Robust and efficient membership management in large-scale dynamic networks

Poonpakdee, P. and Di Fatta, G. (2017) Robust and efficient membership management in large-scale dynamic networks. Future Generation Computer Systems, 75. pp. 85-93. ISSN 0167-739X doi: https://doi.org/10.1016/j.future.2017.02.033 Available at http://centaur.reading.ac.uk/71079/

Published version at: http://dx.doi.org/10.1016/j.future.2017.02.033
To link to this article DOI: http://dx.doi.org/10.1016/j.future.2017.02.033

Publisher: Elsevier

Robust and Efficient Membership Management in Large-scale Dynamic Networks

Pasu Poonpakdee[a], Giuseppe Di Fatta[b]

[a]*Department of Industrial Engineering, Faculty of Engineering, King Mongkuts Institute of Technology Ladkrabang, 10520, Thailand*
[b]*Department of Computer Science, University of Reading, Whiteknights, Reading, Berkshire, RG6 6AY, United Kingdom*

Abstract

Epidemic protocols are a bio-inspired communication and computation paradigm for large-scale network system based on randomised communication. These protocols rely on a membership service to build decentralised and random overlay topologies. In large-scale, dynamic network environments, node churn and failures may have a detrimental effect on the structure of the overlay topologies with negative impact on the efficiency and the accuracy of applications. Most importantly, there exists the risk of a permanent loss of global connectivity that would prevent the correct convergence of applications. This work investigates to what extent a dynamic network environment may negatively affect the performance of Epidemic membership protocols. A novel *Enhanced Expander Membership Protocol* (*EMP+*) based on the expansion properties of graphs is presented. The proposed protocol is evaluated against other membership protocols and the comparative analysis shows that *EMP+* can support faster application convergence and is the first membership protocol to provide robustness against global network connectivity problems.

Keywords: Epidemic protocols, expander graphs, node churn, large-scale systems, decentralised algorithms

1. Introduction

In order to disseminate information in large-scale networked systems, centralised approaches are not efficient and robust solutions as they may suffer from communication bottlenecks and fault intolerance. Epidemic, or

Gossip-based, protocols are a fully decentralised paradigm for communication and computation, which are intrinsically fault-tolerant. They have been shown to be particularly suitable for information dissemination and data aggregation in large-scale networks.

A number of applications based on Epidemic protocols have been proposed to serve different purposes in different environments. For example, Epidemic protocols have been employed to implement applications for Peer-to-Peer (P2P) overlay networks [1, 2, 3], distributed computing [4], mobile ad hoc networks (MANET) [5], wireless sensor networks (WSN) [6, 7, 8, 9, 10], failure detection [11], distributed data mining [12, 13, 14] and exascale high performance computing [15, 16, 17].

The fundamental mechanism of Epidemic protocols inherits the advantages of robustness and scalability from a randomised communication strategy. To perform randomised communication, a peer sampling operation is required to be available as an underlying network service. In large-scale and dynamic networks it is unrealistic to assume that each node has a complete knowledge of the global topology. In this context, Epidemic Membership Protocols are used to provide a practical implementation of the peer sampling service, which returns a random node with uniform probability, similarly to a random selection from the global view of the system [18]. Instead of maintaining a complete list of nodes at each node, a membership protocol builds a local partial view (cache) of the system, which is continuously and randomly changed. The distributed set of local views implicitly defines a dynamic overlay topology.

Several membership protocols have been proposed (e.g., [19, 20, 21]), which have been designed for generating random overlay topologies. Among other properties of graphs, the expansion quality is a fundamental mathematical concept [22] that provides a direct measure of the ability of a graph to support fast information propagation. The expansion property of graphs [23] has inspired the *Expander Membership Protocol* (*EMP*) [24]. *EMP* is based on a push-pull scheme that introduces a bias in the random node selection in order to maximize the expansion property of the overlay topology.

A specific aim of this work is to address the problems that arise in Epidemic protocols when adopted in dynamic networks. Node churn and node failures have a detrimental effect on the overlay topology and its global connectivity can be significantly weakened. Failed nodes leave behind cache entries referring to them, which have become invalid (broken links). A decrease of the propagation efficiency and a loss of global connectivity are

the two main issues that may be caused by a degradation of the overlay topology. Surprisingly, in weakly connected overlay topologies, the internal mechanisms of membership protocols can turn the topology from a single connected component into multiple connected components, seriously hindering the accuracy and performance of the applications.

Many membership protocols provides mechanisms aimed at maintaining the global connectivity in strongly connected graphs, while none of them has been devoted to recover the connectivity when lost. This work includes an analysis of the external causes and of the internal mechanisms in membership protocols that may cause a loss of the global connectivity.

In this work a comprehensive description and analysis of the Enhanced Expander Membership Protocol (*EMP+*) is provided. The objectives of *EMP+* are preserving the global connectivity and providing fast and robust convergence to random overlay topologies to support the convergence speed of applications. *EMP+*, is obtained with the introduction of two procedures into *EMP*, the Interleave Management Procedure (IMP) [25] and a connectivity recovery mechanism.

EMP+ is an epidemic membership protocol that, apart from being fault-tolerant and scalable as any epidemic protocol, exhibits a better performance in terms of the convergence speed of global aggregation operations than previous protocols. Moreover, EMP+ has unique features that address some real network issues for asynchronous and dynamic conditions. The main contributions of EMP+ are as follows:

- it limits the negative effects of message interleaving events,
- it incorporates a novel mechanism for global connectivity recovery,
- it efficiently addresses the problem of broken links caused by churn and failures and
- it ultimately improves the convergence speed of global aggregation operations under dynamic network conditions.

A detailed comparative analysis for both static and dynamic networks is carried out for several membership protocols and with three different types of initial overlay topology. The experimental analysis shows that *EMP+* improves the convergence speed of global data aggregation operations and is the only membership protocols that is able to recover from the loss of global connectivity.

The rest of the paper is organised as follows. Section 2 reviews related work on Epidemic membership protocols and discusses their mechanisms and drawbacks. The intuition behind *EMP+* and a detailed description of its internal mechanisms are presented in Sections 3 and 4. The simulations that were carried out, the experimental results and their analysis are presented in Section 5. Finally, Section 6 draws some conclusions and provides possible directions of future work.

2. Membership Protocols

The node sampling service provided by Epidemic Membership Protocols is considered a fundamental abstraction in distributed systems [18]. In large-scale systems, nodes cannot build and maintain a complete directory of memberships. A membership protocol builds and maintains a partial view of the system, which is used to provide the random node selection service. The distributed set of views implicitly defines an overlay topology $G = (V, E)$. A membership protocol periodically and randomly changes the local views, thus generating a sequence of random overlay topologies $\Gamma = \{G_i\}$, with $G_i = (V_i, E_i)$ being the overlay topology at the protocol cycle i.

The required assumptions are that the physical network topology is a connected graph, a routing protocol is available and an initialisation mechanism for the overlay topology is provided. Typically the formulation of Epidemic protocols is based on a periodic process (Gossip cycle) for the exchange of the local state with a random node in the system.

Several membership protocols have been proposed in the literature. This section describes how they provide the node sampling service and the broken link removal mechanism, and the drawbacks of their specific algorithms.

The Node Cache Protocol [21] is a simple membership protocol that adopts a symmetric push-pull mechanism to exchange and shuffle local membership information (node cache). At each node, the protocol contains a local cache Q of node identifiers ($|Q| \leq q_{max}$), where q_{max} is the maximum local cache size (this parameter is applied to all membership protocols used in this work). At each cycle, the local cache is sent in a push message to a node randomly selected from the local cache. When a push message is received, the local cache is sent in a reply (pull message) to the remote node originating the push message. The local cache is merged with the remote cache and the remote node ID (refreshing mechanism). The local cache is finally trimmed by randomly eliminating the number of entries

exceeding q_{max}. In the Node Cache Protocol, the trimming operation is the component that may cause connectivity problems, because the removed entries could be the single link between two connected components in an overlay topology with weak connectivity. The Node Cache Protocol does not provide an explicit mechanism to remove broken links.

Cyclon [19] is a membership protocol that is an enhanced version of a basic node cache shuffling. The mechanism of Cyclon is similar to the Node Cache Protocol, which also adopts a push-pull mechanism. In Cyclon, cache entries are assigned an age attribute to limit their lifetime. At each cycle, a number of entries randomly selected from the local cache are sent (push message) to the node corresponding to the oldest entry in the local cache. When a push message is received, the node replies with a pull message containing a number of randomly selected entries from its local cache. The received entries are used to replace the donated entries at both ends. Connectivity problems in Cyclon may arise when there is message interleaving between independent pairs of push-pull exchanges involving the same node. Message interleaving has been identified as a potential threat to the accuracy of those Epidemic aggregation protocols [21] that would require the push-pull operation to be atomic. Similarly, message interleaving in Cyclon introduces the risk of removing critical cache entries, as in distributed asynchronous systems the atomicity of the push-pull operation for the cache exchange is not guaranteed. In Cyclon, the broken link removal mechanism is a separate process, which adopts a maximum cache entry lifetime and a message timeout to detect failed nodes and to remove the broken links.

Eddy [20] is arguably the most complex membership protocol. In order to provide a better random distribution of node samples in the system, Eddy tries to minimize temporal and spatial dependencies between local caches. The mechanism in Eddy can be separated into two independent processes: cache mixing (gossiping) and cache item refreshing. Cache mixing is based on a symmetric push-pull operation: some local cache entries are removed and sent to a random peer for an exchange. Item refreshing adopts a maximum cache entry lifetime and a push-forwarding mechanism. Expired entries are replaced with fresh entries: when an item in the local cache expires, the node generates a new item with its own identifier and sends it to a random node within two hops. The cache item refreshing procedure in Eddy provides an effective mechanism for removing invalid cache entries generated by node churn and node failures. However the shuffling mechanism introduces a problem when an invalid link is selected for an

exchange and in early timeouts. A message timeout is raised when a pull message does not arrive within the expected time interval: the list of entries that were sent in the push message are reinserted in the local cache. However, if the pull message arrives after the timeout, the remote entries in the pull message are discarded. The refreshing process of Eddy is effective, however it adopts an entry removal mechanism that can cause connectivity problems and introduces a significant communication overhead.

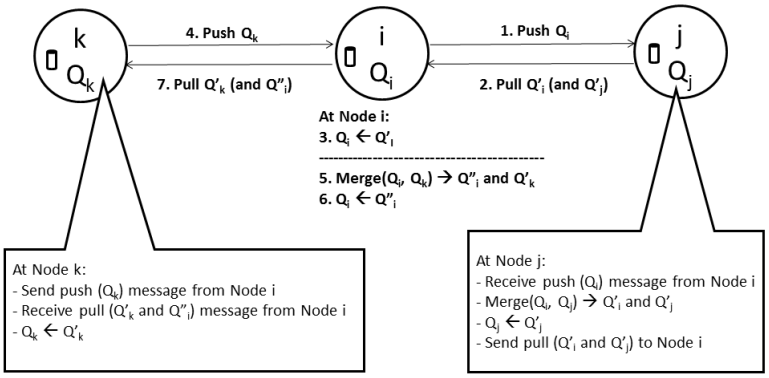
*2.1. The Expander Membership Protocol (*EMP*)*

Memberships protocols can be considered as a distributed implementation of multiple random walks and aim to transform any connected overlay topology into a sparse random graph, which provides strong connectivity and high robustness to the system. *EMP* ([24]) is designed with these considerations and is based on the concept of expander graphs.

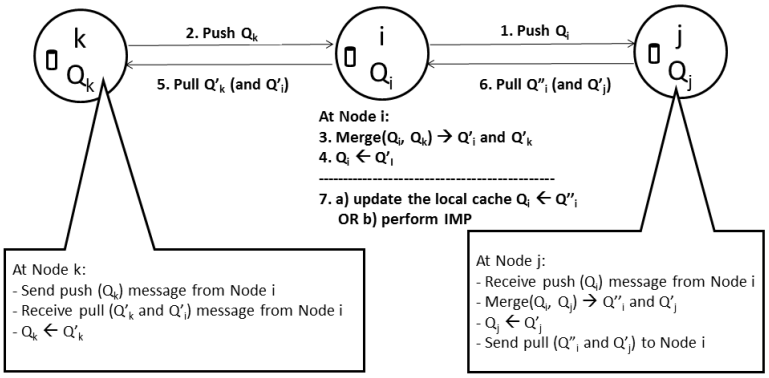
Expander graphs, or simply expanders, are sparse graphs that have strong connectivity properties. A graph is described as an expander when any vertex subset (not too large) has a relatively large set of one-hop distant neighbours. A few alternative definitions of the graph expansion property exist: the vertex expansion, the edge expansion or the spectral gap. In this work, the definition of expanders based on the vertex expansion is used and the minimum vertex expansion index, as defined in [24], has been adopted.

EMP adopts a symmetric push-pull mechanism with a push-forwarding mechanism (quasi-random gossiping). At each Gossip cycle and at each node i, a destination node x_0 is randomly chosen from the local cache Q_i. A push message m_i containing the local cache Q_i is sent to x_0. When node x_0 receives the push message, it computes a measure of neighbourhood similarity ($Q_{x0} \cap Q_i$) between the local cache Q_{x0} and the remote cache Q_i. The message is accepted if there is low similarity, that is, if $|Q_{x_0} \cup Q_i \cup \{i\}| \geq 2 \cdot q_{max}$, where q_{max} is a maximum local cache size and $\{i\}$ is introduced as refreshing mechanism. If the push message is accepted, the two caches are merged to generate two disjoint partitions and a pull message is sent to node i with one of the two partitions. If the push message is not accepted, the message m_i is forwarded to a node x_1 randomly selected from the local cache of node x_0. This procedure is repeated up to a maximum number of hops (h_{max}). If no node along the random walk of the message m_i exactly matches the criterion, the message is finally sent to the visited node with the minimal neighbourhood similarity.

The aim of this random walk is to maximize the expansion quality of the overlay topology by exchanging and merging membership information



(a) without message interleaving



(b) with message interleaving

Figure 1: The scenarios of message transmission with and without message interleaving

between nodes with low neighbourhood similarity. Hence, clusters of nodes with high neighbourhood similarity are expected to break up sooner than in the 1-hop push-pull scheme used in most protocols.

3. Message Interleaving

In real-world distributed systems, applications have often to cope with asynchronous communication and network latency. As a result of these, there is the possibility that a node accepts a pull message whilst it is waiting for a pull message. This event is referred to as message interleaving. In weakly connected overlay topologies, message interleaving can lead to the loss of the global connectivity [25].

To describe the effect of message interleaving, it is better to start with comparing two message transmission scenarios: with and without message interleaving. The first scenario (figure 1(a)) considers three nodes (i, j and k), which are exchanging their membership information without interleaving. First, node i sends a push message to node j; then node i receives a pull message from node j. Eventually node k sends a push message to node i and, finally, node i sends a pull message to node k. In this scenario, the two independent push-pull operations happen in the expected sequence without message interleaving.

Let Q_i be the local cache at node i, the sequence of events at node i are as follows.

1. Node i sends a push message (Q_i) to node j.
2. Node i receives a pull message (Q'_i and Q'_j) from node j.
3. Node i updates the local cache $Q_i \leftarrow Q'_i$.
4. Node i receives a push message (Q_k) from node k.
5. Node i merges Q'_i and Q_k and generates two partitions Q''_i and Q'_k.
6. Node i updates the local cache $Q'_i \leftarrow Q''_i$.
7. Node i sends a pull message (Q'_k and Q''_i) to node k.

The second scenario (figure 1(b)) is similar to the first scenario, except that the two push-pull operations are overlapped and message interleaving occurs. Node i sends a push message to node j. Before node i can receive a pull message from node j, it accepts a push message from node k. The sequence of events at node i are as follows.

1. Node i sends a push message (Q_i) to node j.
2. Node i receives a push message (Q_k) from node k.
3. Node i merges Q_i and Q_k and generates two partitions Q'_i and Q'_k.
4. Node i updates the local cache $Q_i \leftarrow Q'_i$.
5. Node i sends a pull message (Q'_i and Q'_k) to node k.
6. Node i receives a pull message (Q''_i and Q'_j) from node j.
7.
   (a) Node i updates the local cache $Q_i \leftarrow Q''_i$.
   OR
   (b) Node i detects message interleaving and performs IMP.

After step 6 the three local caches have been updated (Q'_i, Q'_k, Q'_j) and an additional buffer (Q''_i) has been received at node i. In step 7, if node i

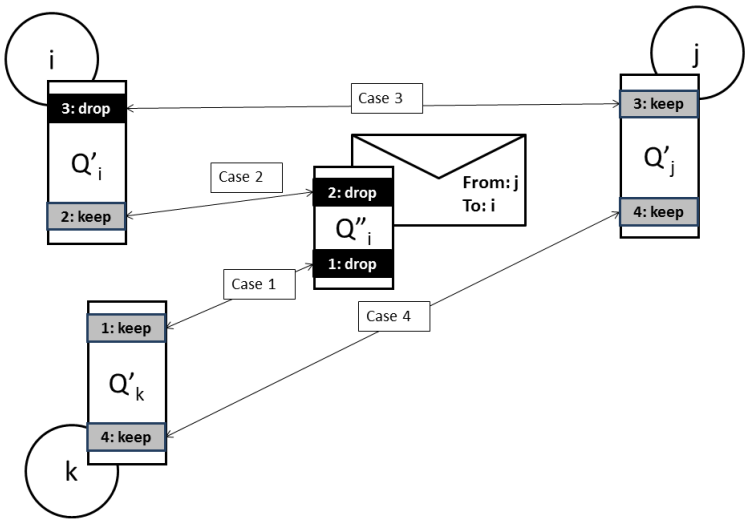


Figure 2: The snapshot of cache configuration when node i detects a message interleaving event

would perform the simple update operation of 7(a), as in the case without message interleaving, there would be a risk of removing critical links, thus exposing the system to potential connectivity problems. Alternatively, node i can actually detect the message interleaving event in 7(b) and a more complex operation, the Interleaving Management Procedure (IMP), can be performed to process the incoming pull message.

Figure 2 is a snapshot of the cache configuration after step 6, when node i detects the message interleaving event. The problem in this configuration is that some duplicated cache entries have been generated by the two merging operations. The total number of cache entries in the whole system is bounded by $N \cdot q_{max}$, where N is the number of nodes. When a cache duplicate is generated, another cache entry needs to be discarded in order to accommodate the duplicate. Duplicates negatively affect the node outdegree distribution in the system and introduce the risk that critical links are arbitrarily removed.

Duplicated entries must be detected and eliminated when possible. In this case, the number of duplicates introduced in the system is bounded by the number of entries in Q''_i. The next section describes the procedure used to detect and discard duplicated cache entries.

3.1. Detection of Cache Duplicates

The aim of this procedure is to identify and eliminate the duplicated entries generated by the message interleaving event. Let us assume that the initial local caches (Q_i, Q_j and Q_k) of the three nodes in the message interleaving scenario do not share any entry, that is $Q_i \cap Q_j = \varnothing$, $Q_i \cap Q_k = \varnothing$ and $Q_j \cap Q_k = \varnothing$. The duplicates have been generated because the push-pull operations between i, j and k have not been performed atomically: the merge and partition operation on Q_i and Q_j has generated Q'_j and Q''_i at node j, while node i has changed its local cache. Figure 2 shows the possible sets of duplicated entries in the four caches of the scenario when node i detects the interleaving event (after step 6).

In order to detect and remove duplicated entries, node i requires the information of the four partitions generated by the two merging operations: Q'_i and Q'_k are locally available at node i; Q''_i and Q'_j can both be received by i in the pull message from node j.

The possible sets of duplicates in these four buffers correspond to $\binom{4}{2} = 6$ possible combinations. Two combinations cannot generate duplicates, as $Q'_i \cap Q'_k = \varnothing$, $Q'_j \cap Q''_i = \varnothing$. Four sets of potential duplicates can be identified and are shown as four cases in figure 2: 1) $Q''_i \cap Q'_k$, 2) $Q'_i \cap Q''_i$, 3) $Q'_i \cap Q'_j$, 4) $Q'_k \cap Q'_j$.

In the figure some buffer subsets have been indicated as 'drop' and others as 'keep': node i can identify and drop the duplicates of three cases. However, it is not possible for node i to remove all of the duplicates, because node i cannot remove the duplicates of case 4, which are located in node j and node k.

A mechanism (IMP) to manage the effect of message interleaving and remove most of duplicates has been designed and is based on this analysis. This procedure can reduce the likelihood of connectivity problems, but cannot eliminate the risk completely (case 4). For this reason, a further mechanism for connectivity recovery is also required.

4. The Enhanced Expander Membership Protocol

The previous section presented the details of how to detected and eliminate duplicates in order to reduce the degradation of the overlay topology and the risk of connectivity problems due to message interleaving events. However, the problem has not been eliminated completely and, moreover, as discussed in section 2, there are other mechanisms in membership protocols which can also affect the quality of the overlay topology.

In graphs with strong connectivity, e.g. a random graph, these problems may not occur as these topologies are robust by definition. But, in weakly connected overlay topologies, e.g. a ring of communities [24], discarding even a few critical cache entries may cause the loss of global connectivity.

For this reason *EMP+*, an enhancement of *EMP*, is obtained with the introduction of two procedures into *EMP* the Interleave Management Procedure (IMP) and a connectivity recovery mechanism. The goal of *EMP+* is maintaining a global connectivity, while still supporting a good convergence speed of the applications. This section explains the details of the algorithm including the cache entries shuffling mechanism, IMP, the connectivity recovery mechanism and the broken link removal.

i	a node in the network, $i \in V$, where V is the set of nodes
Q_i	main cache at node i, $\|Q_i\| \leq q_{max}$
q_{max}	maximum main cache size
R_i	reserve cache at node i, $\|R_i\| \leq r_{max}$
r_{max}	maximum reserve cache size
H_i	history cache at node i
hl	history cache item lifetime (number of cycles)
h_{max}	maximum number of hops in random walks
$m_{\rightarrow}$	push message with payload: - s, node originating the push - Q_s, main cache at s - d, current best destination node - v_d , current minimum overlap - h, hop count
$m_{\leftarrow}$	pull message with payload: - d, node originating the pull - Q, set of donated cache entries - Q_d, main cache at node d

Table 1: Notation adopted in the *EMP+* pseudocode

Table 1 describes the notation adopted in the following algorithms. Algorithm 1 shows the pseudocode of *EMP+*. The fundamental mechanism of *EMP+* is based on *EMP* [24], which is also reviewed in section 2.1.

The connectivity recovery method is based on a reserve cache R_i, which is used to store the entries that are removed during the merging operation. To avoid that the size of the reserve cache may grow indefinitely, a maximum reserve cache size (r_{max}) is enforced.

A storage of the entries donated with pull messages is required and a history cache H_i is introduced for this purpose (line 16). In this cache, entries are associated with a maximum lifetime to avoid that this cache may grow indefinitely.

Lines 10 and 11 describe how entries from the reserve cache are used

Algorithm 1 EMP+

```
1:  procedure SendPushMessage
2:      remove expired entries from H_i
3:      j ← get the oldest node from Q_i
4:      send a push message to j :m→(s = i, Q_s = Q_i, d = null, v_d = ∞, h = 0)
5:
6:  procedure ReceivePushMessage( message m→)
7:      compute total cache size v = |Q_i ∪ m→.Q_s ∪ R_i|
8:      if (v + 1 ≥ 2 * q_max) or (m→.h > h_max) then
9:          Q_m ← Q_i ∪ m→.Q_s
10:         while (|Q_m| + 1 < 2 * q_max) and (|R_i| > 0) do
11:             insert entry from R_i into Q_m
12:         while (|Q_m| + 1 < 2 * q_max) do
13:             insert random entry into Q_m
14:         randomly partition Q_m into Q_1 and Q_2(|Q_1 ∪ {m→.s}| = |Q_2| = q_max)
15:         update local main cache: Q_i ← Q_1 ∪ {m→.s}
16:         add the donated cache entries to the history cache: H_i ← H_i ∪ Q_2
17:         send a pull message to m→.s :m←(d = i, Q = Q_2, Q_d = Q_i)
18:     else if (m→.h < h_max) then
19:         if v < m→.v_d then
20:             set m→.d = i and m→.v_d = v
21:         select random node j from Q_i
22:         m→.h + +
23:         send m→ to j
24:     else if (m→.h == h_max) then
25:         m→.h + +
26:         send m→ to m→.d
27:
28: procedure ReceivePullMessage( message m←)
29:     if message interleaving == false then
30:         update local main cache: Q_i ← m←.Q
31:     else
32:         perform IMP(m←)
33:     while |R_i| > r_max do
34:         remove the oldest entry in R_i
```

during the merge operation to generate the two disjoint partitions Q_1 and Q_2. Lines from 28 to 34 describe the procedure to process incoming pull messages with and without message interleaving. If there is no message interleaving, the local cache can be immediately updated with the content of the message. When message interleaving occurs, the procedure IMP is performed.

Algorithm 2 shows the pseudocode of IMP, which performs the removal of the duplicated entries. Lines from 2 to 7 show how the duplicates from the cases 1, 2 and 3 are detected and discarded. IMP is the only procedure in *EMP+* that inserts entries into the reserve cache as shows in line 10.

The broken link removal process is embedded into two mechanisms of *EMP+*: the shuffling operation and IMP. In the shuffling operation (Algorithm 1), the item lifetime is used to select the oldest entry in the main

Algorithm 2 IMP

```
1:  procedure IMP( message m_←)
2:      D_1 ← m_←.Q ∩ H_i, detect and remove the duplicates from case 1
3:      m_←.Q ← m_←.Q \ D_1
4:      D_2 ← m_←.Q ∩ Q_i, detect and remove the duplicates from case 2
5:      m_←.Q ← m_←.Q \ D_2
6:      D_3 ← Q_i ∩ m_←.Q_d, detect and remove the duplicates from case 3
7:      Q_i ← Q_i \ D_3
8:      T ← Q_i ∪ m_←.Q, create a temporary cache
9:      while |T| > q_max do
10:         remove the oldest entry in T and add it to R_i
11:     while |T| < q_max and |R_i| > 0 do
12:         remove the oldest entry in R_i and add it to T
13:     while |T| < q_max do
14:         select a random entry from the duplicates and add it to T
15:     update local main cache: Q_i ← T
```

cache (line 3) for the push operation. The selected entry is flagged and the push operation also serves as a failure detection mechanism for the removal of the broken links. When a push message is sent, if a source node does not receive the pull message within an expected time interval, it then assumes that the peer node has failed or has left the system. Early timeouts may occur and can be tolerated because of the refreshing mechanism.

Another mechanism to remove broken links is provided by IMP. It has already been mentioned that the main goal of IMP is to protect the global connectivity from the negative effect of message interleaving events. However, a secondary effect is that the number of broken links is reduced: in line 10 of Algorithm 2 older entries of the main cache are moved to the reserve cache. Some broken links in the reserve cache may return into the main cache, nevertheless the shuffling mechanism can select and discard them.

5. Experimental Analysis

The experimental analysis aims at evaluating *EMP+* and comparing it against other membership protocols by means of simulations. The simulations have been carried out in PeerSim [26], a Peer-to-Peer Java-based network simulator based on discrete events. Five membership protocols (Table 2) have been included in the simulations, where their parameters have been set for best performance according to the references and a preliminary analysis. In particular, the effects of the *EMP+* parameters have been discussed in details in section 5.4.

An ideal random membership protocol has also been included to provide a benchmark performance. The protocol generates a random regular graph

at each iteration based on the global knowledge of the system.

Membership Protocol	Parameters
Node Cache Protocol [21]	$q_{max} = 30$
Cyclon [19]	$q_{max} = 30$
Eddy [20]	shuffle length= 6
EMP [24]	$q_{max} = 30$ and $h_{max} = 5$
EMP+	$q_{max} = 30$, $h_{max} = 5$, $r_{max} = 100$ and $hl = 2$
Ideal random membership protocol	$q_{max} = 30$

Table 2: Membership protocols and their parameters

The simulations are based on an asynchronous network model and a Gossip cycle structure similar to [21] with random clock offsets, randomised initiation of push messages and uniform distribution of network latency. It is assumed the physical network topology is connected and a lossless routing protocol (e.g., TCP) is available. Three different initial overlay topologies have been used: a regular ring lattice [24], a ring of communities [24] and a regular random graph. In each test the initial overlay topology, the refresh rate of Eddy and the shuffle length of Cyclon have been set according to the aim of the specific test and are reported.

The tests have been grouped in four sets and presented in the following sections. First the convergence speed of an Epidemic application is investigated. In particular, the tests intend to show how the different membership protocols may affect the convergence speed of a data aggregation task. The second group of simulations is used to investigate how the different protocols may preserve or loose the global connectivity of the overlay topology. In the third group of simulations the effect of node failures on a random overlay topology is analysed. This analysis is important because it shows how membership protocols can recover the structure of the overlay topology from the degradation generated by node churn and failures. Finally, a set of simulations is used to study the importance of the *EMP+* parameters and their effect on its performance.

5.1. Application Convergence Speed

This section shows how membership protocols can affect the convergence speed of a global aggregation task when the overlay topology is not an ideal random graph.

Each node holds a local value and requires to compute the global average of the distributed set of values. A peak distribution of the distributed values is used: the peak value is set equal to the network size and the target

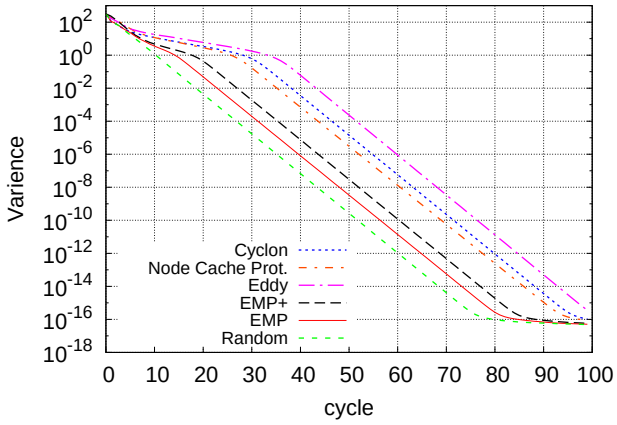


Figure 3: Convergence speed of Epidemic aggregation (initial overlay: regular ring lattice; network size: 100000)

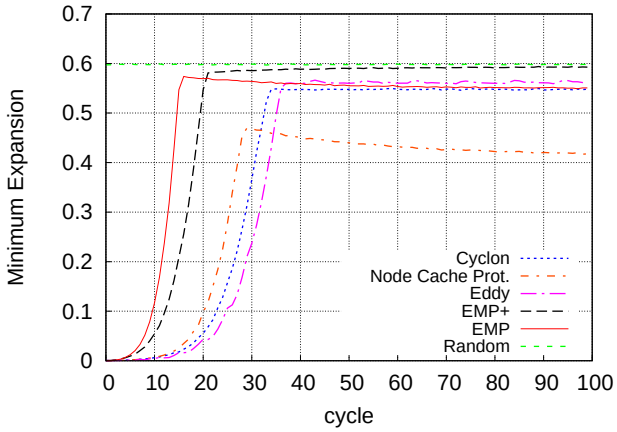


Figure 4: Minimum expansion quality of the graph (initial overlay: regular ring lattice; network size: 100000)

global average is 1. The aggregation task is performed by means of an Epidemic aggregation protocol, the Symmetric Push Sum Protocol (SPSP) [12]. SPSP is used at each node to estimate the global average and the variance of the estimated values in the network is used to determine the speed of convergence of the application.

A random graph would be ideal to support a fast information propagation in the network. The speed of rewiring from the initial overlay topology into a random graph affects the convergence speed of the aggregation task. The initial topology is set as a regular ring lattice, which is far from random.

In this test Eddy refreshing mechanism is executed every 10 cycles and the shuffle length of Cyclon is set to 24 for a faster graph rewiring.

The ideal random membership protocol is the only protocol that can use a random overlay topology from the first cycle and is used to provide an ideal performance.

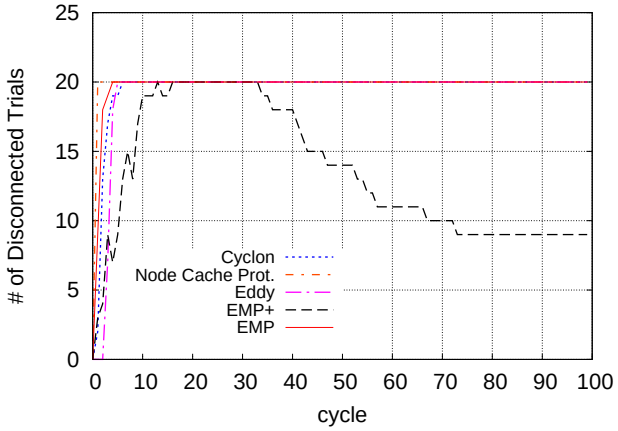
Figure 3 show the variance of the estimated aggregation values in the network over time. A curve with a constant slope in the chart indicates an exponential convergence to the target value. *EMP* and *EMP+* are the protocols that provide the closest convergence speed to the ideal slope.

At the beginning of the simulation the overlay topology is far from random and the membership protocols are expected to rewire the graph in such a way to quickly converge to a random graph. Obviously different membership protocols perform this operation at different rates. The minimum expansion index [23, 24] of the overlay topology can be used to measure directly the ability of a protocol to rewire the overlay topology. For the same simulation settings of figure 3 the minimum expansion index is shown in figure 4. The horizontal line at 0.6 of the ideal random protocol provides the target for the performance of the other protocols. The other protocols start with a very poor expansion quality (index close to 0) and gradually improve it while they rewire the graph. The results of the figures 3 and 4 show that the speed of graph rewiring and the convergence speed of an Epidemic aggregation have a clear correlation. *EMP* is the fastest membership protocol that can transform the overlay topology. Nevertheless, *EMP+* is able to reach the best expansion quality and is the most similar to the ideal random protocol.

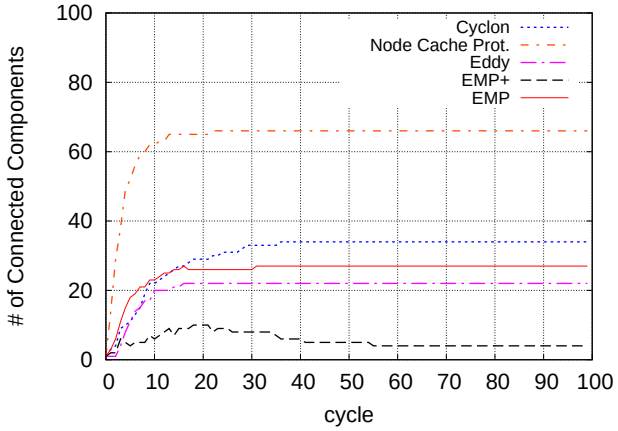
5.2. Global Connectivity

In this set of simulations, each protocol was run for 100 Gossip cycles and each simulation was repeated for 20 times with different seeds of the random number generator. The initial topology is a ring of communities which presents a strong local connectivity and a weak global connectivity. The topology was generated with 100 well connected communities of 1000 nodes. The refresh rate of Eddy is set to 10 cycles. The shuffle length of Cyclon is set to 12 for reducing the chance of removing critical entries.

Figure 5(a) shows the number of trials in which each protocol has lost the global connectivity over time. In all the 20 trials all the protocols permanently lost the global connectivity with the exception of *EMP+*. The chart shows that *EMP+* initially lost the connectivity in all the trials too, but it was able to recover it in 11 trials.



(a) Number of trials in which the overlay graph is disconnected



(b) Maximum number of connected components

Protocol	*min*	*max*	*avg*	*stddev*
Node Cache Prot.	46	66	53.50	5.08
Cyclon	9	34	21.85	5.57
Eddy	6	22	14.80	4.12
EMP	12	27	20.35	4.63
EMP+	1	5	2.60	1.31

(c) minimum, maximum, average and standard deviation of the number of connected components at cycle 50

Figure 5: Global connectivity test over 20 trials (initial overlay: ring of communities; network size: 100000)

Epidemic protocols require a single connected component in order to work correctly. However, the level of topology disaggregation each protocol determines is also an indicative measure of robustness. Figure 5(b) shows the maximum number of connected components over the 20 trials. The simplest protocol, the node cache protocol, induces the worst level of disaggregation. *EMP*, Cyclon and Eddy present a similar behaviour: their curves are monotonic in the sense that once a connected component is isolated from the rest of the network, connectivity to the rest of the system cannot be recovered. In *EMP+*, initially the maximum number of connected components also grows, but it is then reduced to 4 in the second part of the time interval. *EMP+* has the smallest maximum number of connected components and its non-monotonic curve indicates the ability to recover from the loss of global connectivity.

Additional statistics have also been analysed: the minimum and the average follow a similar trend over time. The table in figure 5(c) shows the additional statistics at cycle 50, at which the protocols have reached their steady state.

5.3. Node Failures

The aim of this group of simulations is to observe the performance of the membership protocols when node failures occur. In this case the initial overlay topology is a random regular graph and at cycle #60 20% of the nodes fail. A fail-stop model has been used: the failed nodes completely stop working without notice. The failed nodes introduce a significant perturbation in the overlay topology: all the cache entries referring to the failed node are suddenly invalid (broken links). In this case the protocols are expected to remove the broken links and to replace them with valid entries as quickly as possible.

In these simulations the refresh rate of Eddy is set to 5 cycles and the shuffle length of Cyclon is set to 12.

The numbers of broken links over time in the system is shown in figure 6. The membership protocol that is able to remove the broken links faster is Eddy, followed by *EMP+*, *EMP* and Cyclon. The Node Cache Protocol is the worst protocol, which increases the number of broken links.

It is interesting to observe the minimum expansion index for this set of simulations in figure 7. After the node failures, the expansion quality of the protocols suddenly drops. The first observation is that Eddy is the protocol with the largest expansion quality degradation. There are only three protocols, namely *EMP+*, *EMP* and Eddy, that have the ability to

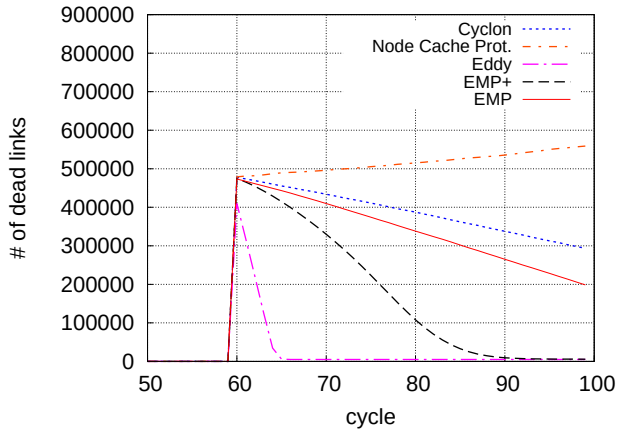


Figure 6: Number of broken links in the system due to node failures (initial overlay: random regular graph; network size: 100000)

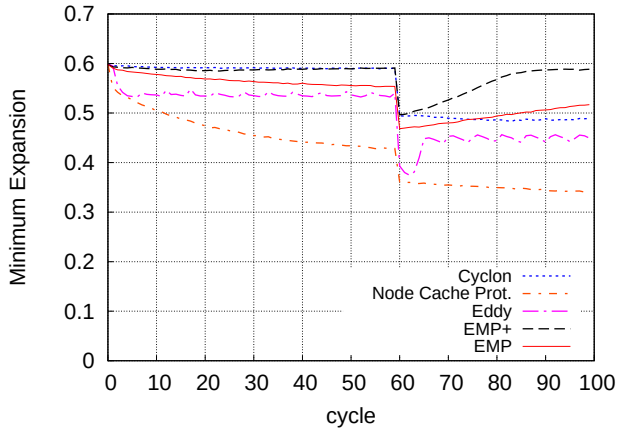


Figure 7: Minimum expansion quality of the graph with node failures (initial overlay: random regular graph; network size: 100000)

recover the expansion quality. *EMP+* is the only protocol that can fully recover to the quality of the initial random topology within the maximum simulation time.

The relation between the speed of the broken link removal and the minimum expansion index can be observed. The number of cycles that the protocols require to remove broken links is similar to the number of cycles that they use to recover the expansion quality.

Although Eddy provides the fastest mechanism to remove broken links, it is not so efficient in term of expansion quality and communication overhead.

Eddy's refreshing mechanism efficiently make use of the item lifespan (or refresh rate) to clean old entries and insert fresh entries into the system. The majority of the broken links is cleared by Eddy within 5 cycles, which is the refresh rate used in this case. However, the message overhead associated to Eddy's refreshing mechanism is very high. Regardless of node failures, the total number of messages that Eddy spends over 100 cycles, is more than 130 million. *EMP+* and *EMP* employ, respectively, about 21 million and 27 million messages. The membership protocols who adopt a simple push-pull gossiping, Cyclon and the Node Cache Protocol use exactly 20 million messages as expected.

5.4. EMP+ *Parameters*

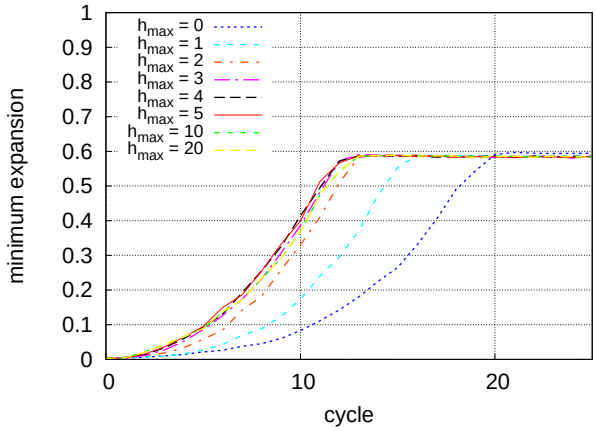
This section provides an analysis of the effect of the important *EMP+* parameters. The following four parameters are employed in *EMP+*:

1. the maximum main cache size (q_{max}),
2. the maximum hop count (h_{max}),
3. the maximum reserve cache size (r_{max}) and
4. the item lifetime in the history cache (hl).

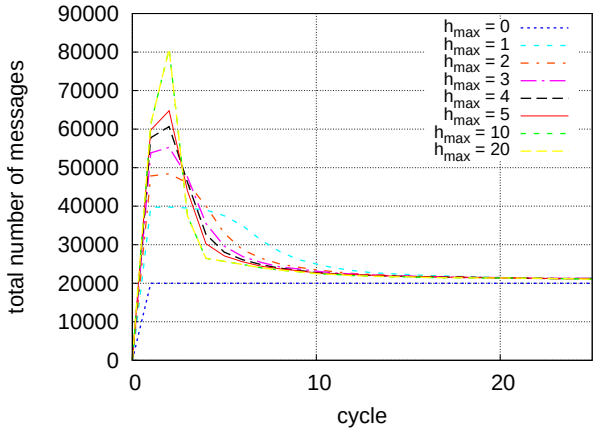
The first parameter q_{max} is shared by most Epidemic membership protocols and is used to limit the local view of the global system at each node. In this section we investigate the effect of the other three parameters which are specific to *EMP+*.

The maximum hop count h_{max} limits the number of times that a push message can be forwarded in the random walk. The forwarding mechanism is meant to improve the speed of graph rewiring at the cost of additional communication. This parameter controls the trade off between the graph rewiring speed and the extra communication overhead for the forwarding mechanism.

The speed of graph rewiring can be measured by means of the minimum expansion index. The results shown in Figure 8(a) were obtained by tests run on a network of 10000 nodes with an initial regular ring lattice topology and for various values of the parameter h_{max}. For $h_{max} = 0$ no forwarding mechanism is used and the push message is always accepted at the first destination without considering cache similarity. In this case, the index reaches the ideal value of a random graph after 20 cycles. Any positive value of h_{max} provides an improvement in the convergence speed with a clear saturation for values greater than 5, which is the value used in all the other simulations. This corresponds to a long enough random walk to find



(a) Minimum expansion index



(b) Communication overhead

Figure 8: Effect of the maximum hop count h_{max} (initial overlay: regular ring lattice; network size: 10000)

the node that provides the fastest improvement of the expansion quality of the overlay topology.

Figure 8(b) show the effect of the parameter h_{max} on the communication overhead in terms of the number of messages (send operations) for various h_{max} values. As expected, longer random walks require more messages. The extra communication overhead is used to converge towards a random overlay topology: the more messages are sent, the faster the overlay topology is rewired. It is interesting to notice how the extra overhead is greater in the initial phase when the rewiring process is most needed. After the rewiring

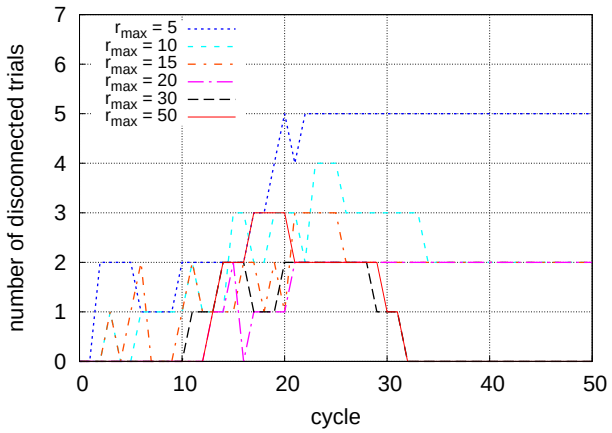


Figure 9: Cases with loss of global connectivity for different values of the maximum reserve cache size r_{max} (initial overlay: ring of communities; network size: 10000)

is complete, every case produces a similar communication overhead because the overlay topology has already been transformed into a random graph, which means that a push message has a high likelihood of being accepted at the first hop similarly to the simple push-pull approach with no forwarding mechanism ($h_{max} = 0$). For this reason choosing the h_{max} value according only to the criterion of faster convergence (figure 8(a) is appropriate, since the extra communication overhead is only temporary.

The maximum reserve cache size (r_{max}) affects the connectivity robustness of the protocol. This cache holds cache items which are removed from the main cache and can later be reused. The reserve cache items can help to prevent and to fix the loss of global connectivity in weakly connected graphs. The size of this cache is prevented to grow indefinitely by r_{max}. However, too small values of r_{max} may have a detrimental effect: a critical entry may be removed prematurely from the reserve cache. The evaluation of the effect of this parameter is carried out in a topology with a ring of communities and a network size of 10000. Each case with a different r_{max} value is tested 20 times with different seeds of the pseudo-random number generator. Figure 9 shows the number of trials in which the global connectivity was lost. Cases with larger r_{max} values have less chance of loosing the connectivity. When the parameter is equal to or larger than 30, the protocol can always maintain the global connectivity in all tested trials. However, setting the correct parameter must also consider the network size: a larger maximum reserve cache size is required for a larger network size. For the

sake of brevity, this analysis has not been included.

Finally, the lifetime (hl) of the history cache items is the number of cycles after which items can be removed from the history cache. It is used to limit the size of the history cache: it is sufficient that items stay in this cache for a time sufficiently long for the reply (pull) to a push message to be received. Considering a network of 10000 nodes, at steady state for $hl = 1$ the average size of the history cache is 81 and at each unit increment of the parameter, the average size increases by about 30, which corresponds to the size of the main cache (or the payload of a cache message). The value of this parameter does not affect the performance of the protocol as far as it is greater than twice the expected maximum propagation delay in the network. For this reason a more detailed analysis for this parameter is not reported.

6. Conclusions

This work has investigated the effect of a dynamic network environment on the performance of Epidemic membership protocols. The main task of these protocols is to support fast Epidemic data aggregation and information dissemination by providing the fundamental network service of peer sampling. The internal mechanisms of membership protocols transform any overlay topology into a random graph by continuously and randomly rewiring the graph edges. In dynamic environments, node churn and message interleaving events can be very frequent and can deteriorate the efficiency of Epidemic applications. Most importantly, in graphs with weak global connectivity, it has been show that the cache shuffling mechanism that all protocols adopt, can cause a permanent loss of global connectivity.

In this work, the problems in dynamic environments and the drawbacks of the previous membership protocols have been identified and discussed. The main contribution of this work is an Enhanced Expander Membership Protocol (*EMP+*), which is suitable for dynamic network conditions. The design of the protocol is based on the mathematical concept of expander graph and on the considerations of the effect of message interleaving events and connectivity problems which may occur in asynchronous and dynamic environments.

EMP+ has been evaluated by means of several simulations and compared to other membership protocols. The results show that *EMP+* is superior to the other protocols in every performance index. Moreover, *EMP+*

is the only protocol that has the ability to recover the global connectivity when lost.

Future work will focus on other network aspects related to real-world applications, e.g. firewalls, overlay graph initialisation, type of communication and security issues. Another important aspect to consider is the most appropriate procedure to let nodes join during the execution. Finally, more experimental analysis is required to evaluate the protocols in a range of realistic churn scenarios.

References

[1] N. Bansod, A. Malgi, B. K. Choi, J. Mayo, Muon: Epidemic based mutual anonymity in unstructured P2P networks, Computer Networks 52 (5) (2008) 915–934.

[2] B. Ghit, F. Pop, V. Cristea, Epidemic-style global load monitoring in large-scale overlay networks, in: P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on, 2010, pp. 393–398.

[3] S. Tang, E. Jaho, I. Stavrakakis, I. Koukoutsidis, P. Van Mieghem, Modeling gossip-based content dissemination and search in distributed networking, Comput. Commun. 34 (2011) 765–779.

[4] D. H. H. Sheng Di, Cho-Li Wang, Gossip-based dynamic load balancing in an autonomous desktop grid, in: Proc. of the 10th International Conference on High-Performance Computing in Asia-Pacific Region, 2009, pp. 85–92.

[5] Y. Ma, A. Jamalipour, An epidemic P2P content search mechanism for intermittently connected mobile ad hoc networks, in: IEEE GLOBECOM, 2009, pp. 1–6.

[6] L. Chitnis, A. Dobra, S. Ranka, Aggregation methods for large-scale sensor networks, ACM Transactions on Sensor Networks (TOSN) 4 (2008) 1–36.

[7] N. Marechal, J.-M. Gorce, J. Pierrot, Joint estimation and gossip averaging for sensor network applications, IEEE Transactions on Automatic Control 55 (5) (2010) 1208–1213.

[8] A. Dimakis, S. Kar, J. Moura, M. Rabbat, A. Scaglione, Gossip algorithms for distributed signal processing, Proceedings of the IEEE 98 (11) (2010) 1847–1864.

[9] C. Ragusa, A. Liotta, G. Pavlou, An adaptive clustering approach for the management of dynamic systems, Selected Areas in Communications, IEEE Journal on 23 (12) (2005) 2223–2235.

[10] S. Galzarano, C. Savaglio, A. Liotta, G. Fortino, Gossiping-based aodv for wireless sensor networks, in: Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on, 2013, pp. 26–31.

[11] R. Van Renesse, Y. Minsky, M. Heyden, A gossip-style failure detection service, in: Middlewarte '98 Proceeding of the IFIP International Conference on Distributed Systems Plateforms and Open Distributed Processing, 2009 International Federation for Information Processing on, 2009, pp. 55–70.

[12] G. Di Fatta, F. Blasa, S. Cafiero, G. Fortino, Epidemic k-means clustering, in: Proc. of the IEEE 11th Int.l Conference on Data Mining Workshops, 2011, pp. 151–158.

[13] G. Di Fatta, F. Blasa, S. Cafiero, G. Fortino, Fault tolerant decentralised k-means clustering for asynchronous large-scale networks, Journal of Parallel and Distributed Computing 73 (3) (2013) 317–329.

[14] H. Mashayekhi, J. Habibi, T. Khalafbeigi, S. Voulgaris, M. van Steen, GDCluster: A general decentralized clustering algorithm, IEEE Transactions on Knowledge and Data Engineering 27 (7) (2015) 1892–1905.

[15] H. Strakov, G. Niederbrucker, W. N. Gansterer, Fault tolerance properties of gossip-based distributed orthogonal iteration methods, Procedia Computer Science 18 (0) (2013) 189 – 198, 2013 International Conference on Computational Science.

[16] P. Soltero, P. Bridges, D. Arnold, M. Lang, A gossip-based approach to exascale system services, in: Proceedings of the 3rd International Workshop on Runtime and Operating Systems for Supercomputers, ROSS '13, ACM, 2013, pp. 3:1–3:7.

[17] A. Katti, G. Di Fatta, T. Naughton, C. Engelmann, Scalable and fault tolerant failure detection and consensus, in: Proceedings of the 22Nd European MPI Users' Group Meeting, EuroMPI '15, ACM, 2015, pp. 13:1–13:9.

[18] M. Jelasity, S. Voulgaris, R. Guerraoui, A. M. Kermarrec, M. van Steen, Gossip-based peer sampling, ACM Trans. Comput. Syst., 25(3).

[19] S. Voulgaris, D. Gavidia, M. van Steen, Cyclon: Inexpensive membership management for unstructured p2p overlays, Journal of Network and Systems Management 13 (2) (2005) 197–217.

[20] E. Ogston, S. A. Jarvis, Peer sampling with improved accuracy, Peer-to-Peer Networking and Applications 2 (1) (2008) 24–36.

[21] F. Blasa, S. Cafiero, G. Fortino, G. Di Fatta, Symmetric push-sum protocol for decentralised aggregation, in: Proc. of the Int.l Conf. on Advances in P2P Systems, 2011, pp. 27–32.

[22] S. Hoory, N. Linial, A. Wigderson, Expander graphs and their applications, Bullettin of the American Mathematical Society 43 (4) (2006) 439–561.

[23] P. Poonpakdee, G. Di Fatta, Expansion quality of epidemic protocols, in: International Symposium on Intelligent Distributed Computing (IDC) VIII, Vol. 570 of Studies in Computational Intelligence, Springer, 2014, pp. 291–300.

[24] P. Poonpakdee, G. Di Fatta, Expander graph quality optimisation in randomised communication, in: Proc. of the 2014 IEEE International Conference on Data Mining Workshop (ICDMW), IEEE, 2014, pp. 597–604.

[25] P. Poonpakdee, G. Di Fatta, Connectivity recovery in epidemic membership protocols, in: Proceedings of the 8th International Conference on Internet and Distributed Computing Systems (IDCS), Vol. 9258 of LNCS, Springer, 2015, pp. 177–189.

[26] A. Montresor, M. Jelasity, PeerSim: A scalable P2P simulator, in: Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09), 2009, pp. 99–100.