HiScamp: self-organizing hierarchical membership protocol

Ayalvadi J. Ganesh, Anne-Marie Kermarrec and Laurent Massoulié
Microsoft Research
7 J J Thomson Avenue
Cambridge CB3 0FB, UK
{ajg,annemk,lmassoul}@microsoft.com

Abstract

Gossip-based or epidemic algorithms rely on a peer-to-peer model for dissemination of multicast messages, and are simple, scalable and reliable. However, traditional gossip-based protocols suffer from two major drawbacks: (i) they rely on each peer having knowledge of the global membership and (ii) they are oblivious to the underlying network and impose a high load on core router links. In this paper we present a self-organizing hierarchical membership protocol which attemps to solve these two issues. Nodes organize themselves into clusters reflecting the topology, and obtain partial views of the membership both within and outside the cluster. The size of the partial views is tuned automatically to achieve high reliability. Gossip messages are targeted mainly within clusters, thereby reducing network load.

1 Introduction

The lack of deployment of IP multicast has led to interest in application-level multicast [11]. Centralized or partially-centralized approaches, proven efficient in local-area networks [12], do not scale to large size groups. Recently some application-layer multicast protocols [3, 16] have been deployed on top of large-scale peer-to-peer routing infrastructures. These approaches have proven to be efficient and scalable but require the existence of such a peer-to-peer infrastructure.

In this context, gossip-based protocols [1, 4, 6] have been shown to be both scalable and reliable. This class of protocols was first introduced to manage consistency in replicated databases [4, 8] but also serves other purposes such as failure detection [18], garbage collection [5] or more recently system management [17].

These protocols rely on epidemic style dissemination, and have several attractive features: *(i)* they are scalable, *(ii)* they are highly resilient to failures and *(iii)* they do not require the maintenance of global state information. Their scalability relies on a peer-to-peer interaction model and reliability on the use of redundant messages: they use randomization and redundancy to fight random unreliability (of nodes and network links) in a distributed system. In gossip-based protocols, messages are propagated as follows. When a node generates a message, it sends it to a randomly chosen subset of other nodes. When any node receives a message for the first time, it does the same. The number of gossip targets is called the fan-out. Many variants of gossip-based protocols exist, which differ in the number and choice of gossip targets. Consider first the case when gossip targets are chosen uniformly at random from among all group members. It has been shown [13] that, if there are N members in the group, then the fanout required to ensure that the message reaches every group member (with high probability) is of the order of $\log(N)$. Moreover, with such a choice of fanout, the number of hops a message traverses to reach an arbitrary group member is of the order of $\log(N)$ [15]. Therefore, the load on each group member and the latency of multicast delivery increase only logarithmically with the group size. In addition, performance degrades gracefully in the presence of node failures and message losses.

Two main factors limit the applicability of gossip in large-scale, wide-area settings. First, traditional gossip protocols rely on each node employing full knowledge of the group membership in choosing gossip targets. This has motivated work on distributed algorithms [6, 14] to provide each node with a partial view of the membership that is adequate to achieve high reliability. However, even if each node has a partial view, some coordination is required to set and update the fanout value as the size of the system changes. Second, existing gossip protocols are oblivious to the underlying network topology and hence impose a high load on core routers in wide-area systems.

Recently hierarchical epidemic protocols have been proposed to limit the load on core routers in the networks [9, 17]. In this paper, we present HiScamp, a hierarchical self-organizing membership protocol for gossip-based dissemination which addresses these issues. HiScamp uses a distance measure to dynamically cluster the nodes in a hierarchy. At each level of the hierarchy, it implements an instance of Scamp [7], a peer-to-peer membership protocol which automatically tunes the size of partial views to the logarithm of the system size. By targeting most gossip messages within clusters, the load on core routers is reduced without compromising reliability. We evaluate HiScamp on a simulator using the Georgia Tech [19] transit-stub model. We compare HiScamp (hierarchical gossip) to Scamp (flat gossip) in terms of link stress and reliability. Preliminary results show a great reduction in network load at the cost of a small decrease in reliability and small increase in latency.

2 Fanout and reliability in gossip-based protocols

We begin by briefly reviewing the relation between fanout and reliability for gossip-based protocols. These results are used to parametrize the membership schemes so that they provide partial views that are large enough to ensure high reliability.

2.1 Flat gossip

Let N denote the number of nodes in the system and suppose each node receiving a message gossips it to K other nodes, chosen uniformly at random among all the nodes. It was shown in [13] that, if the fanout K is $\log(N)+b$, for an arbitrary constant b, then the probability that a gossip message reaches all nodes goes to $\exp(-\exp(-b))$ as N grows large.[1] Note that this refers, not to the probability that a given node receives the message, but to the probability that *every node* receives it. Traditionally, atomic multicast verifies the “all or nothing” property, whereas the notion of atomicity in this paper refers to the “all” property. This result implies that there is a sharp threshold at $log(N)$; the probability that a gossip message reaches all nodes is close to 1 if each node gossips to slightly more than $\log(N)$ other nodes, and close to 0 if it gossips to slightly fewer than $\log(N)$ other nodes. Table 1 below shows the relationship between b and the quantity $1-e^{-e-b}$, which is the probability of failing to reach at least one node. By choosing the design parameter b appropriately, we can provide a suitable probabilistic reliability guarantee. The result can be extended to account for node and link failures [13].

2.2 Hierarchical gossip

In this model, nodes are grouped into clusters, and each node can gossip to other nodes either within its cluster or in other clusters. The question is how to choose the number of gossip targets within the cluster for each node (intra-cluster fanout, denoted k), and the total number of gossip targets outside the cluster aggregated over all nodes in the cluster (inter-cluster fanout, denoted f). In order for gossip to succeed in this model, i.e., for a gossip message to reach all nodes, it is sufficient that the message goes from its originating cluster to every other cluster and that gossip also succeeds within each cluster. Let there be M distinct clusters with n nodes in each cluster and let $N = Mn$ denote the total number of nodes. We want to choose the intra-cluster fanout k and the inter-cluster fanout f in order to guarantee bounds on the probability of success. It was shown in [13] that, if $f = \log(M) - \log(\beta/2)$ and $k = \log(N) - \log(\beta/2)$ for some parameter $\beta > 0$, then the probability that a gossip message reaches all nodes is at least $e^{-\beta}$. Thus, it is sufficient to take the inter-cluster fanout to be logarithmic in the number of clusters, and the intra-cluster fanout to be logarithmic in the total number of nodes (not just the nodes within that cluster). These estimates can be modified to account for node and link failures as in the case of flat gossip, and can also be extended to a multi-level hierarchy of clusters.

3 Hierarchical membership protocol

The objective is to develop a fully decentralized and self-organising membership protocol which ensures that each group member has a partial view of the membership of a size adequate to support gossip reliably (i.e., with reliability comparable to traditional schemes employing full knowledge of group membership). These partial views should be created in a peer-to-peer fashion without the use of servers, and their size should adapt automatically to changes in the size of the group as members join and leave. We described in earlier work [7] how these objectives can be met in order to support flat gossip. We proposed a self-organizing membership protocol called Scamp, which provides each node with a partial view of the membership. The size of this view scales automatically as $c\log(N)$, the multiple c being a design parameter. The partial view is used to propagate gossip messages, and is of the right size to ensure high reliability. The choice of c depends on the degree of resilience to failures required; atomic multicast can be achieved with high probability if the proportion of failed nodes or links is strictly smaller than $(c-1)/c$. The partial views generated by Scamp do not take locality into account. We wish to augment the basic protocol to support cluster-based gossip, as described above. When nodes are organized into clus-

[1] The same result applies if the fanout is itself a random number, with mean $E[K] = \log(N) + b$, under some mild additional conditions on the distribution of K.

b	-2	0	2	4	6	8	10
$1-e^{-e-b}$	0.999	0.632	0.127	0.018	0.002	3E-4	5E-5

Table 1. Dependence on b of probability of non-atomic multicast

ters using topological information, we need to provide each node with two partial views, one of nodes within the same cluster, and another of nodes in other clusters. For clarity of exposition, we describe a protocol with just two levels in the hierarchy. The scheme can be extended in an obvious way to handle hierarchies with more than two levels. We begin by describing the basic Scamp protocol and then show how it can be modified to maintain a hierarchy of views.

3.1 Protocol overview of Scamp

Scamp is a scalable membership protocol which operates in a fully decentralized manner and provides each group member with a partial view of the group membership. The size of the partial views naturally converges to the value required to support a gossip algorithm reliably. The protocol consists of mechanisms for nodes to subscribe (join) and unsubscribe (leave) from the group, and for nodes to detect and recover from isolation. The partial views at nodes evolve in response to changing group membership in a fully decentralised way.

The subscription algorithm proceeds as follows:

1. **Contact** New nodes join the group by sending a subscription request to an arbitrary pre-existing member, called a *contact*. They initialize their partial view with the nodeId of the contact.

2. **New subscription** When a node receives a new subscription request, it forwards the new nodeId to all members of its own partial view. It also forwards $c-1$ additional copies of the new nodeId (c is a design parameter that determines the proportion of failures tolerated) to randomly chosen nodes in its partial view.

3. **Forwarded subscription** When a node receives a forwarded subscription, it integrates the new subscriber in its partial view with a probability p which depends on the size of its view. If it doesn't keep the new subscriber, it forwards the subscription to a node randomly chosen from its local view. These forwarded subscriptions may be kept by the neighbours or forwarded, but are not destroyed until some node keeps them.

4. **Keeping a subscription** Each node maintains two lists, a *PartialView* of nodes it sends gossip messages to, and an *InView* of nodes that it receives gossip messages from, namely nodes that contain its nodeId in their partial views. If a node i decides to keep the subscription of node j, it places the id of node j in its partial view. It also sends a message to node j telling it to keep the nodeId of i in its *InView*.

Observe that this subscription protocol only requires local information available at the node treating the subscription request. It is shown in [7] that the system configures itself towards views of size $c \log N$ on average, where N is the total number of nodes in the system. Performance evaluation has shown that multicasting done on top of Scamp exhibits a similar degree of reliability as traditional gossip-based schemes which requires each member to maintain the list of all group members.

The unsubscription mechanism works as follows. Assume the unsubscribing node, say node n_0, has ordered the id's in its partial view as $i(1),\ldots,i(\ell)$, and the id's in its *InView* as $j(1),\ldots,j(\ell')$. The unsubscribing node will then inform nodes $j(1), j(2),\ldots,j(\ell'-c-1)$ to replace its id with $i(1), i(2),\ldots i(\ell'-c-1)$ respectively (wrapping around if $\ell'-c-1 > \ell$). It will simply inform nodes $j(\ell'-c),\ldots,j(\ell')$ to remove it from their list but without replacing it by any node id. This protocol remains local and only the unsubscribing node and its direct neighbours in the graph are involved in the unsubscription process. It is shown in [7] that this unsubscription mechanism preserves the scaling relation between view and system sizes.

3.2 HiScamp protocol description

We now describe how to organise nodes into clusters to reflect the network topology, and how to modify the membership protocol above to maintain hierarchically structured partial views of the membership. These partial views can be used to support the hierarchical version of gossip described above.

Let $D(a,b)$ denote some agreed measure of distance between nodes a and b, which each of them can assess. For instance, it could be the round-trip time between them on the Internet, measured using ping, or the number of hops on the path between them. Alternatively, it could be some measure of the cost of this path, such as the available bandwidth, which could be measured using *pathchar* [10] or packet-pair [2]. HiScamp dynamically builds a hierarchy of clusters of nodes based on $D(\cdot,\cdot)$, using Scamp at each level in the hierarchy. We illustrate the protocol using a 2-level hierarchy for simplicity.

Each node maintains two partial views, organized hierarchically. The first, called *hview*, has 2 levels (or as many levels as there are in the hierarchy): level 1 specifies the nodes to gossip to within the cluster, and level 2 specifies gossip targets in other clusters. The second partial view, called *iview*, has 1 level (or one fewer than the number of levels in the hierarchy) and specifies the inter-cluster view of the cluster to which this node belongs. In other words, the *iview* of a node is the union of the level 2 *hviews* of all the nodes in its cluster. Thus, the *hview* is specific to a node, while the *iview* is common to all nodes in the same cluster. The *iview* is not used directly for gossiping but only for handling subscriptions. We now describe how the nodes are grouped into clusters, and how these views are created and maintained.

The subscription algorithm works as follows.

1. A node j joins the system by sending a subscription request to the node s which is closest to it[2] under the distance $D(j,\cdot)$. To the extent that the actual topology of the group permits efficient clustering, it may be adequate to choose any nearby node rather than the closest, as this would still ensure that the new node joins the right cluster.

2. If the distance $D(j,s)$ is smaller than a preset threshold t, then node s includes j in the cluster to which it belongs. In this case, the subscription is processed using Scamp within this cluster. Thus, as in Scamp, the subscription of j is forwarded to all nodes that s would gossip to at the lowest level, namely the nodes in level 1 of the *hview* of s. In addition, $|iview(s)|+c-1$ copies of the subscription are also forwarded to randomly chosen nodes in this view, where $|iview(s)|$ denotes the cardinality of the *iview* of s. Each node receiving a forwarded subscription either keeps it, or forwards it to a node in its level 1 *hview*. The decision whether to keep or forward a subscription is made just as in Scamp, but based on the size of the level 1 *hview* of the node. Finally, the views of j are initialized as follows. Its level 1 *hview* consists just of s, its level 2 *hview* is empty, and its *iview* is initialized to be the same as the *iview* of s. The last is achieved by having s send a message to j with its *iview*.

3. If $D(j,s)$ exceeds the threshold t, then node j initiates a new cluster, and its subscription is handled at level 2 in the hierarchy. To do this, node s uses its *iview* to initiate Scamp. Thus, it forwards the subscription to nodes in other clusters as specified by its *iview*, as well as forwarding $c-1$ additional copies of the subscription to random elements of its *iview*. The *hview* plays no role. Forwarded inter-cluster subscriptions are treated just like in Scamp, and are either kept or forwarded to other clusters using only the *iview*'s at each step. The decision of whether to keep or forward a subscription is based on the size of the *iview* only. When a node keeps such a subscription, it includes the node-id of the new subscriber in its level 2 *hview* as well as its *iview*. This change is gossiped to all members of its own cluster, so that they can update their *iview*'s accordingly.[3] Finally, the level 1 *hview* of j is initialized to be empty, while its level 2 *hview* and its *iview* are both initialized to $\{s\}$.

4. In addition to the partial views, each node maintains a hierarchically organised *Inview* consisting of all nodes that target it for gossip. The level 1 *Inview* of node j consists of all nodes i in the same cluster that contain node j in their level 1 *hview*, and its level 2 *Inview* consists of all nodes i in other clusters that contain node j in their level 2 *hview*.

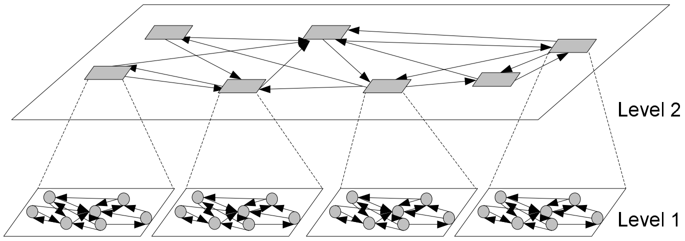

Figure 1. HiScamp overview: each level-2 rectangle represents a cluster in which an instance of the Scamp protocol is implemented; in addition, an instance of the Scamp protocol is implemented at level 2 to connect clusters.

We show in the next section that the sizes of *iview*'s converge automatically to $c\log M$, where M is the number of clusters. Likewise, the sizes of level 1 *hview*'s converge to $c\log N$, where $N = Mn$ is the total number of nodes, and n is the average number of nodes per cluster.

Figure 1 illustrates a two-levels HiScamp protocol. At the lower level, level 1, each cluster implements an instance of the Scamp protocol to connect nodes within a cluster. Each of this cluster is seen as an abstract individual node at the upper level. An instance of the Scamp protocol is used to connect clusters between them at level 2.

In the protocol as described, inter-cluster links only connect the nodes that have initiated each cluster. Such nodes thus represent a single point of failure per cluster, both for

[2] We assume that a mechanism for providing the Id of the closest node is available; in fact, this is currently an open problem.

[3] This is done so that all nodes in the same cluster always have the same *iview*. Since *iview* updates are rare, the amortized cost of synchronizing *iview*'s is not large.

messages to reach the cluster, or to be sent outside the cluster. In order to avoid this, we use an algorithm in the background which balances the level 2 *hview*'s so as to ensure that inbound or outbound messages are not handled by a unique node. Therefore, the inter-cluster links are handled by different nodes within a cluster.

This is achieved as follows. Any node whose level 2 *hview* contains more than one element periodically attempts to hand over one of its external links to another member of its cluster. To do this, it removes a node from its level 2 *hview* and forwards the nodeId to a random element of its level 1 *hview* (this action is taken only if the level 1 *hview* is non-empty). The node receiving this message treats it as a forwarded subscription; it keeps it with a probability depending on the size of its level 2 *hview* and forwards it if it is not kept. Since the number of inter-cluster links is usually small ($\log M$ aggregated over all nodes within a cluster), the level 2 *hview* of a node will typically be empty or consist of just one node, except in very small clusters. A similar hand-off mechanism is used periodicially by any node whose level 2 *Inview* consists of more than one element.

The unsubscription mechanism is exactly analogous to Scamp. Unsubscribing nodes ensure that nodeIds at each level of their partial views are transferred to nodes in the corresponding level of their *Inview*.

It is possible that nodes leave a group without implementing the unsubscription mechanism described. A lease mechanism to time out subscriptions and force resubscription was proposed in [7] to deal with this problem. The same mechanism can be implemented in HiScamp.

4 Performance evaluation

This section presents preliminary results of simulation experiments to evaluate the performance and properties of HiScamp as compare to Scamp. We evaluate and compare the hierarchical and flat protocols according to the following metrics: *(i)* the impact on the network in terms of link stress; *(ii)* resilience to node failures; and *(iv)* the latency of message delivery.

4.1 Experimental setting

We evaluated HiScamp through simulations using network topologies generated randomly according to the Georgia Tech [19] transit-stub model. The topology used in this paper is composed of a 600 node core, with a LAN attached to each core node. Each LAN has a star topology and is composed of 100 nodes on average. Thus, there are 60000 LAN nodes; we selected 50000 of these at random to compose our group. Link delays are modeled simply by assigning a propagation delay (1ms to each LAN link and 40.5ms to each core link). We evaluate the impact of HiScamp by measuring the stress on each link of the simulated network, *i.e.* the number of messages travelling along that link. This metric provides a good approximation of the load imposed on the network.

4.2 Impact on the network

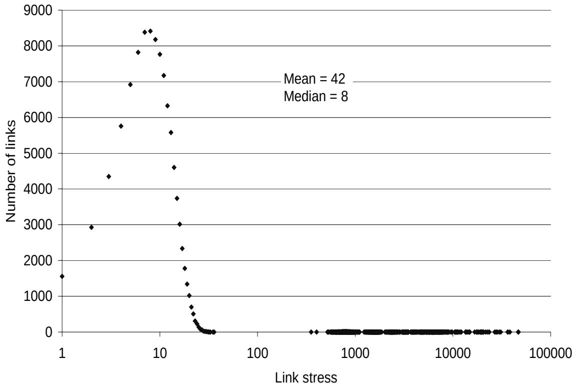

Figure 2. Distribution of link stress in a 50000 node group using Scamp

The main impact expected from HiScamp is to decrease the load on core router links.

In order to evaluate this, we compare the load on each physical link, called the link stress, in Scamp and HiScamp. In our experiments, we broadcast one message, and the link stress is defined as the number of copies of this message that traverse the link. We evaluate the link stress in a non-faulty environment.

Figures 2, 3 and 4 display the distribution of link stress on core router links for the dissemination of one multicast message in an overlay network constructed using HiScamp with 1, 2 and 3 levels respectively. The 1-level configuration is equivalent to a flat Scamp which doesn't take network topology into account. As expected, the average link stress is approximatively $log(N)$ and the maximum link stress is very high since nodes choose gossip targets uniformly, regardless of their physical location. In the 2-level configuration, we set the distance threshold t to 3, which means that nodes within the same LAN (delay 2) belong to the same cluster. In the 3-level configuration, we set distance thresholds at $t_1 = 3$ and $t_2 = 150$. These thresholds were fixed by observing the distribution of delays in typical 10000 node configurations. Detection and dynamic setting of thresholds is an interesting issue that we defer at this point of our study.

The maximum link stress decreases from 46738 [4] in the

[4] This means that almost every node gossips remotly, using a core router link.

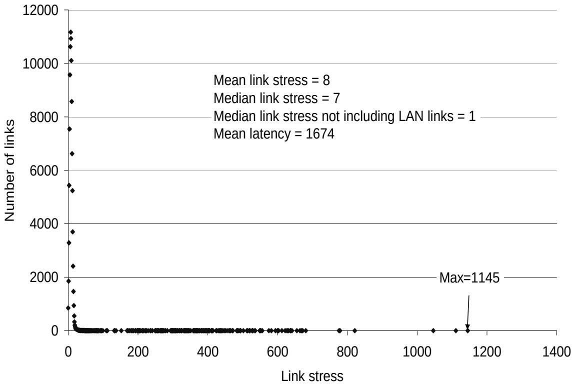

Figure 3. Distribution of link stress in a 50000 node group using a two-levels HiScamp

flat setting to 1145 with 2 levels and to 521 with 3 levels. In both configurations, the medians (with and without including the LAN links) indicate that the network traffic mostly occurs within LANs. In the 2-level configuration, 952 clusters were built dynamically whith 18 nodes per cluster on average. In the 3-level configuration, 488 clusters were built at the lowest level and 107 at the top level. In other words, there were about 4 level-2 clusters on average in each level-3 cluster. The results show that clustering significantly reduces the stress on core router links. The main impact of increasing the number of levels in the hierarchy is to significantly reduce the maximum load on a few core router links. In both HiScamp configurations, if we consider 99% of the links, the maximum link stress is 18 (instead of 1145 and 521 if we consider all links).

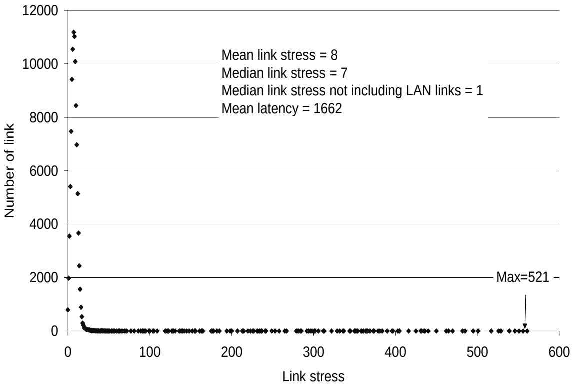

Figure 4. Distribution of link stress in a 50000 node group using a three-levels HiScamp

4.3 Reliability

As mentioned earlier, the performance of gossip-based protocols degrades gracefully in the presence of failures. Another important metric to evaluate HiScamp against is the reliability it provides in comparison to a flat implementation of Scamp. We evaluate the impact of failures, ranging from 10 to 30% of group members, on the proportion of nodes reached by a broadcast message. We consider here a fail-stop model where faulty nodes do not gossip messages they receive. The table below shows reliability figures, expressed as the fraction of non-faulty nodes reached by a multicast message, against the number of node failures. Results show that Hi-Scamp has a lower resilience to failure than a flat implementation. We are currently working on this issue.

% of faulty nodes	0	10	20	30
% of nodes reached in Scamp	100	99.76	99.4	98.77
% of nodes reached in HiScamp (2-levels)	100	97.39	93.64	88.99

Table 2. Comparison of reliability in HiScamp and Scamp in a 50000 node system

4.4 Latency

We also measured the average latency of delivery of one multicast message from the source to every group member: group members experience an average delay of 914 with Scamp, of 1674 with HiScamp with 2 levels and 1662 with HiScamp with 3 levels. So, increasing the number of levels in the hierarchy has an impact on latency but achieves a substantial reduction in the maximum link stress.

5 Conclusion

In this paper, we presented HiScamp, a peer-to-peer hierarchical membership protocol for gossip-based group communication. HiScamp is fully decentralized and leverages the self-organizing properties of Scamp, a scalable membership protocol. Experiments using the Georgia Tech topology model show that, by exploiting locality, HiScamp achieves a much smaller link stress than Scamp, but at the expense of slightly increased latency and degraded reliability. This effect is amplified as the number of hierarchical levels in HiScamp is increased.

The network stress/reliability trade-off needs further investigation, as does the automated selection of thresholds and number of hierarchical levels for cluster creation.

Acknowledgments

We would like to thank Miguel Castro and Antony Rowstron for their substantial role in the development of the simulator.

References

[1] K.P. Birman, M. Hayden, O.Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM TOCS*, 17(2):41–88, May 1999.

[2] J.-C. Bolot. End-to-end packet delay and loss behavior in the internet. In *Proceedings of SIGCOMM*, 1993.

[3] M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure, To appear. IEEE Journal on Selected Areas in communications (JSAC).

[4] A.J. Demers, D.H. Greene, C. Hauser, W. Irish, and J. Larson. Epidemic algorithms for replicated database maintenance. In *PODC*, pages 1–12, Vancouver, Canada, August 1987.

[5] K. Guo et al. Gsgc: an efficient gossip-based garbage collection scheme for scalable reliable multicast. Technical Report TR-97-1656, Cornell University, Department of Computer Science, 1997.

[6] P.T. Eugster, R. Guerraoui, S.B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *IEEE Intl. Conf. Dependable Systems and Networks (DSN)*, 2001.

[7] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Scamp: Peer-to-peer lightweight membership service for large-scale group communication. In *Networked Group Communication Workshop (NGC)*, volume 2233 of *Lecture Notes in Computer Science (LNCS)*, pages 44–56, London, UK, Nov 2001.

[8] R. Golding and K. Taylor. Group membership in the epidemic style. Technical Report UCSC-CRL-92-13, UC Santa Cruz, Dept. of Computer Science, 1992.

[9] I. Gupta, A.-M. Kermarrec, and A.J. Ganesh. Adaptive and efficient epidemic-style protocols for reliable and scalable multicast. In *Submitted to publication, http://research.microsoft.com/camdis/gossip.htm*, 2002.

[10] V. Jacobson. Pathchar – a tool to infer characteristics of internet paths. April 1997.

[11] J. Jannotti, D.K. Gifford, K.L. Johnson, M.F. Kaashoek, and J.W.O'Toole. Overcast: Reliable multicasting with an overlay network. In *Fourth Symposium on Operating Systems Design and Implementation (OSDI 2000)*, Paradise Point Resort, San Diego, California, USA, October 23-25 2000.

[12] F Kaashoek, a. Tanenbaum, A.S. Hummel, and H.E. Bal. An efficient reliable broadcast protocol. *Operating System Review*, 23, 1989.

[13] A.-M Kermarrec, L. Massoulié, and A.J. Ganesh. Probabilistic reliable dissemination in large-scale systems. available at http://research.microsoft.com/camdis/gossip.htm.

[14] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide-area network. Technical Report CS1999-0622, University of California, San Diego, Computer Science and Engineering, June 1999.

[15] B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47:213–223, 1987.

[16] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level multicast using content-addressable networks. In *Proceedings of the Third International Workshop on Networked Group Communication*, November 2001.

[17] R. van Renesse and K.P. Birman. Scalable management and data mining using astrolabe. In *IEEE International workshop on Peer-to-peer systems (IPTPS)*, 2002.

[18] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. In *International conference on Distributed Platforms and Open Distributed Processing ((IFIP)*, 1998.

[19] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *INFOCOM96*, 1996.