Peer-to-Peer Membership Management for Gossip-Based Protocols

Ayalvadi J. Ganesh, Anne-Marie Kermarrec, and Laurent Massoulié

Abstract—Gossip-based protocols for group communication have attractive scalability and reliability properties. The probabilistic gossip schemes studied so far typically assume that each group member has full knowledge of the global membership and chooses gossip targets uniformly at random. The requirement of global knowledge impairs their applicability to very large-scale groups. In this paper, we present SCAMP (Scalable Membership protocol), a novel peer-to-peer membership protocol which operates in a fully decentralized manner and provides each member with a partial view of the group membership. Our protocol is self-organizing in the sense that the size of partial views naturally converges to the value required to support a gossip algorithm reliably. This value is a function of the group size, but is achieved without any node knowing the group size. We propose additional mechanisms to achieve balanced view sizes even with highly unbalanced subscription patterns. We present the design, theoretical analysis, and a detailed evaluation of the basic protocol and its refinements. Simulation results show that the reliability guarantees provided by SCAMP are comparable to previous schemes based on global knowledge. The scale of the experiments attests to the scalability of the protocol.

Index Terms—Scalability, reliability, peer-to-peer, gossip-based probabilistic multicast, membership, group communication, random graphs.

1 Introduction

The expansion of Internet-wide distributed applications is driving the need for scalable mechanisms for reliable group communication [3], [22]. Network-level reliable multicast protocols such as SRM [13] or RMTP [18] rely on IP multicast [8], [9], which is not currently widely deployed. This motivates the need for application-level multicast protocols, which are at present an active research topic [5], [27], [6].

Probabilistic gossip-based dissemination protocols have recently emerged as an attractive alternative and provide good scalability and reliability properties [4], [19], [24]. In these protocols, each member is in charge of forwarding each message to a set of other, randomly chosen, group members. This proactive use of redundant messages provides a mechanism for ensuring reliability in the face of node crashes and high packet loss rates in the network. It can also be shown that the load on each node increases only logarithmically with the size of the group, so these algorithms are scalable. Gossip-based protocols are particularly well adapted to scenarios in which the group membership is fairly static, but the availability of group members is intermittent. Since these protocols tolerate high failure rates, no reconfiguration mechanism is required in such scenarios.

Though these approaches have proven scalable for message dissemination, they rely on a nonscalable membership protocol:[1] They assume that the subset of nodes that a node gossips to is chosen uniformly among all participating nodes, requiring that each node should know every other node. This imposes high requirements on memory and synchronization, which adversely affects their scalability. This has motivated work on distributing membership management [19], [12] in order to provide each node with a partial random view of the system without any node having global knowledge of the membership. However, the size of the partial membership required at each node in order for gossip-based message propagation to succeed is related to the size of the system. Therefore, when the group grows, the size of the partial membership at each node needs to increase accordingly. In earlier work [17], we derived the fanout (number of gossip targets) required to achieve high reliability as a function of system size. When the membership management is centralized or distributed among a few servers, the number of participants is easily determined and the fanout can be adjusted to match reliability requirements. However, in a fully decentralized model, where each node operates with an incomplete view of the system, this is no longer straightforward. None of the previously proposed partial membership schemes, to our knowledge, was able to avoid the need to know the system size.

We propose a scalable probabilistic membership protocol aimed at addressing this problem. The protocol is simple, fully decentralized, and self-configuring. As the number of participating nodes changes, we show both analytically and through simulation that the size of partial views automatically adapts to the desired value. These results are

• *The authors are with Microsoft Research, 7JJ Thomson Avenue, Cambridge CB3 0FB, UK. E-mail: {ajg, annemk, lmassoul}@microsoft.com.*

Manuscript received DD Mmmm, YYYY; revised DD Mmmm, YYYY; accepted DD Mmmm, YYYY.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 117442.

1. Our understanding of scalable membership protocol should not be confused with that of [1], [16], where the aim is to provide each member of the group with an accurate and timely global view of the membership. The problem we consider is instead to achieve reliable multicast without requiring global knowledge of membership at each node.

0018-9340/03/$17.00 © 2003 IEEE

achieved for arbitrary subscription patterns, including the worst-case scenario of all subscriptions targeting the same member. Evaluation results show that gossip based on the partial views provided by our protocol is as resilient to failures as gossip based on random choice from a global membership known at each node. The proposed protocol can potentially be incorporated in existing gossip-based schemes to reduce memory and synchronization overhead due to membership management.

The remainder of the paper is organized as follows: We describe the membership protocol in Section 2, including a sketch of the theory behind it. Section 3 presents two complementary refinements to achieve balanced view sizes even with highly unbalanced subscription patterns. Detailed evaluations are presented in Section 4. Related work is described in Section 5 and we conclude in Section 6.

2 Decentralized Membership Protocol

2.1 Requirements for Supporting Gossip-Based Multicast

Gossip-based protocols use randomization to reliably broadcast messages in a group. They have been used in various contexts such as publish-subscribe systems [12], distributed databases [10], distributed failure detection [26], distributed resource location [25], and virtual synchrony [15]. They provide probabilistic guarantees of delivery which degrade gracefully in the presence of lossy links or failed nodes. When complemented with suitable higher level recovery mechanisms they can provide the basis for offering deterministic guarantees.

There are several implementations of these protocols that differ in the length of gossip rounds and the number and selection of gossip targets. For the sake of clarity, we test SCAMP on a simple gossip-based approach where each node gossips each multicast message once to a random subset of other nodes. However, the mechanisms and results presented in this paper are applicable to other implementations of a gossip-based multicast protocol.

In gossip-based protocols, messages are propagated as follows: When a node generates a message, it sends it to a random subset of other nodes. When any node receives a message for the first time, it does the same. The random choice of gossip targets provides resilience to random failures and enables decentralized operation. Reliability is achieved by introducing sufficient redundancy by making the number of gossip targets chosen by each member large enough, as a function of the group size.

The question is how large these random subsets should be in order for the message to be reliably propagated to all group members with high probability. Results from the mathematical theory of epidemics were used in [10] to relate the number of gossip targets to the fraction of group members who eventually receive the gossip message (this is equal to the probability that an arbitrary group member will receive the message). In earlier work [17], we showed the following, sharper result: If there are n nodes and each node gossips to $\log(n) + k$ other nodes on average, then the probability that everyone gets the message converges to $e^{-e^{-k}}$. Note that this refers, not to the probability that a given node receives the message, but to the probability that *every node* receives it. We call this property *strong atomicity* to distinguish it from the traditional atomicity property[2] which requires that either no node receives the message or every node does. In [17], we also derived expressions for how the success probability depends on the failure rate of nodes and links.

Traditional gossip-based protocols rely on gossip targets being chosen uniformly at random from among all group members. In the rest of the paper, we will refer to this approach as the full membership protocol. This requires each node to maintain membership information about the whole group and is undesirable in large groups or those where the membership changes frequently. In [17], we proposed a scheme whereby a set of servers maintains the global membership list and provides individual nodes with a randomized partial view, which is updated periodically. Our goal in the present work is to eliminate the need for servers and develop a fully decentralized protocol which provides each node with a partial view of the membership. The design requirements for this protocol include:

- **Scalability:** The size of the partial view maintained at each node should grow slowly with the group size.
- **Reliability:** The partial views at each node should be large enough to support gossip with reliability comparable to that of traditional schemes relying on full knowledge of group membership.
- **Decentralized operation:** The partial views should be updated as members subscribe or unsubscribe while maintaining the above properties. The updates should take place using only local information. The partial view sizes should scale automatically to the correct value as a function of the system size, even though no node knows the system size.
- **Isolation Recovery:** An important property of traditional gossip schemes is that, each time a node gossips a multicast message, it selects new gossip targets at random. Hence, while a node may occasionally miss a message, it is very unlikely that it will be left out repeatedly. In contrast, if nodes select their gossip targets from a partial view that remains unchanged for long periods, then a mechanism for recovering from isolation is needed.

In the next section, we present the basic membership algorithm, then point out some of its drawbacks and propose refinements to address these drawbacks.

2.2 Basic Membership Management Protocol

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 decentralized way.

2.2.1 Subscription

The subscription algorithm proceeds as follows:

1. **Contact:** New nodes join the group by sending a subscription request to an arbitrary member, called a

2. Traditionally, atomic multicast verifies the "all or nothing" property, our notion of atomicity in this paper refers to the "all" property.

contact. They start with a partial view consisting of just their contact.

2. **New subscription:** When a node receives a new subscription request, it forwards the new node-id to all members of its own local view. It also creates c additional copies of the new subscription (c is a design parameter that determines the proportion of failures tolerated) and forwards them to randomly chosen nodes in its local view.
3. **Forwarded subscription:** When a node receives a forwarded subscription, provided the subscription is not already present in its list, 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 neighbors 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 node-id 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 node-id of i in its *InView*.

Algorithm 1 depicts the pseudocode for a node receiving a new subscription. Algorithm 2 depicts the pseudocode for a node receiving a forwarded subscription.

Algorithm 1 Subscription management

Upon subscription of a new subscriber s on a contact node *contact*

{The subscription of s is forwarded to all the nodes of view}
for all nodes $n \in \text{PartialView}_{contact}$ **do**
  Send(n,s,forwardedSubscription);
end for
{*c additional copies of the subscription s are forwarded to random nodes of view*}
for (j=0; j < c; j++) **do**
  Choose randomly $n \in \text{PartialView}_{contact}$
  Send(n, s,forwardedSubscription);
end for

Algorithm 2 Handling a forwarded subscription

{*n receiving s adds it with the probability* $p = 1/(1 + \text{size of } \text{PartialView}_n)$}
with probability $p = 1/(1 + \text{size of } \text{PartialView}_n)$
if $s \notin \text{PartialView}_n$ **then**
  $\text{PartialView}_n = \text{PartialView}_n + \{s\}$;
else
  Choose randomly $n \in \text{PartialView}_n$
  send(n_i, s, forwardedSusbcription);
end if

Observe that this protocol only requires local information available at the node treating the subscription request. We show below that it has the following desirable properties: If new nodes join by sending a subscription request to a member chosen uniformly at random from existing members, then the system configures itself toward partial views of size $(c+1)\log(n)$ on average. Here, n is the number of nodes in the system and c is a design parameter. We noted above that the probability that a notification which reaches everyone exhibits a sharp threshold at $\log(n)$: It is close to zero for partial views smaller than $\log(n)$ on average and close to 1 for partial views larger than $\log(n)$. This result applies to a system without failures and can easily be extended to account for link and node failures [17]. For example, if links fail with probability ε independently of each other, then the threshold is at $(\log(n))/(1-\varepsilon)$. Another reason for maintaining partial views of size $(c+1)\log(n)$ for some $c > 0$ is that it enables us to choose different subsets of size $\log(n) + k$ as gossip targets for different notifications. This ensures that link and node failures are unlikely to cause a persistent partitioning of the network and enables us to use recovery mechanisms designed for traditional gossip protocols. We can thus get many of the benefits of these protocols while maintaining fairly small partial views.

It is possible in the basic algorithm that a subscription is forwarded an infinite number of times. This happens when the number of copies of a subscription request forwarded by a node (c plus the outdegree of the contact node) is larger than the group size. This can happen when the group is small (typically less than 10 for c = 0). To avoid this, we limit the number of times a node forwards the same subscription request. When a node has received the same request more than 10 times, it simply discards the thread.

Our choice of $p = 1/(1 + \text{size of } \text{PartialView}_n)$ for the probability of keeping a forwarded subscription has two motivations. First, by making this probability a decreasing function of the current partial view size, we aim to achieve more balanced view sizes at different nodes and, consequently, a distribution of view sizes that is concentrated around the mean view size. Second, assuming that the partial view sizes are all roughly of size $(c+1)\log(n)$, the number of forwarding steps before a subscription is kept is roughly $(c+1)\log(n)$ on average, with the above choice of p. It is known from the results in [21] that the random graph under consideration has a diameter proportional to $\log(n)$. We thus expect this value of p to be sufficient for a forwarded subscription to traverse a sizable part of the graph before it is kept by some node.

We now give a mean value analysis of the subscription protocol. A sharper analysis taking the impact of fluctuations around mean values into account can be found in [14].

We model the system as a random directed graph: Nodes correspond to group members and there is a directed arc (x, y) whenever y is in the partial view of x. When a new node subscribes, the action of our algorithm is to create a random number of additional arcs as follows: Suppose there are n members already in the group. If the new node subscribes to a node with out-degree d, then $d + c + 1$ arcs are added. The new node has out-degree 1, with list consisting of just the node it subscribed to. The node receiving the subscription forwards one copy of the node-id of the subscribing node to each of its neighbors and an additional c copies to randomly chosen neighbors. All forwarded subscriptions are eventually kept by some node.

Let $E[M_n]$ denote the expected number of arcs when the number of nodes has grown to n so that the average out-degree of each node is $E[M_n]/n$. Assuming that new nodes subscribe to randomly chosen members, we have

$$E[M_n] = E[M_{n-1}] + \frac{E[M_{n-1}]}{n-1} + c + 1,$$

from which we find that $E[M_n] \approx (c+1)n\log n$.

2.2.2 Unsubscriptions

We want an unsubscription mechanism that preserves the above scaling of list sizes with system size. Recall that, in addition to its partial view, which is used to send out gossip messages, each node maintains an *InView* list consisting of nodes which contain its node-id in their partial views. Assume the unsubscribing node, say node n_0, has ordered the ids in its partial view as $i(1), \ldots, i(\ell)$ and the ids 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. In the unlikely event that this mechanism requires a node to maintain multiple copies of a node-id or to maintain its own id in its partial view, we simply delete the corresponding id. Note that this protocol remains local and only the unsubscribing node and its direct neighbors in the graph are involved in the unsubscription process.

This mechanism is motivated by the following heuristic reasoning: When a node unsubscribes from a system with n nodes and M_n arcs, the number of arcs decreases by the size of its partial view, which is M_n/n on average. In addition, all but $c+1$ of the nodes that contained the unsubscribing node in their partial view replace it with an element of its partial view. Thus, the number of arcs decreases by a further $c+1$. Thus, assuming that $E[M_n] \approx (c+1)n\log(n)$, unsubscriptions yield the recursion

$$\begin{aligned} E[M_{n-1}] &= E[M_n] - \frac{E[M_n]}{n} - (c+1) \\ &\approx (c+1)(n-1)\left(\log(n) - \frac{1}{n-1}\right) \\ &\approx (c+1)(n-1)\log(n-1). \end{aligned}$$

In other words, unsubscriptions preserve the desired mean degree.

2.2.3 Recovery from Isolation

The subscription mechanism described above creates a connected graph. However, either node failures or unsubscriptions can cause the network to become disconnected. The analysis in [17] shows that the primary mechanism by which the network may become disconnected is the isolation of individual nodes.[3] A node become isolated from the graph when all nodes containing its identifier in their partial views have failed. In order to reconnect such nodes, we propose a heartbeat mechanism. Each node periodically sends heartbeat messages to the nodes in its partial view (these are not notifications and are not propagated any further). A node that hasn't received a heartbeat message in a long time knows that it is isolated and resubscribes through an arbitrary node in its partial view. In addition, the lease mechanism presented in Section 3 for graph rebalancing also helps reduce the likelihood of prolonged isolation.

3. Conditional on disconnection, the disconnection of larger subsets has vanishingly small probability as the system size, n, grows large.

3 Mechanisms for Rebalancing the Graph

We remarked above that the basic protocol creates partial views of the required size provided new subscriptions are targeted uniformly at existing members and unsubscriptions are independent of the current view size. The latter is a reasonable assumption, but the former is not; one would instead expect newcomers to contact one node among a few whose identities are publicly advertised. We would like to ensure that the protocol continues to perform well in such a scenario. We describe two mechanisms below to achieve this. These operate on different time scales and are complementary.

3.1 Indirection

The correct scaling of list lengths with system size depends critically on the fact that the node treating the subscription of new members is chosen at random uniformly among existing members. If we let a few specially designated contact nodes treat all new subscriptions, the desired scaling no longer holds; instead, the average list lengths grow faster than expected and the lists of the contact nodes grow particularly quickly. We therefore propose *indirection* mechanisms whereby the initial contact forwards the newcomer's subscription request to a node which is chosen approximately at random among all existing nodes.

The indirection mechanism consists of two parts, a forwarding rule and a stopping rule. The stopping rule determines whether a node receiving a forwarded subscription request is going to treat it (and act as the contact). If the node decides not to treat it, the forwarding rule specifies to which of its neighbors it should forward the subscription request.

The forwarding step requires node i to forward the request to a node j in its partial view with probability w_{ij}, specified as follows: Let $pred(i)$ denote the set of nodes in i's InView and $succ(i)$ the set of nodes in i's partial view. The first requirement we want to impose is that all weights are nonnegative and that, for all i,

$$\sum_{j \in succ(i)} w_{ij} = 1 \tag{1}$$

so that the probabilistic forwarding rule is indeed well defined. A second requirement is that the steady state distribution corresponding to the random walk associated with the forwarding rule is the uniform distribution on all nodes. This is the case if the weights w_{ij} satisfy the additional constraint that, for all j,

$$\sum_{i \in pred(j)} w_{ij} = 1. \tag{2}$$

A matrix of nonnegative weights w_{ij} satisfying (1) is called stochastic; if it satisfies (2) as well, it is called doubly stochastic. If a matrix is irreducible (the associated graph is connected) and doubly stochastic, then a Markov chain with this transition matrix has the uniform distribution as its unique steady state distribution.

The weights of links incident at a node i are periodically updated by node i as follows:

$$w_{out}(i) \leftarrow \sum_{j \in succ(i)} w_{ij}, \quad \forall j \in succ(i) : \; w_{ij} \leftarrow \frac{w_{ij}}{w_{out}(i)},$$

$$w_{in}(i) \leftarrow \sum_{j \in pred(i)} w_{ji}, \quad \forall j \in pred(i) : \; w_{ji} \leftarrow \frac{w_{ji}}{w_{in}(i)}.$$

After an update of the first (respectively, second) type, node i communicates the new weights w_{ij} (respectively, w_{ji}) to the nodes j in $succ(i)$ (respectively, $pred(i)$). Weight updates are carried out asynchronously by different nodes, relying only on local information.

This update algorithm is a special case of *iterative scaling* [7]. Convergence results for iterative scaling can be stated in terms of the Kullback-Leibler divergence, $D(P||Q)$; for nonnegative matrices P and Q, this is defined as

$$D(P||Q) := \sum_{i,j} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right).$$

Suppose the initial matrix of weights $W^0 = (W^0_{ij})$ is such that there is a matrix W which meets the desired constraints (1), (2) and for which $D(W||W^0)$ is finite. Then, the sequence of weight matrices generated by the above update rule will converge to the matrix W^*, which minimizes the Kullback-Leibler divergence $D(W||W^0)$ subject to the constraints (1) and (2); see [7] for details.

The stopping rule is as follows: When a new subscription is received by a node i, it associates a counter with the subscription, initialized to a value n_i proportional to the size of node i's partial view. We took the proportionality constant equal to two in the experiments. It then forwards the subscription, along with the counter, to a member j of its partial view with probability $p_{ij} = w_{ij} / \sum_{k \in succ(i)} w_{ik}$. Each node receiving a forwarded subscription decrements the counter and continues forwarding it with probabilities computed as above. When the counter is zero, the node receiving the subscription is treated as its contact in the manner described by the subscription protocol above. Our choice of initial value for the counter is motivated by [21], where it is shown that a similar random graph has a diameter proportional to $\log(n)$. The choice of the constant of proportionality is ad hoc.

The implementation of the algorithm is as follows: Periodically, each node updates its arc weights according to Algorithm 3 and communicates the new weights to the appropriate members of its partial view or *InView*. Upon receipt of a *WeightUpdate* message, each node updates the weight of the corresponding arc only. The weights are used afterward to forward a new subscription request according to the pseudocode of Algorithm 4. Finally, when a new node is integrated either in the partial view or in the InView of a node, its weight is initialized to the mean weight of the nodes already present in the view.

Even if all nodes send their subscription requests to the same node, the indirection mechanism described above ensures that the contact node is effectively randomized.

Algorithm 3 Updating arc weights

W_{ij} *on node* n_i *contains the weight associated with the* $arc(i,j)$
W_{ji} *on node* n_i *contains the weight associated with the* $arc(j,i)$
$W_{in} = \sum_{j \in InView_{n_i}} W_{ij}$;
$W_{out} = \sum_{j \in PartialView_{n_i}} W_{ji}$;
{Update weight associated with incoming arcs}
for all $n_j \in$ InView **do**
  $W_{ji} = \frac{W_{ji}}{W_{in}}$
  Send(n_j, W_{ji}, WeightUpdate);
end for
{Update weight associated with outgoing arcs}
for all $n_j \in$ PartialView **do**
  $W_{ij} = \frac{W_{ij}}{W_{out}}$
  Send(n_j, W_{ij}, WeightUpdate);
end for

Algorithm 4 Indirection mechanism for finding a contact node

Upon subscription(s, Counter_s, newSubscription) of a new subscriber s on a node n_i

if n_i is the initial contact **then**
  $\text{Counter}_s = 2 * Card(\text{PartialView}_{n_i})$
{Initialize the length of the walk to reach a random node}
else
  if $\text{Counter}_s \neq 0$ **then**
    {Normalize weight W_{ij} *of* $n_j \in$ PartialView*}*
    for all $n_j \in$ PartialView **do**
      $W_{ij} = \frac{W_{ij}}{W_{out}(n_i)}$
    end for
    Choose $n_j \in$ PartialView with probability W_{ij}
    Decrement Counter_s;
    Send(n_j, s, Counter_s, newSubscription);
  else
    n_i acts as the contact node and applies the basic SCAMP algorithm described in Algorithm 1
  end if
end if

3.2 Lease Mechanism

Each subscription has a finite lifetime, called its lease. This could be set either by individual nodes at the time they subscribe or could be a property of the group which is imposed on all members. When a subscription expires, every node holding it in its partial view simply removes it from the partial view. It is the responsibility of each node to resubscribe at the time that its subscription expires. Nodes resubscribe to a member chosen randomly from their partial view. Resubscriptions differ from ordinary subscriptions in that the partial view of a resubscribing node is not modified.

The lease mechanism serves two functions. Even if initial subscriptions are concentrated at a few nodes, resubscriptions will be less concentrated because they are sent to

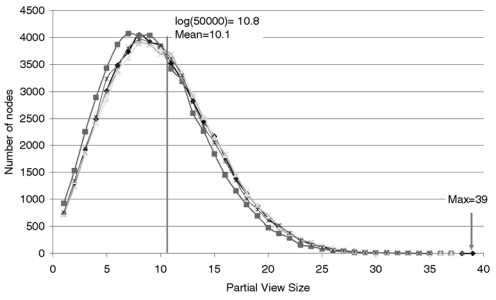


Fig. 1. SCAMP: Distribution of the partial view size in five runs of a 50,000 node group.

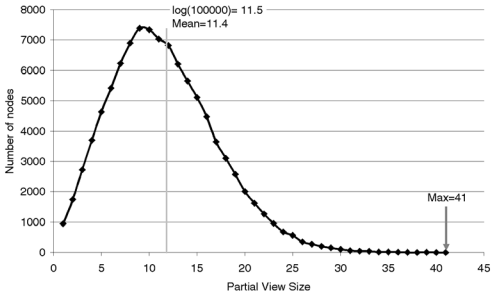


Fig. 2. SCAMP: Distribution of the partial view size in a 100,000 node group.

random members of the partial view of the resubscribing node. This helps to rebalance the size of partial views across group members. Second, it provides a mechanism for coping with nodes which either suffer crash failures or leave the group without unsubscribing using the protocol described above. Since nodes are removed from partial views after some time unless they resubscribe, such nodes will not be present in any views after some time.

Denoting again by M_n the sum of the lengths of all partial views in a system with n nodes, the impact of the lease mechanism on M_n is as follows, assuming that M_n is initially of order $(c+1)\log(n)$: When a subscription expires, the corresponding node's id is removed from all lists, which amounts to reducing M_n by $(c+1)\log(n)$ on average. The resubscription mechanism compensates this decrease in M_n by an increase of $(c+1)\log(n)+c$ on average. We thus see that this does not preserve the scaling relationship between M_n and n, but leads to an inflation in the list lengths. Although we evaluated the above lease mechanism, this problem can easily be circumvented by modifying the resubscription process to not add the c extra copies of the resubscribing node's id.

4 Simulation Results

4.1 Experimental Setting

In this section, we present detailed simulation results for SCAMP using a discrete event simulator. Our simulator implements the pseudocode presented in this paper. For each experiment, we report the mean values of results obtained through 10 runs. To evaluate the consistency of the SCAMP protocol over groups of different sizes, we ran experiments for groups with size varying from 100 to 100,000.

The goal of the simulations is to confirm the theoretical analysis regarding the size of partial membership provided to each node and to evaluate the reliability properties of gossip based on SCAMP. For the latter, we compare a gossip protocol relying on SCAMP with one relying on random choice using full membership knowledge at each node. The comparisons are made after a subscription phase and again after an unsubscription phase involving the unsubscription of half of all group members. Finally, we study the impact of the lease and indirection mechanims on the size and distribution of partial views as well as on the reliability properties.

4.2 Partial Views

In this first set of experiments, c is set to 0. Therefore, the objective of SCAMP is to achieve an average view size of $\log(n)$, where n is the size of the group. Recall that a fanout of this order is required to ensure that gossip is successful with high probability. The results show that the mean value for the partial view size is very close to $\log(n)$.

Fig. 1 depicts the distribution of the partial view size in five runs of SCAMP in a 50,000 node group. Each of the 10 runs (for clarity, only five are displayed) exhibits the same shape for the distribution, with a maximum list size of 39 and a mean value close to log(50,000). Fig. 2 displays the distribution of the partial view size in a 100,000 node group.

The figures show that the mean size of partial views achieved by SCAMP matches the target value very closely, supporting our claim that SCAMP is self-organizing. While analytical results on the success probability of gossip were derived in [17] for two specific list size distributions, namely the deterministic and binomial distributions, the success probability is in fact largely insensitive to the actual degree distribution and depends primarily on the mean degree [2]. This is corroborated by simulations.

In order to confirm that the results are also applicable to small groups, we conducted 100 simulations on a 100 node group. The mean value of partial view size obtained was 3.9 (log(100) = 4.6), which is consistent with the analysis. We don't show the distribution of view size for lack of space, but it was concentrated around the mean as in the results shown for larger groups.

4.3 Resilience to Node Failures

One of the most attractive features of gossip-based multicast is its robustness to node and link failures. It can meet stringent probabilistic reliability guarantees in the presence of failures, without any explicit recovery mechanism. This makes it particularly attractive in highly dynamic environments where members can disconnect for nonnegligible periods and then reconnect.

The goal of this set of experiments is to attest to the quality of the partial view generated by SCAMP. A key issue is that the partial views be close enough to random in an operational sense, namely, that they provide reliability comparable to using random choice based on full membership information at each node. We estimate the resilience to failure achieved by a gossip-based protocol relying on SCAMP as compared to one relying on global knowledge of membership.

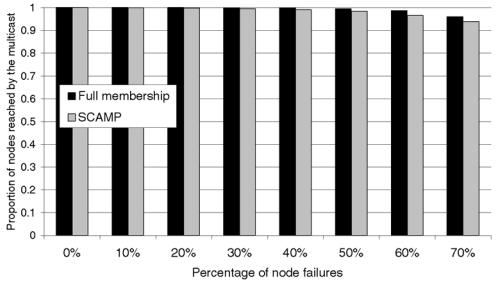


Fig. 3. Reliability of a gossip-based protocol relying on SCAMP versus one relying on a full membership knowledge in a 100,000 node group.

The results in this subsection are based on 10 runs, setting c to 0. The group is created by nodes subscribing successively, choosing a contact node uniformly at random from preexisting group members. Following the subscription phase, a message is multicast in the group. In the gossip-based protocol relying on SCAMP, each node gossips the multicast message once (when it receives it for the first time) to all nodes present in its partial view. In the protocol relying on full membership knowledge, each node gossips the multicast message, the first time it hears it, to $\log n$ other members chosen uniformly at random among all members. We use the proportion of nodes reached by a multicast in the presence of node failures as a measure of the reliability of the protocol.

Fig. 3 depicts the simulation results for a 100,000 node system as 0-70 percent of nodes fail. We plot the fraction of surviving nodes reached by a gossip message as a function of the number of failed nodes. Two observations are notable. First, the fraction of nodes reached remains very high even when close to half the nodes have failed, which confirms the remarkable fault-tolerance of gossip-based schemes. Second, this fraction is almost as high using SCAMP as using a scheme requiring global knowledge of membership. This attests to the quality of the partial views provided by SCAMP and demonstrates its viability as a membership scheme for supporting gossip.

In the experiments depicted in Fig. 3, the source node of the multicast was taken to be the first node that joined the group. This node is likely to have a larger than average partial view. Fig. 4 displays the results obtained in 10 successive runs in a 100,000 node group where the source of the multicast message was randomized. For each simulation, we show the proportion of surviving nodes reached by the multicast in the presence of 10 percent and 50 percent node failures.

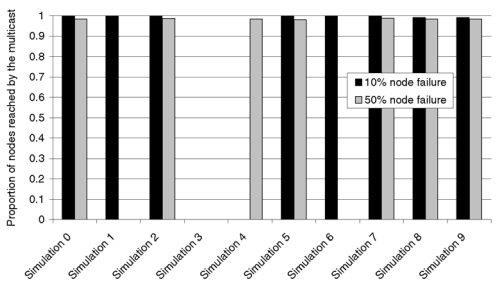


Fig. 4. Reliability in 10 successive simulations.

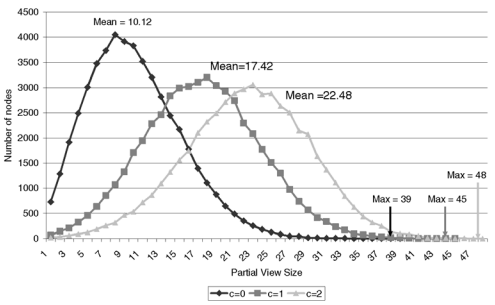


Fig. 5. Impact of c on the partial view size distribution in a 50,000 node group.

The key result here is that we observe a *bimodal* behavior: Either the reliability is as good as in a gossip-based protocol using full membership knowledge or the multicast reaches no node at all. The latter situation arises when the multicast source is disconnected from the rest of the group because the nodes present in its partial view have all failed (as in simulations 1 and 6 for 50 percent node failures, simulation 3 for all configurations or simulation 4 for 10 percent node failures). This situation can be easily detected by the source of the multicast and used to trigger a resubscription.

4.3.1 Impact of the Parameter c

As stated before, there are two different measures of reliability for a gossip-based protocol: the probability that a node receives a multicast message, which we measured in the previous set of experiments, and the probability of a strongly atomic multicast.

As shown in [17], a fanout of $\log(n)$ is not sufficient to ensure strong atomicity. To increase the mean fanout, we increase the value of the parameter c in SCAMP. Fig. 5 depicts the distribution of the partial view sizes in a 50,000 node group for different values of c.

In Fig. 6, the black bars depict the proportion of strongly atomic multicasts (in 10 runs) for different rates of node failure in a 10,000 node group. The gray bars show the proportion of nodes reached by a multicast in nonatomic runs. Results are shown for both c = 0 and c = 1. While the proportion of atomic multicasts is very low when c = 0 and more than 10 percent of the nodes have failed, it remains high for c = 1 with up to 30 percent node failures.[4] Finally, even when the multicast is not atomic, the proportion of nodes reached by the multicast message is very close to 1.

4.4 Impact of Unsubscriptions

SCAMP is targeted at dynamic environments where nodes subscribe and unsubscribe from the group. To evaluate the impact of massive unsubscriptions, we run a set of experiments where a subscription phase involving n nodes is followed by a phase where a random $n/2$ of these nodes unsubscribe. We set c = 0. Table 1 reports the impact on mean list size list as well as on the reliability guarantees. The mean list size decreases by $\log(2)$, as expected, when

4. The observation of a 0.9 probability of atomic multicast with 10 percent node failures as opposed to a probability of 1 with 20 percent node failures is an artifact caused by our running only 10 simulations.

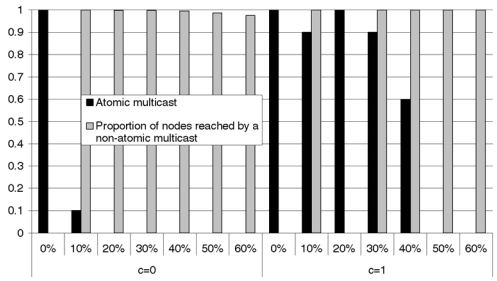


Fig. 6. Proportion of atomic multicast in a 10,000 node system with c = 0 and c = 1.

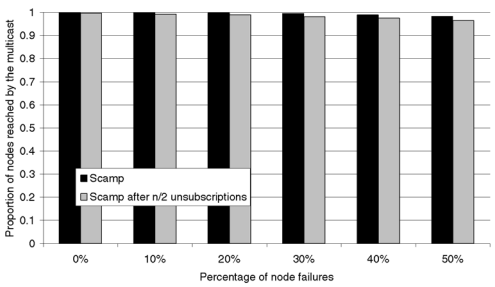


Fig. 7. Impact on reliability of n/2 unsubscriptions in a 50,000 node group.

the group size halves. Fig. 7 confirms that unsubscriptions don't significantly degrade reliability. The reason for the slight decrease in reliability is that some nodes become disconnected as a result of massive unsubscriptions. This can be detected and repaired using the proposed heartbeat mechanism. The lease mechanism also repairs disconnections as nodes resubscribe. The reliability estimates shown here thus correspond to a "worst-case" scenario immediately after massive unsubscriptions.

4.5 Impact of the Lease Mechanism

The nodes subscribing first to the group are likely to have larger partial views than the nodes subscribing last. In particular, the last node to subscribe will have only its contact node in its partial view. To balance the graph and avoid this artifact, we proposed a lease mechanism. In the experiments reported here, we use a group of 50,000 nodes, c is set to 0, and the results are averaged over 10 runs. The experiment is as follows: During the first phase, all the nodes subscribe. During the second phase, all node subscriptions expire in random order and nodes subscribe again to a random node in their partial view. Fig. 8 shows the impact of the lease mechanism on the distribution of partial view sizes. Observe that the distribution becomes more sharply concentrated around its mean value.

Fig. 9 shows the impact of the lease mechanism on reliability. The figure displays the proportion of nodes reached by a multicast as the percentage of node failures varies from 0 to 70: 1) The black bars report the results when using a gossip-based protocol relying on SCAMP without the lease mechanism and the source of the multicast is the first member of the group (Node 0); 2) the gray bars correspond to gossip using SCAMP with the lease mechanism when the source of the multicast is the first member of the group; 3) the light gray bars refer to gossip using SCAMP with the lease mechanism when the source of the multicast is a random member of the group. The results show, first, that the lease mechanim increases the probability of delivery (0.998 for 50 percent node failures) over SCAMP without lease. Second, the lease mechanism overcomes the performance penalty in multicasting from a random source illustrated in Fig. 4. The reason is that, by rebalancing views, the lease mechanism ensures that even the last nodes to join the group don't end up with very small partial views. The risk of disconnection due to failures is thus reduced.

4.6 Impact of Indirection

SCAMP relies on the assumption that a new member chooses its contact at random. In practice, new subscriptions are likely to contact one of a few well-advertised nodes. We proposed a mechanism to redirect new subscriptions to a node chosen approximately at random from the group. To test its efficacy, our experiments were carried out in an extreme scenario where all subscriptions choose the same contact, namely the first member of the group. Without a redirection mechanism, this subscription pattern has a huge impact on list sizes. For example, in a 50,000 node group, the mean size of partial views is 789.83, whereas the target value is 10.8. Moreover, nodes located close to the contact node in the graph have very large partial views. This is reflected in a large standard deviation (208.23) for the list size.

The results described below are for a 50,000 node group. Fig. 10 shows the efficiency of the indirection mechanism we propose. The black curve presents the distribution of list sizes

TABLE 1
Mean Size of Partial Views with c = 0
Before and After the Unsubscription of n/2 Nodes

n	1,000	5,000	10,000	50,000	100,000
before n/2 unsubscriptions	5.97	7.76	8.14	9.76	10.3
after unsubscriptions	5.26	7.07	7.43	9.13	9.6
Proportion of nodes reached	0.978	0.99	0.996	0.997	0.998

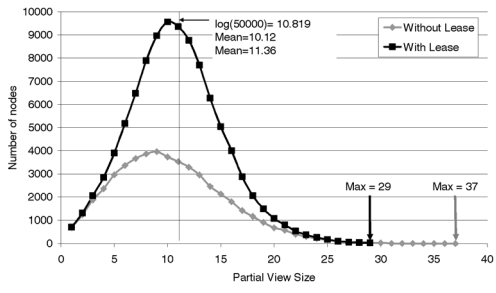


Fig. 8. Distribution of the partial views by size in a 50,000 node group with and without the lease mechanism.

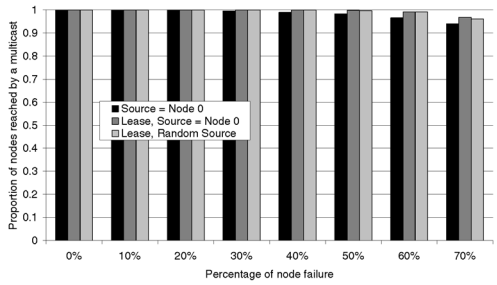


Fig. 9. Impact of the lease mechanism on the reliability guarantee in a 50,000 node group.

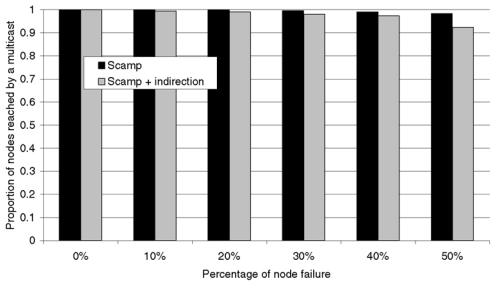


Fig. 11. Impact on reliability of the indirection mechanism in a 50,000 node group.

when all nodes choose the same node as their contact and the indirection mechanism is used to forward the subscription. The mean value of size list is 8.68. To implement this mechanism, each node periodically refreshes its weights (every 10 subscriptions in these experiments). The figure shows that the distribution of list sizes is comparable to what is obtained when nodes choose a contact uniformly at random, though the mean is slightly smaller.

Fig. 11 displays the impact of the indirection mechanism on reliability. We believe that the discrepancy between SCAMP using a random contact and SCAMP using a single contact and indirection is mainly due to the difference in the mean partial view size rather than to the contents (potential nonrandomness) of the partial views.

5 Related Work

In this section, we review different approaches to decentralized membership protocols for application-level multicast. Recently, a class of decentralized application-level protocols relying on peer-to-peer generic object location and routing substrates have emerged ([6], [27], [23]). They use the routing functionality of the peer-to-peer overlay either to build a multicast tree in a decentralized fashion (Scribe [6] and Bayeux [27]) or to build a mini-overlay [23] in which multicast messages are flooded. These approaches are more complex than the method we presented. They require the existence of a peer-to-peer overlay and they achieve reliability reactively: When failures are detected, the tree or the mini-overlay is repaired and lost messages are retransmitted. In contrast, gossip-based algorithms are easy to deploy due to their simplicity and implement a proactive reliability mechanism [15]. However, this simplicity is achieved at the expense of more traffic on the network.

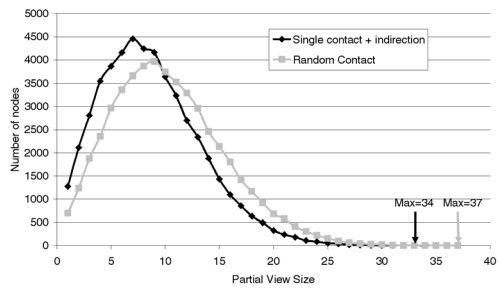


Fig. 10. Distribution of partial view sizes generated by SCAMP in a 50,000 node system with a random contact, compared with a single contact and the indirection mechanism.

While gossip protocols are scalable in terms of the communication load imposed on each process, they usually rely on a nonscalable membership algorithm. This has motivated work on distributing membership management [19], [12] in order to provide each node with a random partial view of the system, without any node having global knowledge of the membership. Another approach to this issue is presented in [20], where a connection graph called a Harary graph is constructed and messages are flooded over that graph. Optimality properties of Harary graphs ensure a good tradeoff between the number of messages propagated and the reliability guarantees. However, building such a graph requires global knowledge of membership and maintaining such a graph structure in the presence of subscriptions and unsubscriptions might prove difficult.

Directional Gossip [19] is primarily aimed at reducing the communication overhead of traditional gossip protocols. A gossip server is associated to each LAN which knows only its immediate neighbors in the wide-area network. As the gossip server gains information on the routes taken by multicast messages, it prunes links from the connection graph defined by the neighbor relation. The pruning reduces communication overhead without significantly degrading the connectivity of the graph. In our work, the emphasis is primarily on reliability under failures rather than on efficiency.

Finally, *Lpbcast* [12] is a fully decentralized membership protocol. Nodes periodically gossip a set of subscriptions they heard about during the last period to a random subset of other nodes, chosen from their partial view. A node receiving such a gossip message updates its partial view by replacing a randomly chosen node-id with a newly received one and gossips the nodeId removed from its partial view. While this mechanism achieves a good randomization of the partial views, the size of the partial view and the number of gossip targets are fixed a priori, which precludes decentralized adaptation to changes in system size.

In contrast to these approaches, SCAMP is self-organizing. It provides fully decentralized membership management with the properties to achieve gossip-based multicast with high reliability.

6 Conclusion

In this paper, we have presented the design, theoretical analysis and evaluation of SCAMP, a membership protocol for gossip-based event dissemination. SCAMP provides each member of the group with a *partial view*, that is, a list of identities of other group members. This forms the basis for broadcasting messages across the group by enabling each member to propagate messages to all or to a subset of those members whose identities are in its own list.

The mechanisms implemented in SCAMP require no centralized operation and no global knowledge needs to be maintained anywhere in the system. As nodes join and leave the group, the partial view sizes scale automatically in proportion to the logarithm of the number of members in the group, even though no group member knows the size of the group. This scaling relationship has been verified both by analysis and simulation, the latter for systems comprised of up to 100,000 members. As former analyses of standard epidemic-style gossip-based event dissemination [17], [11], [2] suggest, taking list sizes of the order of the logarithm of the system size is necessary in order for members to receive disseminated events with a high probability.

We have also verified that message multicasting done on top of SCAMP exhibits the same degree of reliability as traditional gossip-based schemes which require each member to maintain the list of all group members.

The above properties of the basic subscription and unsubscription algorithms of SCAMP depend critically on a symmetry assumption, namely that new members joining the group initially contact a member chosen uniformly at random among all existing members. This led us to complement the basic algorithm with additional mechanisms that allow us to relax this symmetry assumption. The first is an indirection mechanism for forwarding new subscription requests from an arbitrary member to one that is chosen approximately uniformly at random among all group members. The second is a lease mechanism, which helps rebalance the partial views of nodes in the system. These mechanisms enable us to maintain the good properties of SCAMP even in the extreme situation where new members always contact the same member in order to join the group.

We believe that SCAMP is a potentially useful alternative to other membership management schemes implemented in conjunction with gossip-style event dissemination as it offers the same reliability properties while relying only on decentralized operations and putting very low memory requirements on group members. Future work on SCAMP will focus on incorporating criteria such as locality, network load, and message propagation delay into the construction of the partial views maintained by users.

Acknowledgments

The authors would like to thank Miguel Castro and Antony Rowstron for their participation in the development of the simulator.

References

[1] T. Anker, G.V. Chockler, D. Dolev, and I. Keidar, "Scalable Group Membership Services for Novel Applications," *Networks in Distributing Computing (DIMACS Workshop),* M. Mavronicolas, M. Merrit, and N. Shavit, eds., pp. 23-42, Am. Math. Soc., 1998.

[2] F. Ball and A. Barbour, "Poisson Approximation for Some Epidemic Models," *J. Applied Probability,* vol. 27, pp. 479-490, 1990.

[3] K.P. Birman, "The Process Group Approach to Reliable Distributed Computing," *Comm. ACM,* vol. 36, no. 12, pp. 37-52, Dec. 1993.

[4] K.P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal Multicast," *ACM Trans. Computer Systems,* vol. 17, no. 2, pp. 41-88, May 1999.

[5] L.F. Cabrera, M.B. Jones, and M. Theimer, "Herald: Achieving a Global Event Notification Service," *Proc. HotOS VIII,* May 2001.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "SCRIBE: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE J. Selected Areas in Comm.,* to appear, 2002.

[7] I. Csiszár, "Information Theoretic Methods in Probability and Statistics," *Information Theory Soc. Rev. articles,* http://www.itsoc.org/review/frrev.html, year?

[8] S. Deering and D. Cheriton, "Multicast Routing in Datagram Internetworks and Extended LANs," *ACM Trans. Computer Systems,* vol. 8, no. 2, May 1990.

[9] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The PIM Architecture for Wide-Area Multicast Routing," *IEEE/ACM Trans. Networking,* vol. 4, no. 2, Apr. 1996.

[10] A.J. Demers, D.H. Greene, C. Hauser, W. Irish, and J. Larson, "Epidemic Algorithms for ReplicatedDatabase Maintenance," *Proc. Sixth Ann. ACM Symp. Principles of Distributed Computing (PODC),* pp. 1-12, Aug. 1987.

[11] P. Erdös and A. Renyi, "On the Evolution of Random Graphs," *Mat Kutato Int. Közl,* vol. 5, no. 17, pp. 17-60, 1960.

[12] P.T. Eugster, R. Guerraoui, S.B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov, "Lightweight Probabilistic Broadcast," *Proc. IEEE Int'l Conf. Dependable Systems and Networks (DSN2001),* 2001.

[13] S. Floyd, V. Jacobson, C.G. Liu, S. McCanne, and L. Zhang, "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing," *IEEE/ACM Trans. Networking,* pp. 784-803, Dec. 1997.

[14] A. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Scamp: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication," *Proc. Third Int'l Workshop Networked Group Comm.,* Nov. 2001.

[15] I. Gupta, K.P. Birman, and R. van Renesse, "Fighting Fire with Fire: Using Randomized Gossip to Combat Stochastic Scalability Limits," *Quality and Reliability Eng. Int'l,* vol. 18, pp. 165-184, Mar. 2002.

[16] I. Keidar, J. Sussman, K. Marzullo, and D. Dolev, "A Client-Server Oriented Algorithm for Virtually Synchronous Group Membership in WAN's," *Proc. 20th Int'l Conf. Distributed Computing Systems (ICDCS),* pp. 356-365, Apr. 2000.

[17] A.-M. Kermarrec, L. Massoulié, and A.J. Ganesh, "Probabilistic Reliable Dissemination in Large-Scale Systems," *IEEE Trans. Parallel and Distributed Systems,* to appear.

[18] J.C. Lin and S. Paul, "A Reliable Multicast Transport Protocol," *Proc. IEEE INFOCOM '96,* pp. 1414-1424, 1996.

[19] M.-J. Lin and K. Marzullo, "Directional Gossip: Gossip in a Wide-Area Network," Technical Report CS1999-0622, Univ. of California, San Diego, Computer Science and Eng., June 1999.

[20] M.-J. Lin, K. Marzullo, and S. Masini, "Gossip versus Deterministic Flooding: Low Message Overhead and High-Reliability for Broadcasting on Small Networks," *Proc. 14th Int'l Symp. Distributed Computing (DISC 2000),* pp. 253-267, Oct. 2000.

[21] B. Pittel, "On Spreading a Rumour," *SIAM J. Applied Math.,* vol. 47, pp. 213-223, 1987.

[22] D. Powell, "Group Communication," *Comm. ACM,* vol. 39, no. 4, Apr. 1996.

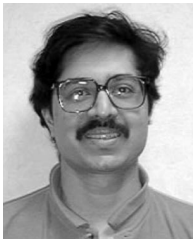
[23] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Application-Level Multicast Using Content-Addressable Networks," *Proc. Third Int'l Workshop Networked Group Comm.,* Nov. 2001.

[24] Q. Sun and D.C. Sturman, "A Gossip-Based Reliable Multicast for Large-Scale High-Throughput Applications," *Proc. Int'l Conf. Dependable Systems and Networks (DSN2000),* July 2000.

[25] R. van Renesse, "Scalable and Secure Resource Location," *Proc. IEEE Hawaii Int'l Conf. System Science,* 2000.

[26] R. van Renesse, Y. Minsky, and M. Hayden, "A Gossip-Style Failure Detection Service," *Proc. IFIP Int'l Conf. Distributed Systems and Platforms and Open Distributed Processing (Middleware '98),* 1998.

[27] S.Q. Zhuang, B.Y. Zhao, A.D. Joseph, R.H. Katz, and J. Kubiatowicz, "Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination," *Proc. 11th Int'l Workshop Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001),* June 2001.



Ayalvadi J. Ganesh graduated from the Indian Institute of Technology, Madras in 1988 and received the MS and PhD degrees in electrical engineering from Cornell University in 1991 and 1995, respectively. His research interests include queuing theory, congestion control and pricing in the Internet, and distributed systems.

Anne-Marie Kermarrec received the PhD degree in computer science from the University of Rennes, France, in 1996. She has worked as a postdoctoral researcher in the Computer Systems Group of Vrije Universiteit in Amsterdam, The Netherlands, in 1996-1997 and as an assistant professor at the University of Rennes from 1997 to 2000. Since March 2000, she has worked as a researcher at Microsoft Research, Cambridge, United Kingdom. Her research interests are in distributed systems, fault tolerance, application-level multicast, and peer-to-peer computing.

Laurent Massoulié graduated from the Ecole Polytechnique and received the PhD degree from the Université Paris Sud, Paris, France. He is currently a researcher with Microsoft Research, Cambridge, United Kingdom, where he works on modeling and performance analysis of networks. His recent research interests are in quality of service and congestion control for the Internet, and in epidemic-style information dissemination. He is currently associate editor of *Queueing Systems: Theory and Applications*.

▷ **For more information on this or any computing topic, please visit our Digital Library at** http://computer.org/publications/dlib.