The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations*

Márk Jelasity[1], Rachid Guerraoui[2],
Anne-Marie Kermarrec[3], and Maarten van Steen[4]

[1] University of Bologna, Italy, jelasity@cs.unibo.it,
and RGAI, MTA SZTE, Szeged, Hungary
[2] EPFL, Lausanne, Switzerland, Rachid.Guerraoui@epfl.ch
[3] INRIA, Rennes, France, Anne-Marie.Kermarrec@irisa.fr
[4] Vrije Universiteit, Amsterdam, The Netherlands, steen@cs.vu.nl

Abstract. In recent years, the gossip-based communication model in large-scale distributed systems has become a general paradigm with important applications which include information dissemination, aggregation, overlay topology management and synchronization. At the heart of all of these protocols lies a fundamental distributed abstraction: the *peer sampling* service. In short, the aim of this service is to provide every node with peers to exchange information with. Analytical studies reveal a high reliability and efficiency of gossip-based protocols, under the (often implicit) assumption that the peers to send gossip messages to are selected uniformly at random from the set of all nodes. In practice—instead of requiring all nodes to know all the peer nodes so that a random sample could be drawn—a scalable and efficient way to implement the peer sampling service is by constructing and maintaining *dynamic unstructured* overlays through gossiping membership information itself.

This paper presents a generic framework to implement reliable and efficient peer sampling services. The framework generalizes existing approaches and makes it easy to introduce new ones. We use this framework to explore and compare several implementations of our abstraction. Through extensive experimental analysis, we show that all of them lead to different peer sampling services none of which is uniformly random. This clearly renders traditional theoretical approaches invalid, when the underlying peer sampling service is based on a gossip-based scheme. Our observations also help explain important differences between design choices of peer sampling algorithms, and how these impact the reliability of the corresponding service.

* in *Middleware 2004*, Springer, LNCS 3231. This work was partially supported by the Future & Emerging Technologies unit of the European Commission through Project BISON (IST-2001-38923) and by the Swiss National Fond project 2100-064994.01/1.

1 Introduction

Motivation Gossip-based communication protocols have been applied successfully in large scale systems. Apart from the well-known traditional application for information dissemination [6, 9], gossiping has been applied for aggregation [16, 14, 13], load balancing [15], network management [29], and synchronization [20]. The common property of these protocols is that, periodically, every node of the distributed system exchanges information with some of its peers. The underlying service that provides each node with a list of peers is a fundamental distributed component of gossip-based protocols. This service, which we call here the *peer sampling* service is usually assumed to be implemented in such a way that any given node can exchange information with peers that are selected following a uniform random sample of *all nodes* in the system. This assumption has led to rigorously establish many desirable features of gossip-based broadcast protocols like scalability, reliability, and efficiency (see, e.g., [24] in the case of information dissemination, or [16, 14] for aggregation).

To achieve this uniform random selection, many implementors opt for the solution where every node *knows* all other nodes of the system [4, 11, 17]. Practically speaking, every node maintains a membership table, also called its *view*, the size of which grows with the size of the system. The cost of maintaining such tables has a non-negligible overhead in a dynamic system where processes join and leave at run time. In short, whereas the application and its underlying gossip-based protocol are supposed to be scalable, it is wrong to assume that this is also the case for the underlying peer sampling service.

Recently, much research has been devoted to designing scalable implementations of this service. The basic idea is to use a gossip-based dissemination of membership information naturally integrated into the service [8]. The continuous gossiping of this information enables the building of unstructured overlay networks that capture the dynamic nature of distributed peer-to-peer systems and help provide very good connectivity in the presence of failures or peer disconnections.

Interestingly, there are many variants of the basic gossip-based membership dissemination idea, and these variants mainly differ in the way new views are built after merging and truncating views of communicating peers (see, e.g., [12]). So far, however, there has never been any evaluation of and comparison between these variants, and this makes it hard for a programmer to choose the implementation of the peer sampling service that best suits the application needs. More importantly, it is not clear whether any of these variants actually lead to *uniform sampling*, which, as we pointed out, lies at the heart of all analytical studies of gossip-based protocols. In search for an answer to these questions, this paper introduces a generic protocol scheme in which known and novel gossip-based implementations of the peer sampling service can be instantiated, and presents an extensive empirical comparison of these protocols.

Contribution First, we identify a new abstract service, the peer sampling service, which is a fundamental building block underlying gossip-based protocols. This

peer sampling service is thus indispensable for gossip-based implementations of a wide range of higher level functions, which include information dissemination, aggregation, network management and synchronization.

Second, as a result of identifying this service and performing its logical separation in a class of existing applications, we present a generic protocol scheme, which generalizes the gossip-based peer sampling service protocols we are aware of. Our scheme makes it possible to implement new protocols as well.

Third, we describe an experimental methodology to evaluate the protocols in question. A key aspect of the methodology is that we focus on the *overlay network* that is induced by the peers that the service returns to nodes. In particular, we examine if these overlays exhibit *stable properties*, that is, whether the corresponding protocol instances lead to the *convergence* of important properties of the overlay. We also measure the extent to which these communication topologies deviate from the desirable uniform random model mentioned earlier. We do so by looking at several static and dynamic properties: degree distribution, average path length and clustering coefficient. We also consider the reliability of the service by examining its self-healing capacity and robustness to failure.

The behavior of the protocol instances we evaluate shows a rather wide variation. A common characteristic, however, is that no instance leads to a uniform sampling, rendering traditional theoretical approaches invalid when these protocols are applied as a sampling service. This result is surprising, as uniform randomness has long been generally assumed based only on (wrong) intuition. As a result of our work, all previous theoretical results about these protocols assuming randomness will have to be revised to properly describe the observed behavior.

Roadmap In Section 2 we define the peer sampling service. Section 3 describes our generic protocol and the various dimensions according to which it can be instantiated. Section 4 presents our experimentation methodology. Sections 5, 6 and 7 discuss our results in different simulation scenarios. In Section 8 we interpret the result of the experiments. Related work is discussed in Section 9. Finally, Section 10 concludes the paper.

2 Peer Sampling Service

The peer sampling service is interpreted over a set of nodes that form the domain of the gossip-based protocols that make use of the service. The same sampling service can be utilized by multiple gossip protocols simultaneously, provided they have a common target group. The task of the service is to provide a participating node of a gossiping application with a subset of peers from the group to send gossip messages to.

The API of the peer sampling service is extremely simple consisting of only two methods: init and getPeer. While it would be technically straightforward to provide a framework for a multiple-application interface and architecture, for a better focus and simplicity of notations we assume that there is only one application. The specification of these methods is as follows.

init() Initializes the service on a given node if this has not been done before. The actual initialization procedure is implementation dependent.

getPeer() Returns a peer address if the group contains more than one node. The returned address is a sample drawn from the group. The specification of this sample (randomness, correlation in time and with other peers) is implementation dependent (one research goal of the present paper is exactly to give information about the behavior of this method in the case of a class of gossip-based implementations).

Many times an application needs more than one peer. To maintain focus we define getPeer to return only one peer. Applications requiring more peers can call this method repeatedly. We note however that allowing getPeer to return more peers at the same time might allow for optimizations of the implementation of the service.

Note that we do not define a stop method. The reason is to ease the burden on applications by propagating the responsibility of automatically removing non-active nodes to the service layer.

The design of the service should take into account requirements with respect to the quality of peer sampling, as well as the costs involved for providing a certain quality.

Based on the growing body of theoretical work cited above, the service should ideally always return a peer as the result of independent uniform random sampling. However, we note that although this quality criterion is useful to allow rigorous analysis, it is by no means the case that all gossiping applications actually require uniform randomness. For example, some applications require only good mixing of random walks, which can also be established without demanding that peers are sampled uniformly. On the other hand, applications such as those that do aggregation do at least require that samples are not drawn from a fixed, static subset of all possible nodes.

These two examples illustrate that the costs of sampling may be reduced if near-uniformity is good enough for the application that makes use of the sampling service. In short, for an implementation of the service there is a trade-off between the required quality of sampling and the performance cost for attaining that quality. Uniform randomness can be conveniently treated as a baseline to compare protocols to, and in particular the quality of the sampling service.

3 Evaluation Framework

To study the impact on various parameters of gossip-based approaches to peer sampling, we define an evaluation framework. A wide range of protocols fits into this framework and in particular the peer sampling components of the protocols Lpbcast [8] and Newscast [12] are specific instances of protocols within this framework.

System model We consider a set of nodes connected in a network. A node has an address that is needed for sending a message to that node. Each node maintains

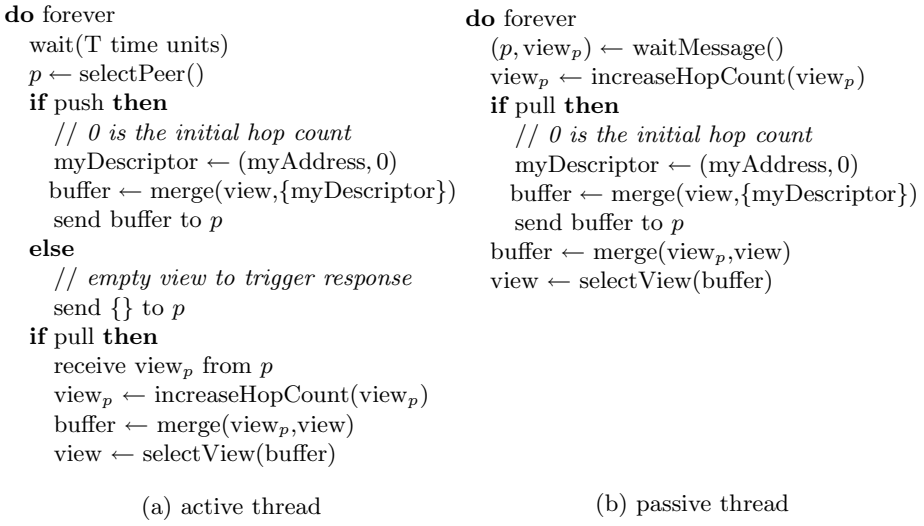


Fig. 1. The skeleton of a gossip-based implementation of a peer sampling service.

addresses by means of a *partial view*, which is a set of c *node descriptors*. The value of c is the same for all nodes. Besides an address, a node descriptor also contains a *hop count*, as we explain below.

We assume that each node executes the same protocol, of which the skeleton is shown in Figure 1. The protocol consists of two threads: an active thread initiating communication with other nodes, and a passive thread waiting for incoming messages. The skeleton code is parameterized with two Booleans (push and pull), and two function placeholders (selectPeer() and selectView()).

A view is organized as a list with at most one descriptor per node and ordered according to increasing hop count. We can thus meaningfully refer to the *first* or *last* k elements of a particular view (note however that all hop counts do not necessarily differ so the first and last k elements are not always uniquely defined by the ordering). A call to increaseHopCount(*view*) increments the hop count of every element in *view*. A call to merge($view_1$,$view_2$) returns the union of $view_1$ and $view_2$, ordered again by hop count. When there is a descriptor for the same node in each view, only the one with the lowest hop count is inserted into the merged view; the other is discarded.

This design space enables us to evaluate in a simple and rigorous way the impact of the various parameters involved in gossip-based protocols along three dimensions: *(i)* Peer selection; *(ii)* View propagation; *(iii)* View selection. Many variations exist along each of these dimensions; we limit our study to the three most relevant strategies per dimension. We shall now define these dimensions.

Peer selection Periodically, each node selects a peer to exchange membership information with. This selection is implemented by the function selectPeer() that returns the address of a *live* node as found in the caller's current view. In this study, we consider the following *peer selection* policies:

rand	Uniform randomly select an available node from the view
head	Select the first node from the view (the one with the *lowest* hop count)
tail	Select the last node from the view (the one with the *highest* hop count)

View propagation Once a peer has been chosen, the peers may exchange information in various ways. We consider the following three *view propagation* policies:

push	The node sends its view to the selected peer
pull	The node requests the view from the selected peer
pushpull	The node and selected peer exchange their respective views

View selection Once membership information has been exchanged between peers and merged as explained above, peers may need to truncate their views in order to adhere to the c items limit imposed as a protocol parameter. The function selectView(*view*) selects a subset of at most c elements from *view*. Again, we consider only three out of the many possible *view selection* policies:

rand	Uniform randomly select c elements without replacement from *view*
head	Select the first c elements from *view*
tail	Select the last c elements from *view*

These three types of policies give rise to a total of 27 combinations, each of which we express by means of a 3-tuple (ps, vs, vp) with ps indicating one of the three possible peer selection policies, vs the view selection policies, and vp the chosen view propagation policy. As an example, Lpbcast corresponds to the 3-tuple (rand,rand,push), whereas Newscast is described by (rand,head,pushpull). In the following, a DON'T CARE value (i.e., a wild card) is denoted by the symbol "*".

Implementation of the peer sampling API The implementation of method init() is done by initializing the view of the node by an arbitrary peer node. This obviously involves a bootstrapping problem, which can be solved by out-of-band methods, for example through well-known nodes or a central service publishing contact nodes, or with any other convenient method. We will experimentally evaluate different bootstrapping methods in Section 5. As the simplest possible implementation, method getPeer() can return a random sample of the current view. Obviously, more sophisticated implementations are also possible that e.g. maximize the diversity of the set of peers returned by consecutive calls to getPeer. From our point of view in this paper the only important feature is that getPeer utilizes the local partial view to return a peer.

4 Experimental methodology

As already mentioned in Section 2 the baseline of our evaluation will be the ideal independent uniform random implementation of the sampling service. It is far from trivial to compare a given sampling service to this ideal case in a meaningful way. Statistical tests for randomness and independence tend to hide the most important *structural* properties of the system as a whole. Instead of a statistical approach, in our methodology, we switch to a graph theoretical framework, which provides richer possibilities of interpretation from the point of view of reliability, robustness and application requirements, as Section 4.2 also illustrates.

To translate the problem into a graph theoretical language, we consider the *communication topology* or *overlay topology* defined by the set of nodes and their views (recall that getPeer() returns samples from the view). In this framework the directed edges of the communication graph are defined as follows. If node a stores the descriptor of node b in its view then there is a directed edge (a, b) from a to b.

In the language of graphs, the question is how similar this overlay topology is to a random graph in which the descriptors in each view represent a uniform independent random sample of the whole node set?

4.1 Targeted questions

There are two general questions we seek to answer. The first and most fundamental question is whether, for a particular protocol implementation, the communication graph has some stable properties, which it maintains during the execution of the protocol. In other words, we are interested in the *convergence behavior* of the protocols. We can expect several sorts of dynamics which include chaotic behavior, oscillations or convergence. In case of convergence the resulting state may or may not depend on the initial configuration of the system. In the case of overlay networks we prefer to have convergence toward a state that is independent of the initial configuration. Sometimes this property is called *self-organization*. In our case it is essential that in a wide range of scenarios the system should automatically produce consistent and predictable behavior.

A related question is that *if* there is convergence then what kind of communication graph does the protocol converge to? In particular, as mentioned earlier, we are interested in what sense do these graphs deviate from certain random graph models.

4.2 Selected graph properties

In order to find answers to the above problems we need to select a set of observable properties that characterize the communication graph. In the following, we will focus on the *undirected* version of the communication graph which we get by simply dropping the orientation of the edges. The reason for this choice is that even if the "knows-about" relation that defines the directed communication

graph is one-way, the actual information flow from the point of view of the applications of the overlay is potentially two-way, since after initiating a connection the passive party will learn about the active party as well. Now let us turn to the properties we will examine.

Degree distribution The degree of a node is defined as the number of its neighbors in the undirected communication graph. We will consider several aspects of the degree distribution including average degree, the dynamics of the degree of a node, and the exact degree distribution. The motivation for looking at degree distribution is threefold and includes its direct relationship with reliability to different patterns of node failures [2], its crucial effect on the exact way epidemics are spread (and therefore on the way epidemic-based broadcasting is performed) [23] and finally its key role in determining if there are communication hot spots in the overlay.

Average path length The shortest path length between node a and b is the minimal number of edges that are necessary to traverse in the graph in order to reach b from a. The average path length is the average of shortest path lengths over all pairs of nodes in the graph. The motivation of looking at this property is that, in any information dissemination scenario, the shortest path length defines a lower bound on the time and costs of reaching a peer. For scalability, small average path length is essential.

Clustering coefficient The clustering coefficient of a node a is defined as the number of edges between the neighbors of a divided by the number of all possible edges between those neighbors. Intuitively, this coefficient indicates the extent to which the neighbors of a are also neighbors of each other. The clustering coefficient of the graph is the average of the clustering coefficients of the nodes, and always lies between 0 and 1. For a complete graph, it is 1, for a tree it is 0. The motivation for analyzing this property is that a high clustering coefficient has potentially damaging effect on both information dissemination (by increasing the number of redundant messages) and also on the self-healing capacity by weakening the connection of a cluster to the rest of the graph thereby increasing the probability of partitioning. Furthermore, it provides an interesting possibility to draw parallels with research on complex networks where clustering is an important research topic (e.g., in social networks) [30].

4.3 Parameter settings

The main goal of this paper is to explore the different design choices in the protocol space described in Section 3. That is, the parameters which we want to explore are peer selection, view selection, and symmetry model. Accordingly, we chose to fix the network size to $N = 10^4$ and the maximal view size to $c = 30$.

During our preliminary experiments some parameter settings turned out not to result in meaningful overlay management protocols. In particular, (head,$*$,$*$) results in severe clustering, ($*$,tail,$*$) cannot handle dynamism (joining nodes)

protocol	partitioned runs	average number of clusters	average largest cluster
(rand,head,push)	100%	58.36	4112.09
(rand,rand,push)	33%	2.27	9572.18
(tail,head,push)	100%	38.19	7150.52
(tail,rand,push)	1%	2.00	9941.00

Table 1. Protocols where partitioning was observed in the growing overlay scenario. Data corresponds to cycle 300.

at all and (∗,∗,pull) converges to a star topology, which is highly undesirable. These variants are therefore excluded from further discussion.

5 Convergence

We now present experimental results that illustrate the convergence properties of the protocols in three different bootstrapping scenarios. The first is the case of a growing overlay discussed in Section 5.1. The second is the initialization of the overlay with a structured large diameter topology (Section 5.2) and finally the initialization with a random topology (Section 5.3).

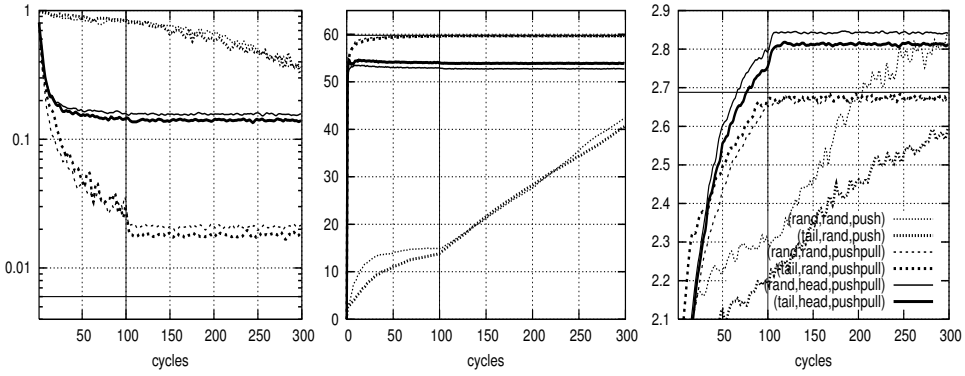
As we focus on the dynamical properties of the protocols, we did not wish to average out interesting patterns so in all cases the result of a single run is shown in the plots. Nevertheless, we ran all the scenarios 100 times to gain data on the stability of the protocols with respect to the connectivity of the overlay. Connectivity is a crucial feature, a minimal requirement for all applications. The results of these runs show that in all scenarios, every protocol under examination creates a connected overlay network in 100% of the runs. The only exceptions (shown in Table 1) were detected during the growing overlay scenario.

5.1 Growing overlay

In this scenario the overlay network initially contains only one node. At the beginning of each cycle, 100 new nodes are added to the network until the maximal size is reached in cycle 100. The view of these nodes is initialized with only a single node descriptor, which belongs to the oldest, initial node.

This scenario is the most pessimistic one for bootstrapping the overlays. It would be straightforward to improve it by using more contact nodes, which can come from a fixed list or which can be obtained using inexpensive local random walks on the existing overlay. However, in our discussion we intentionally avoid such optimizations to allow a better focus on the core protocols and their differences.

Figure 2 shows the dynamics of the properties of the communication topology. Protocols (rand,head,push) and (tail,head,push) are not plotted due to their



(a) clustering coefficient (b) average node degree (c) average path length

Fig. 2. Dynamics of graph properties in the growing scenario. Horizontal line indicates the property in a uniform random topology, vertical line indicates end of growth

instability in this scenario with respect to connectivity of the overlay (see Table 1). A non partitioned run of both (rand,rand,push) and (tail,rand,push) is included however.

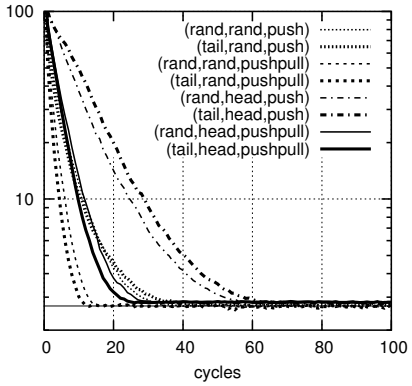
The partitioning of the push version of the protocols is due to the fact that it is only the first, central node that can distribute new links to all new members. For the same reason convergence is extremely slow when push is applied, while the pushpull versions do show fast convergence. Protocols (∗,rand,pushpull) are seemingly closer to the random topology, however, we will see that this is misleading and is a result of a highly non-balanced degree distribution (see Section 6).

5.2 Ring lattice initial topology

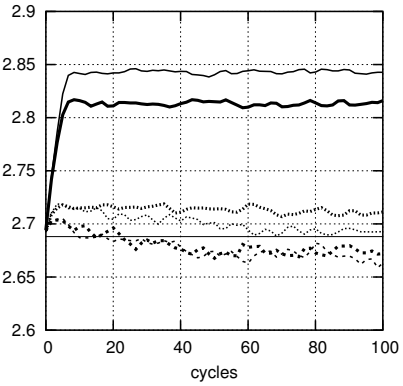
In this scenario, the initial topology of the overlay was a ring lattice, a structured topology. The motivation behind this experiment is to examine if the overlay properties converge to the same random structure with a low average path length even if the initial topology is highly structured and has a large average path length.

We build the ring lattice as follows. The nodes are first connected into a ring in which each node has a descriptor in its view that belongs to its two neighbors in the ring. Subsequently, for each node, we add additional descriptors of the nearest nodes in the ring until the view is filled.

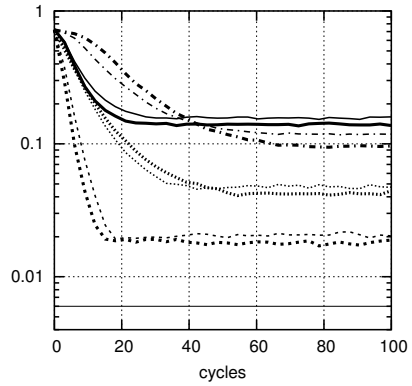
Figure 3 shows the output of this scenario as well. As in the case of the growing scenario, 300 cycles were run but here only 100 are shown to focus on the more interesting initial dynamics of the protocols. We can observe that all versions result in quick convergence which is particularly well illustrated by path length in Figure 3(a) (note the logarithmic scale), but also by the other observed properties.



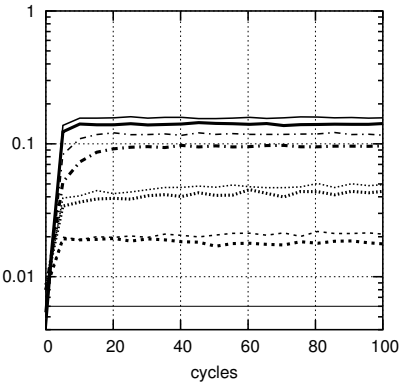
(a) lattice, average path length



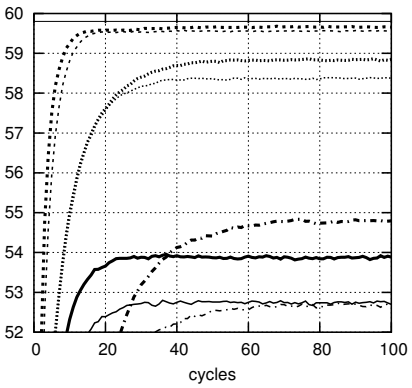
(b) random, average path length



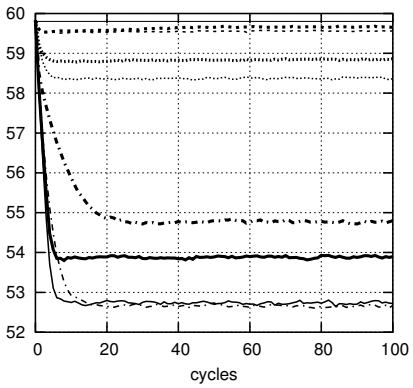
(c) lattice, clustering coefficient



(d) random, clustering coefficient



(e) lattice, average node degree



(f) random, average node degree

Fig. 3. Dynamics of graph properties. Horizontal line shows uniform random topology.

5.3 Random initial topology

In this scenario the initial topology was defined by a random graph, in which the views of the nodes were initialized by a uniform random sample of the peer nodes. Figure 3 includes the output of this scenario as well. As in the other scenarios, 300 cycles were run but only 100 are shown.

The most interesting feature we can notice is that independently of starting conditions, all properties converge to the same value. This cannot be seen in the case of path length, but it is also true. We can also see that the values are rather close to that of the random topology, maybe with the exception of the clustering coefficient. However, to put these results in the appropriate context, we need to consider the degree distribution as well. For instance, the star topology—which has a maximally unbalanced degree distribution—also has a low diameter and low clustering coefficient, while it is obviously far from random.

6 Degree distribution

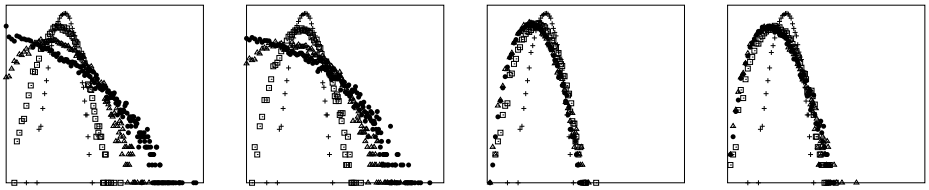
When describing degree distribution in a dynamic system one has to focus on two aspects: the dynamics of the degree of individual nodes and the dynamics of the degree distribution over the whole overlay. In principle, knowing one of these aspects will not determine the other, and both are important properties of an overlay.

The results presented in this section were obtained from the experiments performed according to the random initialization scenario described above. The evolution of the degree distribution over the whole overlay is shown in Figure 4. We can observe how the distribution reaches its final shape starting from the random topology, as the distributions that correspond to exponentially increasing time intervals (cycle 0, 3, 30 and 300) are also shown.

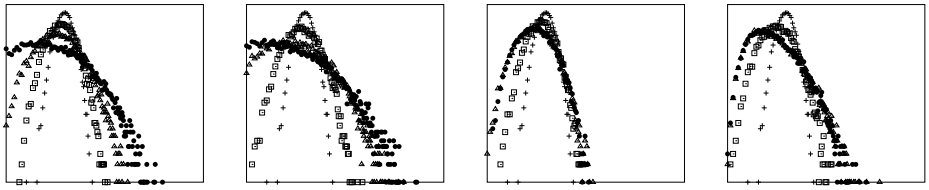
This time the behavior of the protocols can clearly be divided into two groups according to view selection. Note that previous experiments did not reveal this difference. Random view selection results in an unbalanced distribution and slow convergence while head selection is more balanced and very fast. This is a very important difference and it will be reflected in most of the following experiments as well.

Let us continue with the question whether the distribution of the degree of a fixed node over time is the same as the distribution of the converged overlay at a fixed cycle. In the overlay the degree of 50 nodes were traced during $K = 300$ cycles. Table 2 shows statistical data concerning degree distribution over time at the 50 fixed nodes and over the full overlay in the last cycle (i.e. in cycle K). The notations used are as follows. Let $d(i, j)$ denote the degree of node i in cycle j. Let $\overline{d_i}$ be the mean degree of node i over K consecutive cycles. Now, let $\overline{d} = \sum_{i=1}^{50} \overline{d_i}/50$ and $\sigma = \sum_{i=1}^{50} (\overline{d_i} - \overline{d})^2/49$, where $\overline{d}$ is the average and σ is the empirical variance of the time-averages of the degree of the traced 50 nodes. Finally, $\overline{D_K}$ is the average of node degrees in cycle K over all nodes.

We can see that in all cases the degree of all nodes oscillates around the overall average, in other words, all nodes tend to have the same degree, there



(a) (rand, rand, push) (b) (rand, rand, pushpull) (c) (rand, head, push) (d) (rand, head, pushpull)



(e) (tail, rand, push) (f) (tail, rand, pushpull) (g) (tail, head, push) (h) (tail, head, pushpull)

Fig. 4. Degree distributions on the log-log scale, when starting from a random topology. The ranges are [30,300] for the degree axis (horizontal), and [1:1000] for the frequency axis (vertical). Note that degree is guaranteed to be at least 30. The symbol + denotes the random graph (cycle 0). Empty box, empty triangle and filled circle belong to cycle 3, 30 and 300, respectively.

are no emerging higher degree nodes on the long run. On the other hand, we again observe a major distinction according to view selection. In the case of random selection the oscillation has a much higher amplitude, the network is less stable.

The last question we consider is whether the sequence of node degrees during the cycles of the protocol can be considered a random sequence drawn from the overall degree distribution. If not, then how quickly does it change, and is it perhaps periodical? To this end we present autocorrelation data of the degree time-series of fixed nodes in Figure 5. The band indicates a 99% confidence interval assuming the data is random. The autocorrelation of the series $d(i,1),\dots d(i,K)$ for a given time lag k is defined as

$$r_k = \frac{\sum_{j=1}^{K-k}(d(i,j)-\overline{d_i})(d(i,j+k)-\overline{d_i})}{\sum_{j=1}^{K}(d(i,j)-\overline{d_i})^2},$$

which expresses the correlation of pairs of degree values separated by k cycles.

protocol	$\overline{D_{300}}$	$\overline{d}$	$\sqrt{\sigma}$
(rand,head,push)	52.623	52.703	1.394
(tail,head,push)	54.785	55.519	2.690
(rand,head,pushpull)	52.717	52.933	1.756
(tail,head,pushpull)	53.916	53.888	2.176
(rand,rand,push)	58.404	60.804	19.062
(tail,rand,push)	58.844	58.746	17.287
(rand,rand,pushpull)	59.569	61.306	13.886
(tail,rand,pushpull)	59.666	58.616	9.756

Table 2. Statistics describing the dynamics of the degree of individual nodes.

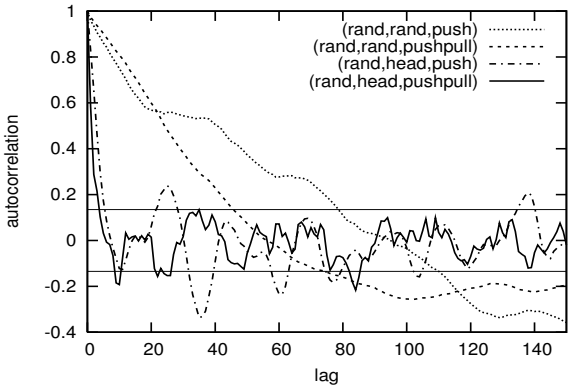


Fig. 5. Autocorrelation of the degree of a fixed random node as a function of time lag, measured in cycles, computed from a 300 cycle sample. Protocols (tail,∗,∗) are omitted for clarity.

For the correct interpretation of the figure observe that (rand,head,pushpull) can be considered practically random according to the 99% confidence band, while the time series produced by (rand,head,push) shows some weak high frequency periodic behavior. The protocols (∗,rand,∗) appear to show low frequency periodic behavior with strong short-term correlation, although to confirm that further experiments are necessary. This means that apart from having a higher oscillation amplitude, random view selection also results in a much slower oscillation.

7 Self-healing capacity

As in the case of the degree distribution, the response of the protocols to a massive failure has a static and a dynamic aspect. In the static setting we are interested in the self-healing capacity of the converged overlays to a (potentially massive) node failure, as a function of the number of failing nodes. Removing a large number of nodes will inevitably cause some serious structural changes in

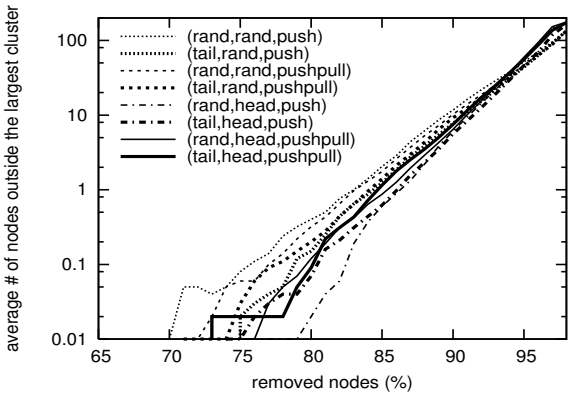


Fig. 6. The number of nodes that do not belong to the largest connected cluster. The average of 100 experiments is shown.

the overlay even if it otherwise remains "usable," that is, at least connected. In the dynamic case we would like to learn if and how the protocols can repair the overlay after a severe damage.

The effect of a massive node failure on connectivity is shown in Figure 6. In this setting the overlay in cycle 300 of the random initialization scenario was used as converged topology. From this topology, random nodes were removed and the connectivity of the remaining nodes was analyzed. In all of the $100 \times 8 = 800$ experiments performed we did not observe partitioning until removing 69% of the nodes. The figure depicts the number of the nodes outside the largest connected cluster. We observe consistent partitioning behavior over all protocol instances: even when partitioning occurs, most of the nodes form a single large connected cluster. Note that this phenomenon is well known for traditional random graphs [21].

In the dynamic scenario we made 50% of the nodes fail in cycle 300 of the random initialization scenario and we then continued running the protocols on the damaged overlay. The damage is expressed by the fact that, on average, half of the view of each node consists of descriptors that belong to nodes that are no longer in the network. We will call these descriptors dead links. Figure 7 shows how fast the protocols repair the overlay, that is, remove dead links. Based on the static node failure experiment it was expected that the remaining 50% of the overlay is not partitioned and indeed, we did not observe partitioning with any of the protocols.

8 Discussion

In our analysis of the output of the experiments presented above we first concentrate of the two main questions we posed: convergence and randomness. Then we move on to discuss the effects of the design choices in the three dimensions of the protocol space: peer selection, view selection, and symmetry of communication.

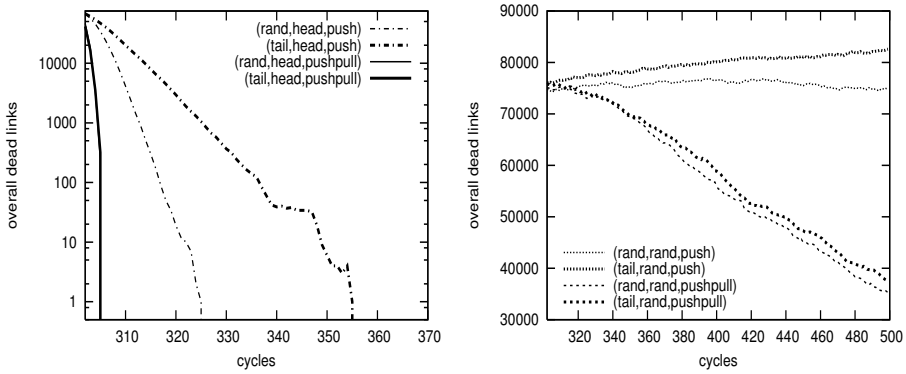


Fig. 7. The evolution of the number of dead links in the overlay following the failure of 50% of the nodes in cycle 300. The (∗,head,pushpull) protocols fully overlap. Note the different scales of the two plots.

Convergence Figures 2(a), 3(c) and 3(d) illustrate especially well that the protocols converge to the same clustering coefficient from extremely different starting conditions. Although it is somewhat less evident due to the different scales of the plots in Figure 3, average path length and average degree converge just as well. Note that the (∗,∗,push) protocols are unstable and converge very slowly in the growing overlay scenario. We will return to this issue below.

Also note that in the case of the lattice initialization scenario the initial diameter is very large but even in that case we observe rapid convergence to the desirable low diameter topology (Figure 3(a)).

Randomness Let us compare the overlays with random graphs in which the view is filled with uniform random samples of the other nodes. The behavior of the protocols we examined shows a rather colorful picture with respect to different graph properties.

In the case of average path length, clustering coefficient and average degree it is clear that protocols (∗,rand,pushpull) give us the closest approximation of the random topology, with the tail peer selection being slightly more random (see Figure 3). However, when looking at other aspects, we see a rather different picture. Degree distribution protocols (rand,head,∗) are the closest to random distribution while protocols (∗,rand,∗) are rather far from it (see Figure 4).

In all cases, we can observe that the clustering coefficient is significantly larger than that of the random graph and at the same time the average path length is almost as small. This adds all our overlay topologies to the long list of complex networks observable in nature, biology, sociology, and computer science that have a so-called "small-world" topology [1].

View selection The view selection algorithms are significantly different. Head view selection results in a more random degree distribution than the others, and it results in much less autocorrelation of the degree of a fixed node over time

(Figures 4 and 5 and Table 2). These properties make the overlays using head view selection much less vulnerable to directed attacks targeting large-degree nodes because there are no nodes with very large degree and the degree of a node changes very quickly anyway. This also means that there are no communication hot-spots in those overlays, which could result in scalability problems.

Also, head view selection repairs the overlay exponentially fast whereas random view selection can at best achieve linear speed, which can hardly be considered scalable (Figure 7). The only scenario when head view selection is not desirable is temporary network partitioning. In that case, with head view selection all partitions will forget about each other very quickly and so quick self-repair becomes a disadvantage. In practical applications the slow and quick self-healing mechanisms should be combined.

Symmetry of communication The symmetry of communication is also an important design choice. In particular, push has severe problems dealing with "bottleneck" topologies, like the star-like topology implicitly defined by the growing overlay scenario. In that case, some protocols using the push communication model were not even stable enough with respect to connectivity to participate in the experiments (Table 1), and even those that were included showed very slow convergence. The reason is that nodes that join the network in the growing scenario can get information only if the contact node pushes it to them which is very unlikely to happen because the contact node communicates only once in each cycle, just like the other nodes.

It appears that this parameter plays a more prominent role in characterizing the overall behavior of the various protocols. In general, the performance of push-pull is clearly superior compared to push-only approaches.

Peer selection In the case of peer selection we cannot observe drastic differences. In general, applying the tail selection algorithm results in slightly more randomness and slightly slower convergence at the same time. The only scenario in which opting for tail selection results in clear performance degradation is self-healing (Figure 7). In that case, (tail,head,push) converges significantly slower than (rand,head,push), although both converge still very quickly. Also, (tail,rand,push) slowly *increases* the amount of dead links which is especially undesirable.

9 Related work

Complex networks The assumption of uniform randomness has only fairly recently become subject to discussion when considering large complex networks such as the hyperlinked structure of the WWW, or the complex topology of the Internet. Like social and biological networks, the structures of the WWW and the Internet both follow the quite unbalanced power-law degree distribution, which deviates strongly from that of traditional random graphs. These new insights pose several interesting theoretical and practical problems [3]. Several dynamic

complex networks have also been studied and models have been suggested for explaining phenomena related to what we have described in the present paper [7]. This related work suggests an interesting line of future theoretical research seeking to explain our experimental results in a rigorous manner.

Unstructured overlays There are a number of protocols that are not covered by our generic scheme but that are potentially useful for implementing peer sampling. An example is the Scamp protocol [10]. While this protocol is reactive and so less dynamic, an explicit attempt is made towards the construction of a (static) random graph topology. Randomness has been evaluated in the context of information dissemination, and it appears that reliability properties come close to what one would see in random graphs. Some other protocols have also been proposed to achieve randomness [18, 22], although not having the specific requirements of the peer sampling service in mind.

Structured overlays A structured overlay [26, 25, 27] is by definition not dynamic so to utilize it for implementing the peer sampling service random walks or other additional techniques have to be applied. It is unclear whether a competitive implementation can be given considering also the cost of maintaining the respective overlay structure. Structured overlays have also been considered as a basic middleware service to applications [5]. Another issue in common with our own work is that graph theoretic approaches have been developed for further analysis [19]. Astrolabe [28] needs also be mentioned as a hierarchical (and therefore structured) overlay which although applies (non-uniform) gossip to increase robustness and to achieve self-healing properties, does not even attempt to implement or apply a uniform peer sampling service. It was designed to support hierarchical information dissemination.

10 Concluding remarks

In this paper we have identified peer sampling as an abstract middleware service. We have shown that dynamic gossip-based unstructured overlays are a natural candidate for implementing this service due to their reliability and scalability. Whereas there has been a lot of work in analyzing the behavior of structured overlay networks, this is the first attempt to analyze the behavior of a class of unstructured overlays, which so far have been simply assumed uniform random.

The main conclusion of our experiments is that the gossip-based constructions of overlays through partial views leads to many different topologies, none of which actually resembles traditional random graphs. Instead all these constructions belong to the family of small-world graphs characterized by small diameter and large clustering. Besides, many of the implementations result in highly unbalanced degree distribution. This observation indicates that gossip-based peer sampling implementations have strong links to the field of complex networks and self-organizing systems, and more generally to statistical physics, a fact which has been largely overlooked so far. This links give hope for the possibility of

the adaptation of the well established theoretical results of dynamic complex networks [7].

When considering the stable properties of various protocols, that is, which emerge from convergent behavior, it also becomes clear that different parameter settings lead to very different properties, which can be exploited according to the needs of the targeted application. For example, a strong self-healing topology may not be appropriate in the presence of temporary network partitions. In many cases, combining different settings will be necessary. Such a combination can, for instance, be achieved by introducing a second view for gossiping membership information and running more protocols concurrently.

References

1. R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, January 2002.
2. R. Albert, H. Jeong, and A.-L. Barabási. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.
3. A.-L. Barabási. *Linked: the new science of networks*. Perseus, Cambridge, Mass., 2002.
4. K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.
5. F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. In *Proc. of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS'03)*, Berkeley, CA, USA, February 2003.
6. A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database management. In *Proc. of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Vancouver, August 1987. ACM.
7. S. N. Dorogovtsev and J. F. F. Mendes. Evolution of networks. *Advances in Physics*, 51:1079–1187, 2002.
8. P. T. Eugster, R. Guerraoui, S. B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. *ACM Transactions on Computer Systems*, 21(4):341–374, 2003.
9. P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. Epidemic information dissemination in distributed systems. *IEEE Computer*, 37(5):60–67, May 2004.
10. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols. *IEEE Transactions on Computers*, 52(2), February 2003.
11. I. Gupta, K. P. Birman, and R. van Renesse. Fighting fire with fire: using randomized gossip to combat stochastic scalability limits. *Quality and Reliability Engineering International*, 18(3):165–184, 2002.
12. M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, November 2003.
13. M. Jelasity, W. Kowalczyk, and M. van Steen. An approach to massively distributed aggregate computing on peer-to-peer networks. In *Proc. of the 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'04)*, pages 200–207, A Coruna, Spain, 2004. IEEE Computer Society.

14. M. Jelasity and A. Montresor. Epidemic-style proactive aggregation in large overlay networks. In *Proc. of the 24th International Conference on Distributed Computing Systems (ICDCS 2004)*, pages 102–109, Tokyo, Japan, 2004. IEEE Computer Society.
15. M. Jelasity, A. Montresor, and O. Babaoglu. A modular paradigm for building self-organizing peer-to-peer applications. In G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, number 2977 in LNCS, pages 265–282. Springer, 2004.
16. D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03)*, pages 482–491. IEEE Computer Society, 2003.
17. A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. Probablistic reliable dissemination in large-scale systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(3), March 2003.
18. C. Law and K.-Y. Siu. Distributed construction of random expander graphs. In *Proc. of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2003)*, San Francisco, California, USA, April 2003.
19. D. Loguinov, A. Kumar, V. Rai, and S. Ganesh. Graph-theoretic analysis of structured peer-to-peer systems: Routing distances and fault resilience. In *Proc. of ACM SIGCOMM*, pages 395–406, 2003.
20. A. Montresor, M. Jelasity, and O. Babaoglu. Robust aggregation protocols for large-scale overlay networks. In *Proc. of the 2004 International Conference on Dependable Systems and Networks (DSN)*, pages 19–28, Florence, Italy, 2004. IEEE Computer Society.
21. M. E. J. Newman. Random graphs as models of networks. In S. Bornholdt and H. G. Schuster, editors, *Handbook of Graphs and Networks: From the Genome to the Internet*, chapter 2. John Wiley, New York, NY, 2002.
22. G. Pandurangan, P. Raghavan, and E. Upfal. Building low-diameter peer-to-peer networks. *IEEE Journal on Selected Areas in Communications (JSAC)*, 21(6):995–1002, August 2003.
23. R. Pastor-Satorras and A. Vespignani. Epidemic dynamics and endemic states in complex networks. *Physical Review E*, 63:066117, 2001.
24. B. Pittel. On spreading a rumor. *SIAM Journal on Applied Mathematics*, 47(1):213–223, February 1987.
25. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proc. of ACM SIGCOMM*, pages 161–172, 2001.
26. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In R. Guerraoui, editor, *Middleware 2001*, volume 2218 of *LNCS*, pages 329–350. Springer, 2001.
27. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM SIGCOMM*, pages 149–160, 2001.
28. R. Van Renesse, K. P. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Transactions on Computer Systems*, 21(2), May 2003.
29. S. Voulgaris and M. van Steen. An epidemic protocol for managing routing tables in very large peer-to-peer networks. In *Proc. of the 14th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, (DSOM 2003)*, number 2867 in LNCS. Springer, 2003.
30. D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.