all levels (Figure 7(a); level 0 is the root). Increased skewedness results in more messages (Figures 7(b,c)) at the internal nodes that are ancestors of large leaf boxes. Yet, the overhead is still low - even at a skew of

$mask = 6$, where the largest leaf boxes contain 65 members, the highest replication factor at an internal node is 257; this compares with a maximum replication factor of 139 in the uniform case. Figures 7(d,e) show that the OL (resp. UL) mapping, as expected, causes a doubling (resp. halving) of domain boundary load with respect to the default mapping. We conclude from these experimental results that skewedness, overloading or underloading cause only a small increase in message overhead.

We conclude 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. Staleness of the Leaf Box Hierarchy does not affect reliability or latency. Hence, even in dynamic groups, the hierarchy can be reorganized rather infrequently, e.g., when group size crosses a power of K or when the skew is high.

5.2 Hierarchical Gossip vs. Adaptive Dissemination

Figures 8 and 9 compare the new Hierarchical Gossip protocol ("HierGssp") with the Adaptive Dissemination framework that uses Hierarchical Gossiping ("Adaptive"). We use an optimized HierGssp that eliminates duplicate messages that stay within leaf boxes (in the original protocol, $\frac{1}{K}$ of gossip messages stay within the sender's leaf box).

Scalability: Figure 8 gives plots when the reliability of both Adaptive and HierGssp is 100% (i.e., all members receive all messages). Figure 8(a) shows that to achieve 100% reliability, the number of copies of each multicast seen at a member grows logarithmically with group size for HierGssp, but is an invariant with group size in Adaptive. Figure 8(b) shows that average domain boundary load per time unit (gossip round) in HierGssp converges to a constant (18 messages/gossip round) as the group size is scaled up - similar behavior is observed in Adaptive as well; however, the convergence limit is much lower (2 messages/gossip round) than in HierGssp. Figure 8(c) shows that the average latency of Adaptive is smaller than the latency in HierGssp.

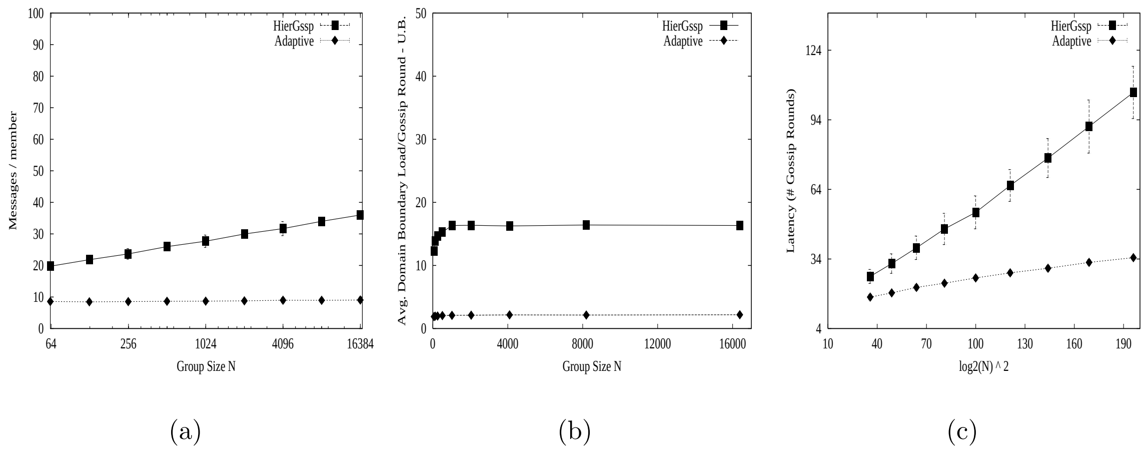
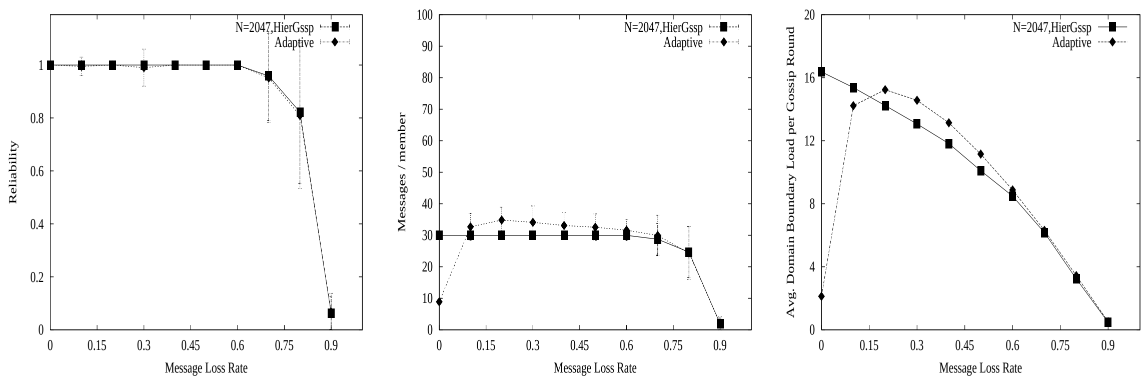


Figure 8: **Hierarchical Gossip versus Adaptive Dissemination: Group Size scaling.** Processes and network are non-faulty. See Section 5.2 for explanation of plots.

Adaptivity: Figure 9 shows the effect of increasing the messages loss rate throughout the system. Figure 9(a) shows that the Adaptive has as good reliability as HierGssp - both can tolerate up to 50 % system-wide message losses. Figures 9(b,c) show that the behavior of Adaptive is better than that of HierGssp until a message loss rate of about 15%. At higher message loss rates, Adaptive might be more expensive because a member may be chosen to participate in the adaptive gossiping at multiple levels. However, the added extra cost at message loss rates > 15% is very small. Finally, we found that the latency of Adaptive is always smaller than that of HierGssp, and that both increase by only a factor of 2 up to a message loss rate of 50%.

We conclude that the Adaptive Dissemination outperforms Hierarchical Gossiping.



(a) (b) (c)

Figure 9: **Hierarchical Gossip versus Adaptive Dissemination: Scaling and Adaptivity.** See Section 5.2 for explanation of plots.

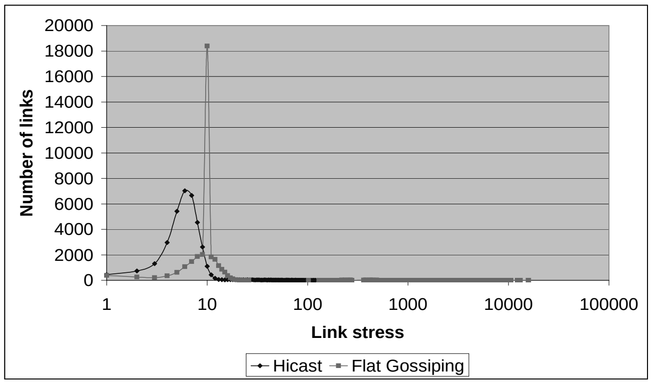
5.3 HierGssp vs. FlatGssp: Transit-Stub Topology Networks

Although protocol deployment issues are beyond our scope here, we implemented and study here our C# implementations (proprietary within MS Research) of HierGssp ("Hi-Cast") and Flat Gossiping, for GT-ITM transit-stub network topologies [29]. The parameter values used were $K = 2$, $b = 2$. Each member gossiped each message for 10 rounds in Flat Gossiping (20 rounds in HiCast) for the size 16,000 group, and for 12 rounds in Flat Gossiping (25 rounds in HiCast) in the size 64,000 group. Figure 10 shows the "stress" (replication factor for a gossip message) across the links of a 16,000 sized group on a network with 600 core routers, 60,000 LANs, 1 ms LAN links, and 40.5 ms core links. Notice that more routers have a higher load when Flat Gossiping is used, while HiCast has a lower load distributions (number and load). Figure 11 shows the variation of reliability with process failure rate and corroborates the data of Figure 6(b). Figure 11(b) is for a 64,000 group on a 5050 core node, 505,000 LAN node network.

6 Summary

This paper has presented techniques that (1) augment flat gossip-based application multicast protocols with topological awareness, and (2) augment arbitrary gossip protocols

(a)



(b)

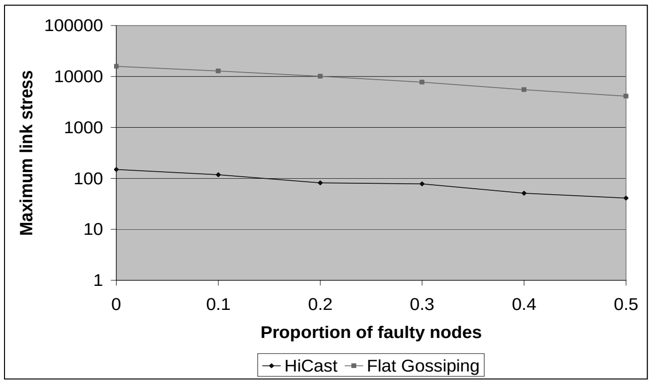


Figure 10: **Transit-stub Topologies: Comparison of network link stress in a transit-stub topology** in Hierarchical Gossiping and Flat Gossiping: (a) Distribution of link stress in a size 16,000 group; (b) Maximum link stress in the presence of failures in a size 16,000 group.

with adaptivity. We designed two new multicast protocols - Hierarchical Gossiping and Adaptive Dissemination - over a topologically-aware membership scheme called the Leaf Box Hierarchy. Analysis and experiments show that both protocols preserve reliability and scalability of gossip-based approaches. Compared to Flat Gossiping, Hierarchical Gossiping lowers network overhead by an order of magnitude, suffering only a small decrease in reliability and small increase in latency. Adaptive Dissemination has low overhead when failures are absent or few, and it automatically adjusts overhead to guarantee high reliability even as failure rates increase. The protocols are also well-suited for dynamic groups, as skewed, underloaded and overloaded hierarchies degrade performance only slightly.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. ACM SIGCOMM*, pages 205–217, 2002.

[2] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proc. IPTPS*, pages 135–140, Feb. 2003.

[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] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay muilticast architecture. In *Proc. ACM SIGCOMM*, pages 55–67, 2001.

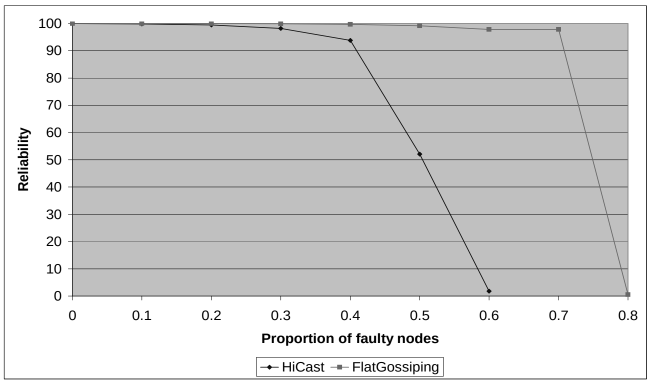
[5] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proc. ACM SIGCOMM*, Aug. 2004.

[6] A. Das, I. Gupta, and A. Motivala. SWIM: Scalable Weakly-consistent Infection-style process group Membership protocol. In *Proc. 2002 Intnl. Conf. Dependable Systems and Networks (DSN '02)*, pages 303–312, 2002.

[7] 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.

[8] P. Eugster, R. Guerraoui, S. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight Probabilistic Broadcast. In *Proc. 2001 Intnl. Conf. Dependable Systems and Networks (DSN '01)*, pages 443–452, 2001.

(a)



(b)

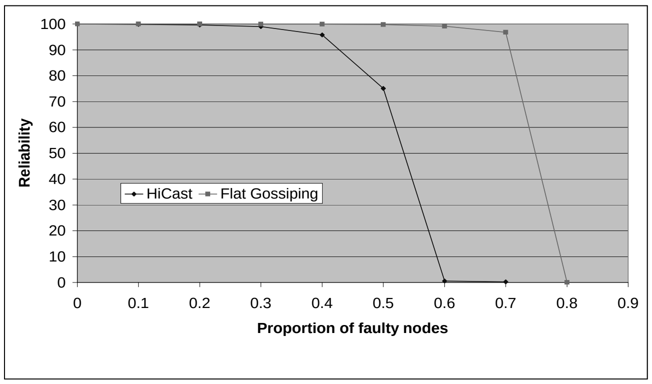


Figure 11: **Transit-stub Topologies: Comparison of resilience to process failures in a transit-stub topology** between Hierarchical Gossiping and Flat Gossiping in (a) a size 16,000 group and (b) a size 64,000 group.

[9] P. T. Eugster and R. Guerraoui. Probabilistic Multicast. In *Proc. 2002 Intnl. Conf. Dependable Systems and Networks (DSN '02)*, pages 313–322, 2002.

[10] 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.

[11] A. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Tr. on Computers*, 52(2):139–149, Feb. 2003.

[12] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *Proc. Symp. Reliable Distributed Systems (SRDS '02)*, pages 180–189, Oct. 2002.

[13] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proc. 2001 Intnl. Conf. Dependable Systems and Networks (DSN '01)*, pages 433–442, 2001.

[14] 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. Intnl.*, 18:165–184, May/Jun., 2002.

[15] A. Iamnitchi, M. Ripeanu, and I. Foster. Locating data in (small-world?) p2p scientific collaborations. In *Proc. 1st Intnl. Wshop Peer-to-Peer Systems (IPTPS '02)*, pages 85–93, Mar. 2002.

[16] A.-M. Kermarrec, L. Massoulié, and A. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Tr. Parallel and Distributed Systems*, 14(3):248–258, Mar. 2003.

[17] 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.

[18] M.-J. Lin, K. Marzullo, and S. Masini. Gossip versus Deterministically Constrained Flooding on small networks. In *Proc. 14th Intnl Conf. Distributed Computing (DISC 2000)*, pages 253–267. LNCS 1914, Springer, 2000.

[19] M. Lucas. *Efficient Data Distribution in Large-Scale Multicast Networks.* PhD thesis, U. of Virginia, May 1998.

[20] D. Malkhi and K. Horowitz. Estimating network size from local information. *ACM Information Processing Letters*, 88(5):237–243, Dec. 2003.

[21] T. Ng and H. Zhang. Towards global network positioning. In *Proc. ACM SIGCOMM Conf. on Internet Measurement*, pages 25–29, Nov. 2001.

[22] S. Paul, K. Sabnani, and S. Bhattacharya. Reliable Multicast Transport Protocol (RMTP). *IEEE Journ. Selected Areas in Communications*, 15(3):405–421, 1997.

[23] 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.

[24] A. Rowstron and P. Druschel. Pastry: scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. IFIP/ACM Middleware*, pages 329–350, 2001.

[25] 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.

[26] R. van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed systems monitoring, management, and data mining. *ACM Tr. on Computer Systems*, 21(3):164–206, 2003.

[27] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *Proc. Middleware '98*, pages 55–70. Springer, 1998.

[28] Z. Xiao and K. Birman. A randomized error recovery algorithm for reliable multicast. In *Proc. 20th IEEE INFOCOM*, volume 1, pages 239–248, 2001.

[29] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proc. 15th IEEE INFOCOM*, volume 2, pages 594–602, 1996.

Dr. Indranil Gupta is assistant professor of Computer Science at the University of Illinois at Urbana-Champaign. Indranil received his PhD from Cornell University in 2004, and his Bachelors from Indian Institute of Technology, Madras, in 1998. He is recipient of the NSF CAREER award in 2005. Indranil and his students in the Distributed Protocols Research Group (DPRG) work on design methodologies, peer-to-peer systems, the Grid, disaster recovery networks, and sensor networks. For more information, visit `http://kepler.cs.uiuc.edu.`

A. J. Ganesh graduated from the Indian Institute of Technology, Madras, in 1988. He received his MS and PhD in Electrical Engineering from Cornell University in 1991 and 1995 respectively. He is currently with the Networks and Systems group at Microsoft Research, Cambridge. His research interests include Internet resource allocation, distributed systems and mathematical modeling.

Anne-Marie Kermarrec obtained her Ph.D. from the University of Rennes in October 1996. She spent a year (Oct.96-Oct97) in the Computer Systems group of Vrije Universiteit, Amsterdam. In 1997, she became an assistant professor at the University of Rennes. In 2000, she joined Microsoft Research in Cambridge and worked in the area of peer-to-peer computing. Since February 2004, she is a senior researcher at INRIA, Rennes, France. Her research interests include large-scale distributed systems, peer to peer overlay networks and applications.