Inexpensive Membership Management for
Unstructured P2P Overlays

Spyros Vouglaris, Daniela Gavidia, Maarten van Steen
Contact author: Spyros Vouglaris (spyros@cs.vu.nl)

Keywords: Membership management, Peer-to-Peer, Epidemic Protocols, Unstructured Overlays, Random Graphs

VRIJE UNIVERSITEIT AMSTERDAM
Inexpensive Membership Management for Unstructured P2P Overlays

Spyros Voulgaris, Daniela Gavidia Simonetti, Maarten van Steen *

Abstract

Unstructured overlays form an important class of peer-to-peer networks, notably when content-based searching is at stake. The construction of these overlays, which is essentially a membership management issue, is crucial. Ideally, the resulting overlays should have low diameter and be resilient to massive node failures, which are both characteristic properties of random graphs. In addition, they should be able to deal with a high node churn (i.e., expect high-frequency membership changes). Inexpensive membership management while retaining random-graph properties is therefore important.

In this paper, we describe a novel gossip-based membership management protocol that meets these requirements. Our protocol is shown to construct graphs that have low diameter, low clustering, highly symmetric node degrees, and that are highly resilient to massive node failures. Moreover, we show that the protocol is highly reactive to restoring randomness when large numbers of nodes fail.

Keywords: Membership management, Peer-to-Peer, Epidemic Protocols, Unstructured Overlays, Random Graphs

*Vrije Universiteit, De Boelelaan 1081A, 1081 HV Amsterdam, The Netherlands
email: {spyros,daniela,steen}@cs.vu.nl

1
1 Introduction

The growth of the Internet in terms of size and speed, as well as the plethora of network applications and services that have been deployed in the last few years clearly indicate a shift from the traditional client-server model to that of highly-distributed, peer-to-peer (P2P) systems. The main philosophy behind these systems is communal collaboration among peers: sharing both duties and benefits. By distributing responsibilities across all participating peers, they can collectively carry out large-scale tasks in a simple and generally scalable way, that would otherwise depend on expensive, dedicated, and hard to administer centralized servers.

A distinguishing feature of P2P systems is that the peers jointly maintain an overlay network. Unlike traditional layer-3 networks, the structure of these overlay networks is not dictated by the (often fairly static) physical presence and connectivity of hosts, but by logical relationships between peers. In particular, P2P overlay networks are generally designed to handle a much higher rate concerning the joining and leaving of nodes, while at the same time assume that membership behavior is roughly the same for all nodes. In other words, they are designed to handle highly dynamic, symmetric networks. Overlay management is therefore a key issue in designing P2P systems.

There are currently two main categories of P2P systems in terms of overlay management. Structured P2P systems impose a specific linkage structure between nodes. Distributed Hash Tables (DHTs) [2] are typical examples in this category. They link peers based on their IDs in a way to enable efficient ID-based routing among them. In contrast, in unstructured P2P systems peers are not linked according to a predefined deterministic scheme. Instead, links are created either randomly (as is typically done in Gnutella), or are based on semantic proximity of nodes (leading to what are known as semantic overlay networks). Unstructured P2P systems are primarily designed to support rapid information dissemination and content-based searching.

In this paper we concentrate on gossip-based unstructured P2P systems [6]. These systems aim at exploiting randomness to disseminate information across a large set of nodes. A key issue is to keep the overlay connected even in the event of major disasters without maintaining any global information or requiring any sort of administration. Connections between nodes in these systems are highly
dynamic. Gossiping networks generally exhibit self-healing behavior with respect to major network
disasters. It has also been observed that these overlays exhibit properties of small-world and scale-free
networks [1, 10]. However, in many cases it is better to construct overlays that are close to random
graphs [3].

The problem we address is that of building and maintaining a very large, connected overlay net-
work that exhibits important properties of random graphs. In particular, we are interested in inexpen-
sive membership management, by which we mean that any disruption of the overlay resulting from
joining or leaving nodes should be quickly and efficiently corrected. We assume that nodes maintain
a small, partial view of the entire network. Our starting point is the view exchange protocol described
in [12]. This so-called shuffling protocol ensures that connectivity of the overlay is maintained as long
as membership does not change.

We make two contributions. First, we provide an experimental analysis of the basic shuffling
protocol for large networks, examining properties such as clustering and node degree distribution.
These experiments have not been conducted before, and, in particular, not on large networks. We
demonstrate that shuffling is indeed a promising exchange protocol.

Second, and most importantly, we describe CYCLON1, a complete framework for inexpensive
membership management. CYCLON introduces an enhanced version of shuffling, which results in
node-degree distributions that exhibit better properties than those found in overlays resulting from
basic shuffling, or even in random graphs. Moreover, it includes better, in terms of efficiency and
quality, management of node additions and removals, which allows us to establish truly inexpensive
membership management that does not disrupt the randomness of the overlay network.

The paper is organized as follows. We start with explaining the basic protocols underlying CY-
CLON in Section 2, followed by an analysis of basic properties in Section 3. The effects of adding and
removing nodes are discussed in Section 4 and 5, respectively. CYCLON’s robustness in the presence
of (massively) failing nodes is discussed in Section 6. Related work is presented in Section 7, and we
conclude in Section 8.

1The name CYCLON was inspired by the protocol’s power in mixing nodes in the network, sort of like a tornado. It is
also inspired by the Greek origin of the word, kyklos (=circle), due to the uniformity it imposes on the overlay.
2 The Protocol

2.1 Basic Shuffling

The basic shuffling algorithm, introduced in [12], is a simple peer-to-peer communication model. It forms an overlay and keeps it connected by means of an epidemic algorithm. The protocol is extremely simple: each peer knows a small, continuously changing set of other peers, called its neighbors, and occasionally contacts a random one of them to exchange some of their neighbors.

More formally, each peer maintains a neighbor list in a small, fixed-sized cache of \( c \) entries (with typical value 20, 50, or 100). A cache entry contains the network address (i.e., IP address and port) of another peer in the overlay. Each peer \( P \) repeatedly initiates a neighbor exchange operation, known as shuffle, by executing the following six steps:

1. Select a random subset of \( l \) neighbors (\( 1 \leq l \leq c \)) from \( P \)'s own cache, and a random peer, \( Q \), within this subset, where \( l \) is a system parameter, called shuffle length.

2. Replace \( Q \)'s address with \( P \)'s address.

3. Send the updated subset to \( Q \).

4. Receive from \( Q \) a subset of no more than \( l \) of \( Q \)'s neighbors.

5. Discard entries pointing to \( P \), and entries that are already in \( P \)'s cache.

6. Update \( P \)'s cache to include all remaining entries, by firstly using empty cache slots (if any), and secondly replacing entries among the ones originally sent to \( Q \).

Upon reception of a shuffling request, peer \( Q \) randomly selects a subset of its own neighbors, of size no more than \( l \), sends it to the initiating node, and executes steps 5 and 6 to update its own cache accordingly.

2.2 Enhanced shuffling

CYCLON employs an enhanced version of shuffling, that, as we shall show in subsequent sections, among other things improves the quality of the overlay in terms of randomness. Enhanced shuffling
follows the same model as basic shuffling. The key difference is that nodes do not randomly pick which neighbor to shuffle caches with. Instead, they select the neighbor whose information was the earliest one to have been injected in the network.

In enhanced shuffling nodes initiate neighbor exchanges periodically, yet not synchronized, at a fixed period $\Delta T$. In addition to the network address, cache entries contain an extra field called age, which denotes roughly the age of the entry expressed in $\Delta T$ intervals since the moment it was created by the node it points at.

The enhanced shuffling operation is performed by letting the initiating peer $P$ execute the following seven steps:

1. Increase by one the age of all neighbors.
2. Select neighbor $Q$ with the highest age among all neighbors, and $l - 1$ other random neighbors.
3. Replace $Q$’s entry with a new entry of age 0 and with $P$’s address.
4. Send the updated subset to peer $Q$.
5. Receive from $Q$ a subset of no more that $l$ of its own entries.
6. Discard entries pointing at $P$ and entries already contained in $P$’s cache.
7. Update $P$’s cache to include all remaining entries, by firstly using empty cache slots (if any), and secondly replacing entries among the ones sent to $Q$.

Like in basic shuffling, the receiving node $Q$ replies by sending back a random subset of at most $l$ of its neighbors, and updates its own cache to accommodate all received entries. It does not increase, though, any entry’s age until its own turn comes to initiate a shuffle.

From now on, any references to shuffling apply to both basic and enhanced shuffling. Otherwise, the shuffling type will be explicitly specified.

Note that after node $P$ has initiated a shuffling operation with its neighbor $Q$, $P$ becomes $Q$’s neighbor, while $Q$ is no longer a neighbor of $P$. That is, the neighbor relation between $P$ and $Q$ reverses direction.
3 Basic properties

In this section we show and analyze the basic properties of the basic and enhanced shuffling epidemic protocols.

3.1 Connectivity

The fundamental property of shuffling is that, given a fail-free environment, connectivity of the overlay is guaranteed. The proof comes as a natural consequence of the shuffling operation. Let $S_p$ and $S_q$ be two connected subsets of the overlay, and let there be at least one link between them, say from $P \in S_p$ to $Q \in S_q$. Only shuffles involving $P$ can have an effect on its link to $Q$, and therefore on the connectivity between the two subsets. If $P$ shuffles with another node $P^* \in S_q$, $Q$ will remain the neighbor of either $P$ or $P^*$. If, on the other hand, $P$ initiates a shuffle with $Q$, $P$ becomes $Q$’s neighbor, so $S_p$ and $S_q$ remain connected. Therefore, no shuffling operation can result in $S_p$ and $S_q$ (or generally any two subsets of the overlay) becoming disconnected. This proves shuffling to be a connectivity preserving operation.

3.2 Convergence

As it turns out, shuffling has some desirable statistical properties. In order to observe the characteristics of the overlay, we need to consider the connectivity graph, that is, the graph having the peers as vertices, and the links between them as (directed) edges. We consider the undirected version of the connectivity graph, which is taken by simply dropping the direction of the edges. The motivation behind this is that we are interested rather in the "can-communicate” than the "knows-about” version of the graph. A node has the same potential to communicate with another node either if the first is a neighbor of the second or vice versa.

The shortest path length between nodes $P$ and $Q$ is the minimum number of edges needed to traverse to reach $Q$ from $P$. The average path length is the average of the shortest path lengths between any two nodes. The average path length is a metric of the number of hops (and hence, communication costs and time) to reach nodes from a given source. A small average path length is therefore essential
Figure 1: (a) Average shortest path length between two nodes for different cache sizes. (b) Average clustering coefficient taken over all nodes.

for broadcasting or, generally, information dissemination applications.

We conducted a series of experiments involving networks containing up to 100,000 nodes. To study the emergent behavior of the protocol, we define a cycle to be the time period during which a number of shuffle operations equal to the number of nodes have been made. Since nodes initiate shuffle operations periodically and with the same period, a cycle coincides with the shuffle period $\Delta T$. Note that during a cycle, each node has initiated a shuffling operation exactly once. We studied the protocol’s emergent behavior by observing its state at times $0, \Delta T, 2\Delta T$, etc.

Figure 1(a) demonstrates a significant aspect of the emergent behavior of shuffling. It clearly shows that the average path length converges to a very small value, which coincides with the average path length of a random graph with the same number of edges.

The clustering coefficient of a node is defined as the ratio of the existing links among the node’s neighbors over the total number of possible links among them. It basically shows to what percentage the neighbors of a node are also neighbors among themselves. The average clustering coefficient is the clustering coefficient averaged across all nodes in the network. It is generally undesirable for an overlay to have a high average clustering coefficient for two reasons. First, it weakens the connectivity of a cluster to the rest of the network, therefore increasing the chances of partitioning. Second, it is not optimal for information dissemination applications due to the high number of redundant message
Figure 2: (a) Average shortest path length between two nodes for different cache sizes. (b) Average clustering coefficient taken over all nodes.

delivers within highly clustered node communities.

Figure 1(b) shows that shuffling exhibits convergent behavior for the clustering coefficient. In our experiments, the clustering coefficient always converged to values practically equal to the clustering coefficient of random graphs. Moreover, both the average path length and the clustering coefficient converge almost exponentially (linearly in the log-linear graph).

Note that these experiments ran for several thousand cycles. However, only the initial cycles are depicted, as the values remained stable for all the subsequent cycles, indicating convergent behavior. Also note that we carried out experiments bootstrapped in various different ways, but they all converged to the exact same values. The experiments presented in figure 1 were —intentionally— bootstrapped with the worst imaginable setting. Nodes were set to form a “chain,” each node having a single neighbor, namely its previous one. This way, the average path length was initially the longest possible (i.e., 99,999 hops were needed to reach the first from the last one).

Figure 2 shows the converged values for the average path length and the clustering coefficient, for overlays of different numbers of nodes and cache sizes. Two initial observations can be made. First, the average path length increases logarithmically as a function of the number of nodes in the overlay. Second, the clustering coefficient drops exponentially (linearly in the log-log graph) as a function of the number of nodes.
What is more interesting is that both the average path length and the clustering coefficient converge to the exact values expected in a random graph of the same number of nodes and links. A random graph is a graph where an edge between two random nodes exists with a probability $p$. Consequently, $p$ is equal to the ratio of existing links among nodes, over the total number of node pairs (total number of possible links). Also, in random graphs, the average clustering coefficient, $C_{\text{rand}}$, is equal to $p$. Therefore, it follows that

$$C_{\text{rand}} = p = \frac{\text{#links}}{\text{total # of possible links}} = \frac{\text{#links}}{\frac{N(N-1)}{2}}$$

(1)

where $N$ is the number of nodes. For the sake of comparison, we consider random graphs with an equal number of links as in our graphs. In our overlay, the number of links is $N \times c$, where $c$ is the cache size. By substituting that in (1), we get:

$$C_{\text{rand}} = \frac{2^c c}{N - 1}$$

(2)

One can observe that formula 2, which gives the clustering coefficient for random graphs of the same number of nodes and edges as ours, also provides the exact values retrieved through experimentation in 2(b). This proves that the overlays formed by shuffling have the same clustering coefficient as equivalent random graphs.

### 3.3 Degree distribution

The degree of a node is the number of links it has to other nodes, in the undirected connection graph. The interest in the degree distribution stems from three reasons. First, the degree distribution is highly related to the robustness of the overlay in the presence of failures as it shows the existence of weakly connected nodes and massively connected hubs. Second, it is an indication of the way epidemics are spread. Third, it provides an indication of how fairly links are distributed among nodes, and, as a consequence, an indication of the distribution of resource usage (processing, bandwidth) across nodes. For the sake of robustness, efficient information dissemination, and load balancing, it is desirable to have a balanced, uniform distribution of links across all nodes of the overlay. In other words, it is desirable to have a degree distribution with low standard deviation.
Figure 3: In-degree distribution in converged 100,000 node overlay, for basic shuffling, enhanced shuffling, and an overlay with randomly chosen links.

In the directed version of the connection graph, we distinguish between the out-degree, and the in-degree of a node, which are the number of edges leaving from and ending at the node respectively. In our case, the out-degree of every node is fixed, and equal to the cache size. Therefore, we concentrate on observing the in-degree distribution of our overlays.

Figure 3 shows the in-degree distribution for both basic and enhanced shuffling for two different cache sizes. In both protocols, the majority of the nodes have an in-degree equal to the cache size, while the number of nodes having larger or smaller in-degrees drops symmetrically according to the shift from the cache size.

It is, however, clear that enhanced shuffling does a significantly better job with respect to spreading out the links extremely evenly across all nodes. For the experiment with cache size 20, 88.89% of the nodes have an in-degree of $20 \pm 5\%$. For the experiment with cache size 50, 97.09% of the nodes have in-degree of $50 \pm 5\%$. The respective percentages for basic shuffling are $36.22\%$ and $38.47\%$.

3.4 Dependency on shuffle length

We conducted a series of experiments to examine the effect of the shuffle length on convergence. Interestingly, for all overlays we tried, the converged state with respect to the average path length, average clustering coefficient, and in-degree distribution, proved to be independent of the shuffle length used. The only effect of the shuffle length was in the number of cycles it took to reach the
converged state.

We used the in-degree distribution as a metric to identify at which cycle an overlay converges. As the in-degree distribution does not converge to exact numbers, but to a certain graph shape whose points keep fluctuating slightly, we had to use an approximation algorithm to identify the convergence point. In particular, we calculated the in-degree distributions for the first 1000 cycles of each experiment. In all cases, by looking at the distribution series we could tell that they were certainly converged well before the 900th cycle. We computed the average in-degree distribution of the last hundred cycles, namely cycles 900-999. Subsequently, for each cycle $i = \{900 \ldots 999\}$ we computed the sum of squared errors $^2E_i$, between this cycle’s in-degree distribution and the average one, and we figured out the maximum, $^2E_{\text{max}\{900\ldots999\}}$. This was used as the maximum threshold of the sum of squared errors to consider a distribution converged. Finally, starting from cycle 0, we checked one distribution at a time to find the first cycle $i$ whose $^2E_i$ was below that threshold. This cycle was logged as the cycle for which this experiment converged.

To demonstrate the shuffle length’s effect independently of the initial condition, two different bootstrapping methods were used. The first one, chain, is the one described in section 3.2: considering nodes in a line, each node has a single link to its previous one, forming a chain topology. In the second bootstrapping method, star, all nodes initially have a single neighbor, the same one for all of them, essentially forming a star topology.

$$^2E_i = \sum_{k=0}^{\infty} (x_{ik} - \bar{x}_i)^2$$, where $X_i = \{x_{i0}, x_{i1}, x_{i2}, \ldots\}$ is the $i$-cycle distribution, and $\bar{X}_i = \{\bar{x}_0, \bar{x}_1, \bar{x}_2, \ldots\}$ is the average distribution.
Figure 4 presents the number of cycles an experiment took to converge as a function of the shuffle length, for 100,000 nodes, and cache sizes 20 and 50. In all cases, shuffling just one neighbor at a time took clearly the longest. Shuffling two, three, or more neighbors, gradually sped up the process. However, no significant improvement was noticed beyond shuffle lengths of about eight or ten. Counter intuitively, convergence speed degraded suddenly when shuffling the whole cache, or almost all of it. The reason behind slow convergence either when shuffling too few or too many neighbors is that in either case caches are not mixed up too much by each shuffling operation. Therefore, a cache is only minimally (if at all) enriched with new links, even if it has moved as a whole to a new node in the case of full cache shuffling.

This result has more significance when bandwidth is an issue. In such a case, choosing a shuffle length as low as around six or eight results in nearly optimal convergence speed, since the shuffle length has a direct effect in the amount of bandwidth used by CYCLON.

4 Adding nodes

CYCLON introduces a new method for nodes to join the overlay efficiently, without disrupting randomness. To join, a new node simply needs to know any node that is already part of the overlay, called its introducer. Such a node can be discovered in various ways, including broadcasting in the local network, making use of a designated multicast group, or even contacting a well-known server, etc. Finding such an introducer is out of the scope of this paper. Here, we are interested in how to join by means of an introducer without affecting the basic properties of the system.

To this end, a key observation is that due to the randomness of the connectivity graph, a random walk of length at least equal to the average path length is guaranteed to end at a random node of the overlay, irrespectively of the starting node.

Based on this observation, a new node $P$ can join in a fairly straightforward manner. $P$'s introducer initiates $c$ (cache size) random walks, setting their TTL (time to live) to a small value close to the expected average path length, such as four or five. A node $Q$ where a random walk ends, replaces one of its cache’s entries with a new entry of age 0 and the address of $P$. $Q$ then forwards the replaced
cache entry to $P$, who, in turn, has to include the entry in one of its cache’s empty slots. In effect, this operation is equivalent to $P$ initiating a shuffle of length 1 with a nonadjacent node, namely $Q$. The join operation ends when all $c$ random walks have ended and the corresponding neighbor exchanges have been accomplished.

We claim that this join operation makes it impossible for an external observer to distinguish $P$ as being different from the rest of the nodes, or to discover any randomness disruption in the overlay. First, $P$’s cache is filled up with $c$ randomly chosen neighbors, which renders the values of $P$’s average path length to reach any other node as well as its clustering coefficient, indistinguishable from the respective values of other, older nodes. Second, since there are $c$ random nodes in the overlay that know $P$, $P$’s in-degree is equal to the cache size. Third, no other node’s in-degree has been modified.

In the presence of node failures or an unreliable network, some of the random walks may fail. Note, however, that a node’s join does not depend on the complete execution of this join procedure. We ran experiments (not presented here) that showed that a node can join by simply being involved in a shuffle with a single participating node. In that case, though, it will take a few cycles until the new node’s properties become indistinguishable from the respective properties of other nodes. The join procedure described above is meant as a means of efficient node joining at a cost of a constant number of messages.

5 Removing nodes

In a dynamically changing overlay, nodes may leave for various reasons and in various ways. We make no distinction between nodes disconnecting gracefully or abruptly. What we are interested in is that, once a node disconnects, other nodes should detect it and remove any pointers to it in a timely manner. We consider pointers to disconnected nodes to consist a sort of contamination for the cache, as they take up slots that could be otherwise holding valid, useful links. Particularly in highly dynamic environments, timely elimination of dead links is crucial for the robustness of the overlay.

In order to keep the protocol simple and inexpensive, we do not add any explicit messages (such as frequent pings) to detect disconnected nodes. Instead, CYCLON uses a transparent dead-link detection
mechanism, based on the default shuffling message exchange. We do, however, employ an effective strategy (by means of the age field introduced in enhanced shuffling) to improve timely detection of disconnected nodes “for free,” as a natural consequence of the emergent behavior of enhanced shuffling.

When a node tries to initiate a shuffle with a neighbor and gets no reply within a predefined timeout, it simply assumes that neighbor to be disconnected and removes the corresponding entry from its cache. This way, dead links are gradually being removed.

In the case of basic shuffling, the detection of dead links relies on chance and takes unbounded time. In CYCLON’s enhanced shuffling, though, the age field defines a key priority in which neighbors are contacted. A fresh entry (one that has low age), and therefore an entry of a node more probable to be still alive, is less likely to be chosen for shuffling. In contrast, an entry that has been injected in the network several cycles ago (and has since been hopping around between nodes due to shuffles), is more likely to be the oldest in the cache currently holding it, and therefore more likely to be selected for shuffling. In general, the longer an entry stays in the network, the higher the chances it is selected for shuffling. When $P$ initiates a shuffle and selects an entry with, say, $Q$’s address, that entry is then replaced by a new entry with $P$’s address and age 0. This process naturally recycles the entries, maintaining an equilibrium with respect to their ages, consequently limiting the lifetime of an entry. This way, it is no longer possible for old entries of disconnected nodes to indefinitely hop between nodes in the overlay.

To demonstrate the advantages of CYCLON’s enhanced shuffling with respect to node removal, we ran experiments where we suddenly killed half of the nodes after the overlay had converged. As we shall show in the next section, such a drastic change poses no threat to the (remaining) overlay’s connectivity. We observed how long it took for the remaining nodes to “forget” the dead ones. Figure 5 shows the respective graphs for the experiment with 100,000 nodes, out of which 50,000 were killed at once. Figure 5(a) shows how long dead nodes are still referenced, while Figure 5(b) shows the number of dead links that are maintained since the nodes were killed. It is clear that enhanced shuffling limits the detection of dead nodes to a number of cycles equal to (in fact less than) the cache size, while basic shuffling takes almost an order of magnitude more cycles to decontaminate the
6 Robustness - Self healing behavior

In the previous section, we dealt with node disconnections, and we mentioned that killing half of the nodes at once does not threaten the connectivity of the remaining ones. In this section, we explore CYCLON’s limits in terms of robustness to node disconnections. Shuffling proves to be a very strong and robust epidemic protocol with respect to keeping an overlay connected. Moreover, it appears to exhibit robustness to node disconnections similar to the one found in random graphs.

We conducted experiments as follows. We used CYCLON to create overlays (until they converged), and subsequently examined the effect on the connectivity of removing nodes. Figure 6 presents the results for networks with initially 100,000 nodes, and cache sizes 20, 50, and 100, respectively. For the sake of comparison, it also presents the same graphs for overlays of cache size 20 and 50, where the neighbors were randomly chosen among the whole node set. Figure 6(a) shows the number of disjoint clusters as a function of the percentage of nodes removed. Note that the number of clusters decreases as we approach 100% node removal because the total number of surviving nodes becomes too small. Figure 6(b) shows the number of nodes not belonging to the largest cluster, in log scale.

These graphs show considerable robustness to node failures, especially considering the fact that in the early stages of clustering very few nodes are out of the largest cluster, which indicates that surviving nodes’ caches.
most nodes are still connected in a single large cluster. Moreover, shuffling appears to share the same robustness properties with overlays where each node’s neighbors are a random sample of the nodes in the network.

Note that the graph for the experiment with cache size 100 is practically a flat line. That is, for 100,000 nodes and cache size 100, the overlay created is so robust, that no matter how many nodes are removed, the remaining ones remain connected in a single cluster.

The effect of the cache size on the overlay’s robustness is shown in figure 7. We carried out 100 experiments, with cache sizes 1, 2, . . . , 100, and for each of them we determined the percentage of random nodes needed to be removed in order to partition the overlay. It can be seen that there is a critical value of the cache size around eleven. Overlays with smaller cache sizes exhibit significantly worse behavior with respect to robustness. On the other hand, overlays with cache size over 85 or 90, are almost impossible to partition, no matter how many nodes are removed.

It is important to point out that the results presented in this and the previous section, suggest that CYCLON is capable of repairing an overlay after a serious disaster, a property often referred to as self-healing behavior. This comes as a consequence of the following two facts. First, the overlay has proven to be highly resilient to large-scale node failures. Second, once such a massive failure has occurred, the surviving nodes quickly strengthen the connectivity among themselves by replacing links to dead nodes with valid links in a timely manner.

7 Related work

There are currently several efforts in constructing unstructured overlays that share properties with random graphs. We have already described basic shuffling introduced in [12], forming the starting point for own work described in this paper.

Another example is the Scamp protocol [5]. Scamp is reactive, in the sense that cache exchanges take place only when nodes join, leave, or a failure is detected. As it turns out, the protocol exhibits similar properties in comparison to random graphs when considering its capabilities for information dissemination and recovering from massive node failures. However, no thorough analysis has been
Figure 6: (a) Number of disconnected clusters (b) Number of nodes disconnected from the largest cluster.

Figure 7: Tolerance to node removal, as a function of the cache size.
undertaken to compare the communication graph with random graphs as we did, but it is known that there are important differences [8].

In many unstructured overlays, such as CYCLON, scalability of the network is achieved by maintaining a partial view on the entire network. The construction of the network itself, that is, membership management, is crucial as we have argued in this paper. It is interesting to see that the assumption is sometimes made that the communication graph resulting from a specific membership protocol is random. However, as is shown in [6], there is a large family of membership protocols for unstructured overlays for which this assumption is false. This includes the work on lightweight probabilistic multicasting [4], as well as the newscast system [7]. As it turns out, such membership protocols generally lead to small-world graphs, which distinguish themselves from random graphs by a high clustering coefficient. So far, shuffling appears to fall outside the category of small-world networks.

Of course, random graphs may not be the best structure for communication networks. Alternative schemes are described in [11, 9]. In these cases, the requirements of low diameter, resilience to massive node failures, and inexpensive membership management lead to specific graph-construction protocols. We argue that with expanded shuffling these requirements are all met, and that the resulting communication graphs allow us to adopt the rigorous analysis of random graphs. Moreover, in contrast to, for example [11], there is no need to use a central server.

In this light, it is also interesting to mention the recent work on Phenix for the decentralized construction of low-diameter, scale-free networks [14]. In Phenix, an additional goal is to construct networks that are resilient to massive malicious attacks. We have not yet examined this feature for CYCLON, but suspect its good randomness properties will help in also making it attack resilient.

8 Conclusions

In this paper we presented CYCLON, a complete framework for inexpensive membership management in very large P2P overlays. CYCLON is highly scalable, very robust, and completely decentralized. Most important is that the resulting communication graphs share important properties with random graphs. Besides the fact that desirable features such as low diameter and robustness are supported,
this similarity justifiably opens the possibility to rigorously analyze the networks.

We also conclude that CYCLON is an improvement of the basic shuffling protocol developed by Stavrou et al. [12]. We offer a scalable and inexpensive membership protocol, achieve better node-degree distributions, and significantly lower the contamination of caches concerning stale references to previous members.

An important next step in our research will be the replacement of the periodic cache exchanges, with a reactive exchange protocol, as in Scamp. We envisage that this replacement will lead to a better utilization of network resources, and incur only minimal costs for detecting failed nodes. Another important subject that we will address is taking network proximity into account. The latter will be largely based on our scalable latency estimation service, described in [13].

References


