X-BOT: A Protocol for Resilient Optimization of Unstructured Overlay Networks

João Leitão, *Member*, *IEEE*, João Pedro Marques,
José Pereira, *Member*, *IEEE*, and Luís Rodrigues, *Senior Member*, *IEEE*

Abstract—Gossip, or epidemic, protocols have emerged as a highly scalable and resilient approach to implement several application level services such as reliable multicast, data aggregation, publish-subscribe, among others. All these protocols organize nodes in an unstructured random overlay network. In many cases, it is interesting to bias the random overlay in order to optimize some efficiency criteria, for instance, to reduce the stretch of the overlay routing. In this paper, we propose *X*-BOT, a new protocol that allows to bias the topology of an unstructured gossip overlay network. *X*-BOT is completely decentralized and, unlike previous approaches, preserves several key properties of the original (nonbiased) overlay (most notably, the node degree and consequently, the overlay connectivity). Experimental results show that *X*-BOT can generate more efficient overlays than previous approaches independently of the underlying physical network topology.

Index Terms—Peer-to-peer systems, unstructured overlay networks, topology adaptation, network protocols

1 Introduction

Gossip, or epidemic, protocols have emerged as a highly scalable and resilient approach to implement several application level services such as reliable multicast [1], [2], [3], [4], data aggregation [5], publish-subscribe [6], among others. Typically, a gossip-based broadcast protocol operates as follows: in order to broadcast a message, a node selects t nodes at random (t is a configuration parameter called *fanout*) and sends the message to them. Upon the reception of a message for the first time, each node simply repeats this procedure.

This approach has several advantages: 1) it is simple to implement, 2) it shares the load evenly across all nodes in the system, making gossip protocols highly scalable; in fact the load imposed by this process in each node of the system only grows logarithmically with the number of participants in order to ensure reliable broadcast with a high probability [1], [7], and finally, 3) its inherent redundancy enables gossip protocols to mask node and link failures, resulting in a highly resilient approach. Unfortunately, the inherent redundancy of gossip protocols also introduces unnecessary communication overheads in steady state, which not only consume bandwidth resources, but also can induce link stress in some links of the physical network.

Some gossip protocols have been designed to operate with full membership information [8], [1], by maintaining locally at each node a list with every other node identifier. Typically, in these protocols an identifier is a tuple $(ip, port)$ that allows a node to be reached. Such an approach, however, is not scalable, not only due to the potential large size of the membership list, but mainly due to the cost of maintaining such large amount of information up-to-date in large-scale dynamic environments where participants can join and leave the system at a high rate (a process known as churn [9]). For scalability, nodes may rely on a *peer sampling service* [10], which is a membership protocol that operates with the goal of maintaining locally, at each node, a small random subset (called a *partial view*) of the full membership list; in this case, nodes use their local partial views to select peers with whom they exchange messages.

Partial views establish *neighboring* associations among nodes that implicitly define an overlay network which can be used for gossip. Typically, a peer sampling service aims at maintaining a random partial view of the system [11], [12], [3] which should ensure that a selection of peers from local partial views is equivalent to a random selection of peers across the full membership. Therefore, the resulting overlay has a random unstructured topology. Although this randomness is desirable, it prevents the peer sampling service to take into consideration any efficiency criteria, which usually leads to scenarios where many of the overlay links are suboptimal, for instance, with regard to a given efficiency criteria, such as network bandwidth or latency. Unfortunately, the inefficiency of the overlay has a direct negative impact in the performance of applications that operate on top of the overlay (such as application-level reliable broadcast services). Moreover, applications may benefit from having the ability to bias the underlying overlay network topology to match some application specific criteria. For instance, in file sharing applications, one may organize the overlay such that neighbors share

- *J. Leitão and J.P. Marques are with Distributed Systems Group (GSD), INESC-ID Laboratory and the Instituto Superior Técnico (IST), Universidade Técnica de Lisboa, Sala 601, Rua Alves Redol 9, Lisboa 1000-029, Portugal. E-mail: {jleitao, jmarques}@gsd.inesc-id.pt.*
- *J. Pereira is with the Departamento de Informática, Universidade do Minho, Campus de Gualtar, Braga 4710-057, Portugal. E-mail: jop@di.uminho.pt.*
- *L. Rodrigues is with the INESC-ID/IST/Technical University of Lisbon, Rua Alves Redol 9, Office 605, Lisboa 1000-029, Portugal. E-mail: ler@ist.utl.pt.*

Manuscript received 9 May 2011; revised 23 Dec. 2011; accepted 5 Jan. 2012; published online 16 Jan. 2012.
Recommended for acceptance by L. Xiao.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2011-05-0294.
Digital Object Identifier no. 10.1109/TPDS.2012.29.

similar content. Overlay efficiency has been recognized as a key research topic for gossip-based protocols [13].

In this paper, we present *X*-BOT, a new protocol to **B**ias the **O**verlay **T**opology according to some target efficiency criteria **X**, for instance, to better match the topology of the underlying network. *X*-BOT is completely decentralized and, unlike prior works, it owns the following set of characteristics:

1. it operates only with local information and it does not require nodes to have a priori knowledge about their target location on the final topology;
2. it employs a new coordinated 4-node optimization technique that allows to achieve better overlay configurations;
3. the protocol strives to preserve the degree of the nodes that participate in an optimization step, which is fundamental to preserve the connectivity of the overlay;
4. every modification performed to the overlay increases its efficiency; this is feasible due to the dynamic nature of our model, which avoids the overlay from stabilizing in local minima;
5. the optimization performed by *X*-BOT preserves several key properties of the overlay such as a low clustering coefficient and low overlay diameter;
6. our scheme is highly flexible, as we rely on a companion *oracle* to estimate the link cost and, therefore, our algorithm can bias the network according to different efficiency criteria that are encoded in specific cost metrics just by using the appropriate oracle.

The rest of this paper is organized as follows: in Section 2, we introduce the most relevant aspects of unstructured overlay networks which are at the core of our proposal, and provide a brief overview of some building blocks that we use as the basis for our work. Section 3 addresses the related work. Section 4 describes the *X*-BOT protocol, explaining the rationale for our architecture and the proposed algorithm. An experimental evaluation of *X*-BOT is provided in Section 5. Finally, Section 6 concludes the paper.

2 Unstructured Overlay Networks

In this section, we enumerate several aspects of unstructured overlay networks and discuss their relevance for reliable broadcast. To make the paper self-contained, we also present an overview of the peer sampling service that we use as a building block in the design of *X*-BOT.

2.1 Structural Requirements

To support fast message dissemination and high level of resilience to node failures, overlay networks are defined by partial views that must exhibit several properties, such as connectivity, balanced (i.e., similar) in-degree across all nodes in the system, and low clustering coefficient. Notice that an overlay network can be seen as a graph, where nodes are represented by vertices, and links, or neighboring relations, are represented by edges. Depending on the nature of these relations, graphs can be directed or undirected.

In the following, we list some of the graph properties that are most relevant for the operation of the overlay. The interested reader can find a more detailed discussion of these and other properties of random overlays in previous work by Leitão et al. [3].

Connectivity. The overlay is connected if there is at least one path that allows every node to reach every other node in the overlay (most applications require the overlay to remain connected to ensure correctness). For instance, gossip-based broadcast protocols will not be able to disseminate messages through all participants if they operate on top of a disconnected overlay.

Degree distribution. The degree of a node is the number of edges of a node, or in other words, the number of neighbors that a given node has. When partial views are asymmetric, the degree has two distinct components: in-degree and out-degree. The in-degree of a node is the number of nodes in the system that contains node identifier in their partial views. The in-degree is a measure of the reachability of the node. The out-degree of a node is the number of distinct node identifiers present in that node's partial view and can be seen as a measure of its contribution to maintain the overlay connected. If the probability of failure is uniformly distributed in the node space, for improved fault-tolerance all nodes should have the same degree value. Nodes that have a small in-degree will become more easily disconnected from the overlay as the number of faults increases, and the failure of nodes with high out-degree may have an undesired impact in the overall connectivity of the overlay network. From the point of view of security, high degree nodes are also good targets if a malicious adversary aims at rendering the overlay disconnected. Furthermore, high degree nodes may lead to an increase in the stress over some underlay links, which might disrupt communications performed over the overlay network (for a discussion of the relevance of in-degree distribution in DHTs we point the reader to the work of Girdzijauskas et al. [14]).

Clustering coefficient. The clustering coefficient of a node is the number of edges between that node's neighbors divided by the maximum possible number of edges across those neighbors. This metric indicates a density of neighboring relations across the neighbors of a given node, having it's value between 0 and 1. The clustering coefficient of a graph (or overlay network) is the average of clustering coefficients across all nodes. The clustering coefficient of an overlay network should be as small as possible, and failure to meet this requirements also has several negative implications: 1) the number of redundant messages received by nodes when disseminating data increases, especially in the first steps of the dissemination process; 2) the diameter of the overlay increases, which in turn increases the overall latency of broadcast processes, and finally 3) it decreases the fault resilience of the overlay, as areas of the overlay which exhibit a high values of clustering can more easily became disconnected.

2.2 Performance Metrics

Several metrics can be used to measure the performance of a gossip-based broadcast protocol operating on top of a random overlay network. In this paper, we are mainly concerned with the efficiency and dependability of gossip-based broadcast protocols and associated applications. Therefore, we focus on biasing the overlay topology to minimize the message dissemination cost while preserving reliability.

Overlay cost. Similarly to what Rostami and Habibi propose [15], we assume that a *cost* may be associated with each link of the overlay. The overlay cost is the sum of cost for all links that form the overlay. *X*-BOT is independent of the cost function and only requires costs to be comparable and totally ordered (in fact, the algorithm does not use the absolute values of link cost, only their relative values). Costs may be associated to a concrete (underlay) network metric such as link latency. However, the link cost may also capture higher level utility metrics; for instance, in a file sharing peer-to-peer system, it could measure the semantic similarity among data stored at the link edges. More generally, a given overlay link cost is inversely proportional to the "utility" of that same overlay link. The cost value is provided by an *oracle* which is locally available at each edge node.

Reliability. In this paper we define gossip reliability as the percentage of correct nodes that deliver a given broadcast message. A reliability of 100 percent means that the protocol was able to deliver a given message to all active nodes. Reliability of 100 percent is impossible to achieve if the overlay network becomes disconnected.

The goal of our protocol is to reduce, as much as possible, the overlay cost without hampering relevant properties of that same overlay. For that purpose, when biasing the overlay topology, each node tries to select better neighbors. This notion of "better" is obtained by comparing the cost value of each possible neighbor, or in other words, the cost of maintaining, or using, an overlay link for each peer.

2.3 Building Blocks

As noted before, most unstructured overlays use some form of peer sampling service. As a building block to construct *X*-BOT, we rely on the *Hy*brid *Par*tial *View* peer sampling service [3] (or simply, HyParView). Unlike previous membership protocols, HyParView relies on two distinct partial views which are maintained using different strategies and for distinct purposes (in fact, the protocol is said to be Hybrid because it combines these two different strategies).

A small symmetric *active view* with $(\mathrm{fanout}+1)$ size is used to support communications among participants and disseminate broadcast messages. The active view is maintained using a reactive strategy, which means it is only updated in response to some external event that affects the overlay (e.g., a node joining or leaving). Each node maintains a TCP connection to each neighbor in its active view. The use of TCP allows the selection of small fanout for disseminating messages, as it masks network omissions. Moreover, TCP is used as an unreliable failure detector, which facilitates the implementation of the reactive maintenance strategy. The symmetry of these views helps to preserve the overlay connectivity, by providing each node with some measure of direct control over its own in-degree.

Each node also maintains a larger *passive view* usually k times larger than the active view, whereas k is a constant that is related with the desired fault tolerance level of the protocol. The passive view is maintained by a cyclic strategy, meaning that periodically each node performs a shuffle operation with one random node in the overlay that results in an update of both nodes passive views. This partial view is used for fault tolerance, as it works like a backup list of nodes that are used to ensure a constant out-degree for nodes.

3 Related Work

The adaptation of unstructured overlay topologies to better match the underlying physical topology (to avoid what is known as topology mismatch) or to better match applications requirements have been studied before. In this section, we survey, and compare with *X*-BOT, some of the most relevant works in the field. We mention additional research that is not so closely related with *X*-BOT in the supplementary document [16].

Narada [17] includes self-organizing protocols to construct and maintain overlay networks. Their approach is based in a utility function that is applied periodically to (some) peers; the output of the utility function is used to take local decisions concerning the addition and removal of links to the overlay. Because Narada is targeted at small and medium scale systems, it operates using full membership information and, therefore, scales poorly.

A work by Gupta et al. [18] also aims at increasing gossip efficiency by eliminating overlay links that transverse a given physical link multiple times. However, it relies on the fact that peers can be organized in a hierarchical manner, for instance, by grouping peers by network domains (e.g., Local Area Networks, Subnets, or even Autonomous Systems). Our approach does not require knowledge concerning the physical location of nodes nor the maintenance of any hierarchy among peers, being therefore more generic.

The Localizer algorithm [19] is an algorithm that aims both at optimizing unstructured overlays according to a proximity criterion and to promote the balancing of node degrees. Localizer is based on a Metropolis scheme where nodes, iteratively, strive to minimize an energy function, by swapping connections. Although this algorithm strives to balance the degree of nodes in the system, unlike our approach, it does not ensure a constant degree in nodes that participate in the optimization. Furthermore, the Localizer algorithm is more likely to fall in local minima of the energy function, due to the fact that the pool of peers available to generate overlay optimization is limited. Therefore, it is required to sometimes increase the energy function, which compromise the stability of the overlay. Moreover, Localizer does not attempt to preserve low clustering and small diameter.

GoCast [20] and Araneola [21] also include mechanisms to bias the topology of overlay networks maintaining symmetric partial views supported by TCP connections. GoCast builds an overlay which is optimized to maintain both random (distant) neighbors and close neighbors while balancing node degree in such a way that degree of nodes converge to a given preestablished value D and varies only between $D-2$ and $D+2$. Araneola is similar to GoCast in the sense that it also builds an unstructured overlay network that presents several properties of k-regular graphs. Similar to GoCast, Araneola controls its topology to take into consideration some network metric, ensuring that better links are kept with a larger probability. However, these protocols rely on mechanisms to bias the overlay that are more complex and where each node makes independent decisions. In sharp contrast *X*-BOT uses a coordinated 4-node optimization technique that is simpler and allows to improve the overlay topology while maintaining a better node degree distribution. In Section 5, we experimentally compare these protocols with our own.

T-Man [22] is a generic topology management scheme for overlay networks that is able to evolve a given overlay topology to a desired target topology (such as a torus or a ring). This is achieved by having neighbors periodically exchanging their partial views. Both nodes update their partial views by merging these views and selecting the best c nodes, where c is the size of a partial view. The selection is based on a single ranking function which captures the desired topology, in the sense that it enables each node in the system to extract clues on its optimal position and the correct neighbors in the overlay network. Unlike *X*-BOT, T-Man does not aim at protecting relevant properties of the original overlay, nor it ensures the stability of in-degree of nodes during the optimization of the overlay (see Section 5).

A more detailed description of existing overlay optimization algorithms can be found in Leitão et al. [23], which includes a survey of several optimization approaches.

4 *X*-BOT

4.1 Rationale

From the analysis of relevant unstructured overlay network properties and observation of previous approaches, we define a set of guidelines that should be respected when trying to bias the topology of these overlays. We followed these guidelines when designing *X*-BOT and believe that the success of our approach stems from these principles.

Limited number of nodes. Each adaptation to the overlay topology should involve only a small limited number of participants. This is a requisite to ensure the scalability of the biasing mechanism.

Limited information. The adaptation of the overlay topology should only rely in limited information. This is required to lower the overhead of the adaptation process and reduce the communication among participants. Additionally, partially incorrect information should not completely disrupt the operation of the biasing mechanism.

Maintain node degree. To avoid an increasing risk of overlay disconnection the degree of nodes that participate in an adaptation of the overlay should remain constant. If this requisite is not met, several adaptation steps in sequence might compromise the overlay connectivity.

Promote stability. Adaptations to the overlay should promote the stability of the overlay topology to avoid the disruption of peer-to-peer services being executed on top of it. This can be achieved by only adapting the overlay topology when a direct benefit stems from that operation. Moreover, nodes whose connectivity is not at its maximum (i.e., that do not have the maximum amount of neighbors possible) should not participate in adaptation operations.

Maintain unbiased links. To protect the diameter and clustering in the overlay a portion of the links of the overlay should remain without being biased. This will ensure that some level of randomness will remain in the overlay topology ensuring the adequate operation of peer-to-peer services and applications being executed on top of it.

4.2 Oracles

We assume that all nodes have access to a local *oracle*. Oracles are components that export a `getLinkCost(Peer` p`)` method, which returns the link cost between the invoking node and the given target node p in the system (since there is a single link to each neighbor, in the paper we use interchangeably link cost or node cost when referring to the output of the oracle). The implementation of the oracle is outside the main scope of this paper. However, for completeness, we provide a brief description of two simple oracles: one based on a network metric and the second based on a more high level metric.

Latency oracle. One of the most intuitive oracles that can be devised is a latency oracle. This oracle operates by collecting and storing measures of round trip times (RTT) between peers, using specific probe messages exchanged by the oracles. The oracle must be aware of the peers which are known at the local host, and it measures the RTT for each known node storing the last reading (or some weighted average), which can be directly used as the link cost value. Moreover, if TCP connections are maintained among peers, the oracle can use the estimated RTT calculated by TCP as the link cost avoiding the explicit exchange of messages to perform these measurements.[1]

Internet service provider oracle. In a setting where sending messages to a node in a different ISP has an increased financial cost compared to sending a message to a node in the same ISP, it might be useful to keep as many neighbors as possible from the local ISP. An oracle to this end could be built by maintaining information concerning the local ISP and a table of costs for each known ISP (this table could simply store a low value for the local ISP and a high value for all others). When the oracle becomes aware of a new peer, it simply exchanges a message with the remote oracle to identify local ISP's, and assert the cost for the link using its local cost table.

We further discuss the type of oracles supported by *X*-BOT as well as some alternative oracles design and limitations in the supplementary document [16].

4.3 Architecture

The rationale of our approach is as follows: in a similar fashion to HyParView, we maintain a small active view and a larger passive view. However, unlike HyParView, which strives to ensure the stability of the overlay denoted by the active views, *X*-BOT relaxes stability in order to continuously attempt to improve the overlay according to some efficiency metric embedded in the companion oracle. This allows the topology of the unstructured overlay to self-adapt in order to better match the requirements of the application, or services, executed on top of it. Periodically, each node starts an *optimization round* in which it attempts to swap one member of its active view with one (better) neighbor of its passive view. While executing the optimization algorithm, a node uses its local oracle to obtain an estimate of the link cost to some randomly selected peers of its passive view. The number of nodes scanned in each set of optimization rounds is a protocol parameter called

1. We discuss how measurements can be performed by taking into consideration the variation of Latency in the supplementary document [16].

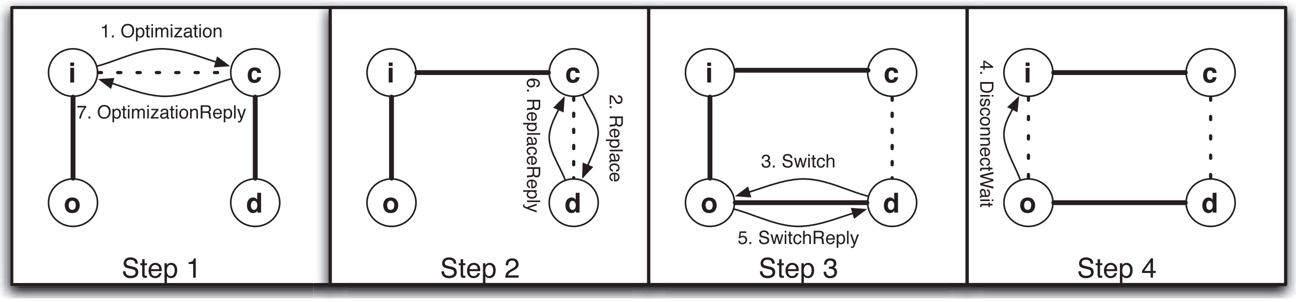


Fig. 1. Typical *X*-BOT optimization round.

Passive Scan Length and simply denoted π. This parameter limits the maximum number of optimization rounds started by each node each time it runs the optimization procedure.

The passive view is not biased. However, the passive view should be continuously updated during the system operation, so that it reflects the changes in the global membership (e.g., nodes that leave the system, are eventually purged from all passive views, and nodes that join the system eventually appear in some of the passive views). Therefore, passive views can be used as a continuous source of potential nodes that can be upgraded to the active view, in order to bias the overlay to a better topology. This also helps in avoiding our algorithm from falling into local minima configuration.

X-BOT strives to preserve the connectivity of the overlay. This has two implications in our scheme: 1) nodes only make an effort to optimize their active views when they have a full active view (i.e., no bias is applied to active views until connectivity of the nodes is ensured). Furthermore, each node attempts to maintain some unbiased neighbors, as we explain in the next section; 2) we try to preserve the degree of nodes that participate in an optimization procedure. Moreover, we ensure that optimizations preserve the symmetry of the active view, which is essential to ensure a good distribution of in-degree, which in turn has a significant impact on the connectivity of the overlay. Typically, each optimization round involves four nodes in the system as we will detail later in the text.

Unbiased neighbors. By eagerly imposing a bias on the topology of the overlay, one may easily break some of the key desirable properties of a random overlay, such as the low clustering coefficient, low average path length, or connectivity [20].[2] The negative effect of such bias can be even more notorious in an architecture such as ours, that relies on small active views. To avoid this flaw, we do not apply the bias to all members of the active view. Instead, each node should maintain both "high-cost" (unbiased) and "low-cost" (biased) neighbors. The number of "high-cost" neighbors each node keeps is a protocol parameter called *Unbiased Neighbors* and simply denoted μ.

Unfortunately, it is not trivial to decide which peers have a "high-cost," given that nodes are not required to have global knowledge of the system, not only regarding membership information but also regarding global metrics, such as the overlay cost. To circumvent this problem, we maintain the active views sorted by link cost (the first element of each active view is the neighbor with the largest link cost). A node never attempts to apply the bias to the first μ members of its active view.

Notice that this scheme implicitly imposes a passive biasing to unbiased neighbors, as the unbiased neighbors kept by each node in *X*-BOT are those neighbors that were in that node's active view that present the higher cost accordingly to the local oracle. We say that this is a passive biasing process as no active effort is made by any node to produce this effect (i.e., nodes do not actively exchange their unbiased neighbor to promote more distant neighbors). In the supplementary document [16] we provide experimental results that compare the average link cost of unbiased and biased neighbors of nodes.

4.4 Algorithm

The *X*-BOT algorithm is depicted in Algorithm 1 and illustrated in Fig. 1. The reader should notice that the algorithm presented has been simplified for clarity. For instance, we omitted some insertions of nodes into passive views and the mechanisms required to ensure the symmetry of active views.[3] A typical optimization round involves four nodes of the system, and each round is composed of four steps, one for each node that participates in the optimization. We use the following definitions to identify each of the participating nodes. Node i (***initiator***)**:** is the node that starts the optimization round. Node o (***old***)**:** is a node from i's active view which is replaced during the optimization process. Node c (***candidate***)**:** is a node from i's passive view which is upgraded to the active view. Node d (***disconnected***)**:** is the node to be removed from the candidate's active view in order to accept i.

Algorithm 1. *X*-BOT: Improving Procedure

```
1:  every $\Delta T$ do
2:      if isFull(activeView) then
3:          candidates ⟵ randomSample(passiveView, $PSL$)
4:          for $i$ := $UN$ ; $i <$ sizeof(activeView); $i$ :=$i + 1$
5:              $o$ ⟵ activeView[$i$]
6:              while candidates $\neq$ {} do
7:                  $c$ ⟵ removeFirst(candidates)
8:                  if isBetter($o$,$c$) then
9:                      Send(OPTIMIZATION, $c$, $o$, myself)
10:                     break
```

2. We provide experimental results further ahead in Section 5 that illustrate the effect of *X*-BOT's operation over these overlay properties.

3. The complete algorithm is available online at http://gsd.inesc-id.pt/~jleitao/.

```
11: upon Receive (OPTIMIZATIONREPLY,answer,$o$,$d$,$c$) do
12:     if answer then
13:         if $o \in$ activeView do
14:             if $d \neq$ null then
15:                 Send(DISCONNECTWAIT, $o$, myself)
16:             else
17:                 Send(DISCONNECT, $o$, myself)
18:             activeView $\longleftarrow$ activeView $\setminus\{o\}$
19:         passiveView $\longleftarrow$ passiveView $\setminus\{c\}$
20:         activeView $\longleftarrow$ activeView $\cup\{c\}$

21: upon Receive(OPTIMIZATION, $o$, peer) do
22:     if $\neg$ isFull(activeView) then
23:         activeView $\longleftarrow$ activeView $\cup$ {peer}
24:         Send(OPTIMIZATIONREPLY, true, $o$, null,
            myself)
25:     else
26:         $d \longleftarrow$ activeView[UNOPT]
27:         Send(REPLACE, $d$, $o$, peer, myself)

28: upon Receive(REPLACEREPLY,answer,$i$,$o$,$d$) do
29:     if answer then
30:         activeView $\longleftarrow$ activeView $\setminus\{d\}$
31:         activeView $\longleftarrow$ activeView $\cup\{i\}$
32:     Send(OPTIMIZATIONREPLY,answer,$o$,$d$,myself)

33: upon Receive(REPLACE, $o$, $i$, $c$) do
34:     if $\neg$ isBetter($peer$,$o$) then
35:         Send(REPLACEREPLY, $c$, false, $i$, $o$, myself)
36:     else
37:         Send(SWITCH, $o$, $i$, $c$, myself)

38: upon Receive(SWITCHREPLY,answer,$i$,$c$,$o$) do
39:     if answer then
40:         activeView $\longleftarrow$ activeView $\setminus\{c\}$
41:         activeView $\longleftarrow$ activeView $\cup\{o\}$
42:     Send(REPLACEREPLY,answer,$i$,myself)

43: upon Receive(SWITCH,$i$,$c$,$d$) do
44:     if $i \in$ activeView or received(DISCONNECTWAIT
        from $i$) then
45:         Send(DISCONNECTWAIT,$i$,myself)
46:         activeView $\longleftarrow$ $\tilde{a}$ctiveView $\setminus\{i\}$
47:         activeView $\longleftarrow$ $\tilde{a}$ctiveView $\cup\{d\}$
48:     Send(SWITCHREPLY,answer,$di$,$c$,myself)
49: isBetter($old$,$new$)
50:     return Oracle.getCost($old$) $>$ Oracle.getCost($new$)
```

4.4.1 Step 1

Step 1 is executed at node i (Algorithm 1, lines 1-10) and its purpose is to contact one, or more, potential candidates to participate in a set of optimization rounds.[4]

This step starts with the random selection of, at most, π nodes from the i's passive view. This random sample is a set of candidates for executing the optimization round. To check if a target node is a suitable candidate, i iterates over its active view, consulting the oracle to compare the cost of its neighbors with the cost of the target (Algorithm 1, lines 49-50). When a suitable candidate c is found, which presents a possibility for improving a given neighbor o, i sends to c an OPTIMIZATION message, stating its interest in exchanging o for c in its active view. The reception of that message will trigger the execution of Step 2 in node c.

This step ends with the reception of an OPTIMIZATIONREPLY message from node c (Algorithm 1, lines 11-20), or with the suspicion of failure of node c. If node c accepts the exchange, then i will add c to the active view. If o is still in its active view,[5] i will send a DISCONNECTWAIT or DISCONNECT message to o. The difference between these messages is simple: DISCONNECT, only removes the sender from the active view (as described in our previous work [3]) while DISCONNECTWAIT also notifies the node that it should maintain (until an internal timeout expires) that free slot in the active view, which will be used in step 4. Node i chooses which message to send, based on information received from c, specifically, if c had to remove some node from its active view in order to insert i in its active view.

4.4.2 Step 2

Step 2 is initiated at node c with the reception of an OPTIMIZATION message from node i (Algorithm 1, lines 21-27) and ends when c replies to i with a OPTIMIZATIONREPLY message.

If c does not have a full active view it immediately replies to i by sending an OPTIMIZATIONREPLY message accepting the exchange, and notifying i that no other node was involved in the optimization. In this case i will disconnect itself from o and insert c in its active view. Note that in this particular scenario, our algorithm does not preserve the degree of node o, although it preserves the number of links in the overlay. However, this is a uncommon scenario given that, according to our experiments, in steady state usually more than 97 percent of nodes in the system have full active views. On the other hand, if c has a full active view, c has to select some neighbor d from its active view to exchange for i. Therefore, c sends to d a REPLACE message, stating its desire to remove d from its active view; this message also indicates to d that it can connect to o in exchange. The REPLACE message also carries information concerning the identification of the initiator of the optimization procedure. In order to decrease the average link cost, the selection of d is deterministic, in such a way that d is the neighbor of c with the higher cost (excluding, naturally, the first μ protected members).

In the latter case, to conclude this step, c has to receive a REPLACEREPLY message from node d (Algorithm 1, lines 28-32) or suspect that d has failed (in which case, node c acts as if it had a free slot in the active view from the beginning of this step). If d accepts the exchange, c will remove d from its active view and replace it with i. If d declines the exchange, c does not change its own views. In either case, c will notify i of d's answer using the OPTIMIZATIONREPLY message.

4. More than an optimization round might be triggered in the context of this step as discussed previously.

5. Note that o might have already disconnected from i as a result of the execution of step 4.

4.4.3 Step 3

This step begins with the reception at node d of a REPLACE message (Algorithm 1, lines 33-37) and ends when a REPLACEREPLY message is sent back to node c.

A REPLACE message explicitly requests node d to exchange node c with node o in its active view. Node d consults the oracle to assess if o has a lower link cost than c. Note that our algorithm only requires 2 of the 4 nodes involved in an optimization round to consult the oracle in order to assess the merit of the proposed peer exchange. This is enough as we assume that link costs are approximately symmetric (in Section 5 we evaluate the performance of the protocol in a realistic scenario), and effectively, we are exchanging two existing links in the overlay, for other two. We exchange the link between i and o for a link between i and c and the link between d and c with the link between d and o. Therefore, only nodes i and d need to assess the gain resulting from these exchanges. The conservative use of the oracle provides a benefit if the oracle implementation adds some overhead (e.g., messages sent) whenever it is consulted by a node.

Naturally, if node d verifies that there is no gain in the exchange of c for o, d aborts the exchange by notifying c. Otherwise, it will send a SWITCH message to node o notifying him to switch node i in its active view for himself. Moreover, it notifies node o. Finally, the answer received from o in a SWITCHREPLY (Algorithm 1, lines 38-42) is forwarded to c.

4.4.4 Step 4

This step is executed by node o upon the reception of a SWITCH message (Algorithm 1, lines 43-48) and ends when o sends a SWITCHREPLY to d. This step is required only to ensure the symmetry of active views. The behavior of node o in this step is deterministic. For clarity, in Algorithm 1 we only depict 2 of the constraints that are checked before accepting the exchange. The complete list of constraints that have to be checked can be found in Leitão et al. [3]. After checking all constraints (related for instance, with active view symmetry and nonrepetition of node identifiers between the active and passive views), node o sends a DISCONNECTWAIT message to i and adds d to its active view. This concludes an optimization round.

4.4.5 Accuracy of Oracles

Depending on the implementation and efficiency metrics computed by the oracle, it can provide values that are not completely accurate. However, this will not have a significant impact in the overlay correctness. In fact in scenarios where the oracle provides random cost values, the resulting topology will have the same characteristics of a random unbiased overlay while remaining connected and with an adequate in-degree distribution.

In scenarios where the (average) accuracy of the oracle is known (for instance a work by Karwaczyński et al. [24] states that longest IP prefix and latency has an approximate correlation of -0.85), the X-BOT protocol can be easily adapted to this, by changing the **isBetter** evaluation function to include some hysteresis, namely, a link new should only be considered as having a lower cost than other link *old*, if the difference between the cost (obtained through the oracle) offers a gain above a given *Threshold*, which should be a protocol parameter that takes into consideration the accuracy of the deployed oracle. Algorithm 2 depicts the required changes to the **isBetter** function.

Algorithm 2. Alternative for the isBetter procedure

```
isBetter($old$,$new$)
    cOld := Oracle.getLinkCost($old$)
    cNew := Oracle.getLinkCost($new$)
    return cOld > cNew $\wedge \frac{(c\,Old - c\,New)}{c\,Old} \geq THRESHOLD$
```

4.4.6 Oracle Cost

Some oracle implementations may require the explicit exchange of messages among distinct nodes, in order to provide a cost for the execution of the algorithm. For instance, consider the simple oracle that we described earlier in the paper that exchanges ping messages to compute an approximation of the latency between nodes.

This could introduce some undesired overhead to our protocol. However, in our solution each optimization step only requires two nodes to consult their local oracles. Moreover, the number of times that a node consults its oracle in the first step of our protocol can be limited by the π parameter; our experimental results have shown that it can be set to a small value (such as 2).

Finally, the oracle can also perform their measurements in background, decoupling the traffic generated by the oracle from the traffic created by the X-BOT protocol.

4.4.7 Additional Aspects

The interested reader may consult the supplementary document [16] for further discussion on some aspects of X-BOT namely, in terms of complexity analysis, theoretical background, and discussion on the protocol properties, such as avoiding local minima, ensuring low clustering coefficient, and low average shortest path.

5 EVALUATION

5.1 Experimental Setting

We conducted an extensive experimental evaluation of X-BOT against GoCast, Araneola, and T-Man using the PeerSim simulator [25]. To that purpose, all protocols were implemented in the simulator using the cycle-based engine of PeerSim. Furthermore, all protocols use the same oracle to allow a fair comparison. Also, to assess that our implementations of GoCast, Araneola, and T-Man are accurate, we did compare their performance with the results that have been published in the corresponding papers. In the experiments reported here, we use the following scenarios, that allow us to assess the benefits of X-BOT in environments with different characteristics.

Cartesian scenario. This scenario uses a network of 10,000 nodes organized in a cartesian plan (a 100×100 square), where two direct neighbors are at a distance of 1. We model the cost of a link between two nodes, as being equal to the distance between those nodes in the cartesian space. This scenario is interesting as it offers a high potential to optimize a random overlay. Moreover the link costs in this scenario are symmetric and have a gaussian distribution.

PlanetLab scenario. This scenario is composed of 341 nodes in which the cost of a link is defined according to the *all pair pings* trace[6] that contains ping times measured among a set of PlanetLab[7] nodes. Each simulated node represents a PlanetLab node and the cost between any two nodes, n_i and n_j, is set as half of the round trip time between these two nodes as measured from n_i according to the PlanetLab traces. Notice that in this scenario, link costs are not necessarily symmetric. This scenario allows us to observe the performance of *X*-BOT in a realistic setting.

Inet-3.0 scenario. This scenario is composed of 10,000 nodes. In this scenario we used the network generator Inet-3.0 [26] to distribute 10,000 nodes over a network of autonomous systems using the default parameters of Inet-3.0.[8] We then computed the shortest path for every pair of nodes, and defined the latency of link to be equal to the distance between routers that are transversed by shortest path connecting the end-points of the link. The link cost was set to be the latency between the end-points of a link. This scenario allows to evaluate the performance of protocols in a large-scale realistic scenario, where links between nodes present a wider range of possible values.

We start by providing an experimental study on the properties of the overlays that result from the operation of *X*-BOT while maintaining distinct numbers of unbiased members (e.g., using different values for the μ parameter). We provide these results as a proof of concept, allowing us to show the improvement capabilities and the flexibility of our approach. Moreover, these results provide some insight on the effect of our optimization procedure to the clustering of the overlay.

We follow by providing experimental results for configurations where all protocols attempt to maintain a single random/unbiased neighbor (both our experiments and previous work [20] suggest that this is the most useful configuration for most scenarios, namely when the overlay is used to support gossip-based broadcast protocols).

In all experiments we conducted, the size of the passive views maintained by *X*-BOT was set to 30. The remaining protocols benefit from a (similar) random partial view maintained by a companion membership protocol. To ensure a fair comparison, we also set the size of these views to 30. Moreover, we initialized these views with contents extracted from the passive views of HyParView after 250 simulation cycles. Simple maintenance routines for these views, similar to the ones employed by *X*-BOT, were also added to each protocol. Furthermore, because T-Man lacks a join procedure, we initialized its views with contents extracted from the active views of HyParView in similar conditions.

Furthermore, for all simulations presented in this paper, *X*-BOT was configured as follows: *Period between optimizations* was set to two simulation cycles. This increases the probability of having new peers in the passive view of a node in each optimization step. As described previously, passive views are updated each cycle; *Passive scan length* (π) was set to 2, so each time a node executes the step 1 of the optimization algorithm, it tests, at most, two nodes from its passive view. This also limits to two the number of nodes which are exchanged in a single round for a node.

Finally, the initial (external) partial views provided to Araneola and GoCast were sorted by link cost. This is the most favorable configuration for these protocols. However, this strategy requires additional invocations to the oracle, which might incur in additional overhead if the oracle implementation requires the exchange of messages. All results reported in the paper are an average of 5 independent runs of each simulation. Confidence intervals were verified and were small. We omit these from the figures for improved readability.

5.2 Overlay Properties

5.2.1 X-BOT Individual Evaluation

In this section, we evaluate a set of relevant metrics that focus on properties of the overlay topology that results from the operation of *X*-BOT. Experiments were conducted by executing *X*-BOT for 1,000 simulation cycles, and observing the evolution of the overlay and its final state. Values presented here are an average of five independent runs, and address all the simulation scenarios described earlier.

In this set of experiments, we are mainly concerned with evaluating the effect of keeping a different number of *Unbiased Neighbors* in the overlay. To this end, we tested the protocol using values for the μ parameter that range from 0 (none) to 5 (all). In Fig. 2 we plot the evolution of the overlay cost, clustering coefficient, and average shortest path of the *X*-BOT overlay, as the protocol operates in both the cartesian, PlanetLab, and inet-3.0 scenarios. We now discuss these results.

Overlay cost. Figs. 2a, 2b, and 2c show, respectively, the evolution of the overlay cost in the cartesian, PlanetLab, and inet-3.0 scenarios. In all scenarios the protocol shows a similar behavior. As the protocol keeps less unbiased neighbors (e.g., as the μ parameter lowers) the final cost of the overlay becomes lower. Notice that the overlay cost in all scenarios with a μ value of 5 is constant. In fact, in this case the overlay is not biased, and this case is used as a baseline to assert the benefits of *X*-BOT.

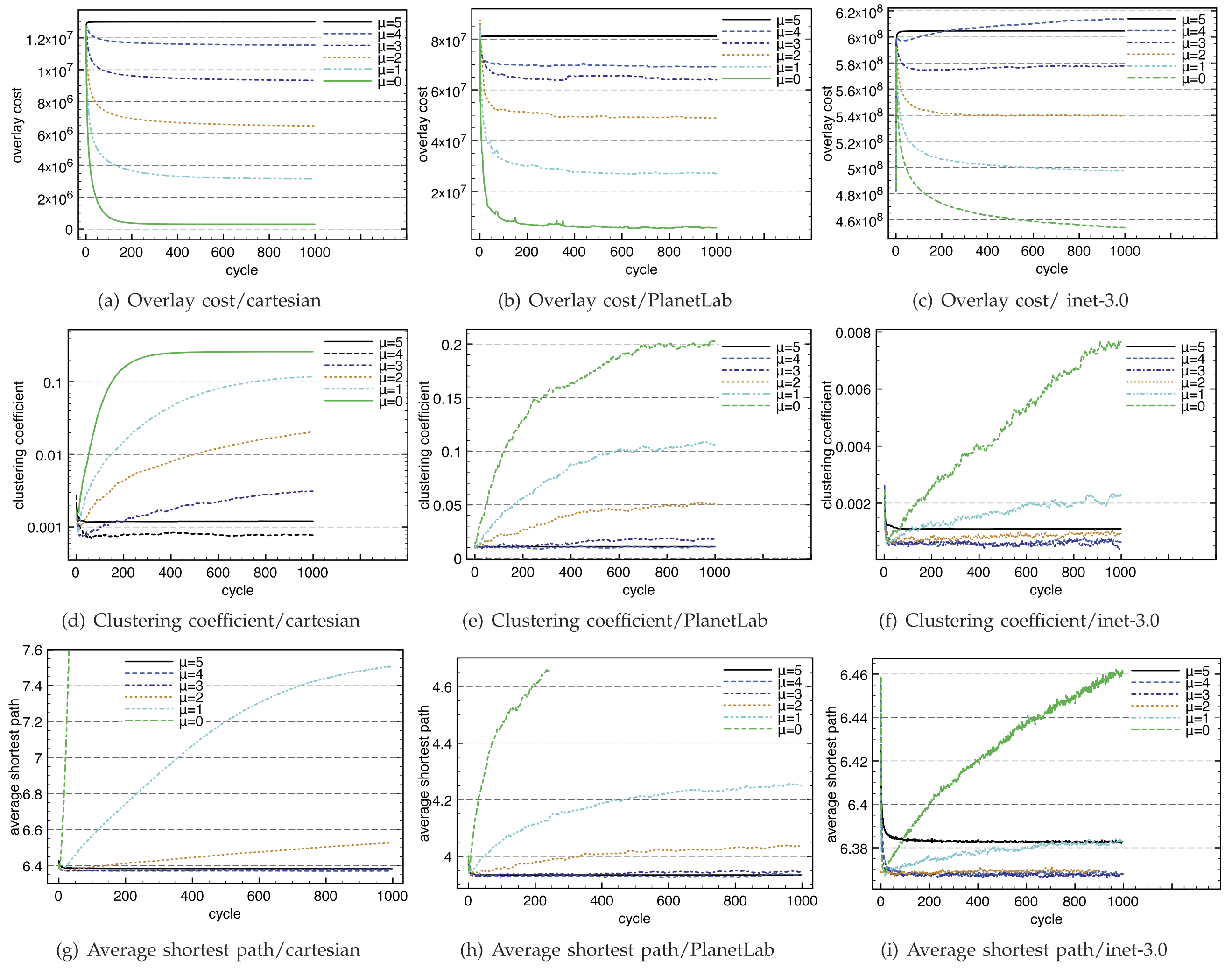
Although the protocol behavior is consistent in all scenarios, the reduction in the overlay cost is not. For instance, whereas in the cartesian scenario with μ value of 0, the overlay cost is reduced by a value on the order of 1×10^7 presenting a final overlay cost close to 3.5×10^5, the same configuration in the PlanetLab scenario allows the overlay to lower its cost by a value close to 1×10^8, still exhibiting a final cost in the order of 3×10^8. In the Inet-3.0 scenario the overlay cost drops by approximately 1.4×10^8. This is due to the nature of the underlying network and the metric being computed by the oracle. In fact, distinct physical topologies and different oracles offer distinct potential to improve the overlay network above it.

Interestingly enough, in the Inet-3.0 scenario with a μ value of 4 (only one neighbor is biased) the overlay cost slightly rises. This is due to the fact that the operation of *X*-BOT implicitly bias *unbiased* neighbors to promote distant

6. A repository with these traces can be found in: http://pdos.csail.mit.edu/~strib/pl_app/.

7. http://www.PlanetLab.org/.

8. In fact we generated five different network topologies, each generated topology was used in each individual simulation run.



(a) Overlay cost/cartesian (b) Overlay cost/PlanetLab (c) Overlay cost/ inet-3.0

(d) Clustering coefficient/cartesian (e) Clustering coefficient/PlanetLab (f) Clustering coefficient/inet-3.0

(g) Average shortest path/cartesian (h) Average shortest path/PlanetLab (i) Average shortest path/inet-3.0

Fig. 2. Evolution of *X*-BOT overlay properties (overlay cost, clustering coefficient, and average shortest path) when varying the number of unbiased neighbors.

connections in order to improve the overlay diameter. This scenario is the one where link costs present a wider range of values, which leads, with a single biased neighbor, the overlay to bias it topology to promote higher cost links as we further detail below.

Clustering coefficient and average shortest path. Figs. 2d, 2e, and 2f depict results for clustering coefficient in the cartesian, PlanetLab, and inet-3.0 scenarios, respectively. As expected, biasing all elements in the partial view of all nodes highly increases the clustering coefficient of the overlay. On the other hand, maintaining a single unbiased neighbor is enough to mitigate this effect.

Interestingly, in the cartesian scenario with four unbiased neighbors and in the inet-3.0 scenario with 2 to 4 unbiased neighbors the resulting clustering coefficient drops to values below that of the unbiased overlay. This phenomenon can be explained as follows: by maintaining active views sorted by cost, the selected unbiased neighbors are those with a larger cost. In other words, our algorithm, at no extra cost, promotes the maintenance of "long distance" nodes in each view.

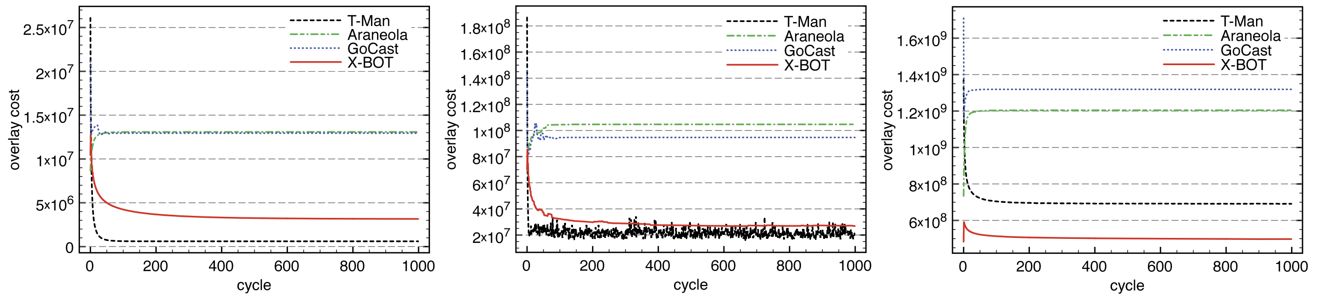
The reader should also notice that Fig. 2d has a logarithmic scale. In fact, the increase of clustering coefficient is much higher in this scenario. This is not a surprise, as in this scenario the cost function computed by the oracle is related with the distance between nodes, which is a transitive property. This leads us to the conclusion that when biasing an overlay by relaying a transitive performance metric one should relay in additional unbiased neighbors.

We present similar results concerning the average shortest path of the resulting overlays in Figs. 2g, 2h, and 2i. As we have noted earlier, there is a strong correlation between the clustering coefficient and the average shortest path of an overlay network. Therefore, these plots show results that are consistent with the previous ones.

When no unbiased neighbor is kept, the average shortest path of the overlay rapidly rises to values off the scale in the cartesian scenario (Fig. 2g). This happens due to the nature of the cartesian scenario, in a consistent manner with results presented before.

5.2.2 Comparative Evaluation

In this section, we extend the previous experiments to provide comparative figures of the *X*-BOT performance against other protocols. The experimental work discussed in this section was conducted as described in the previous section. As stated earlier, we focus our evaluation in



(a) Overlay cost/cartesian (b) Overlay cost/PlanetLab (c) Overlay cost/inet-3.0

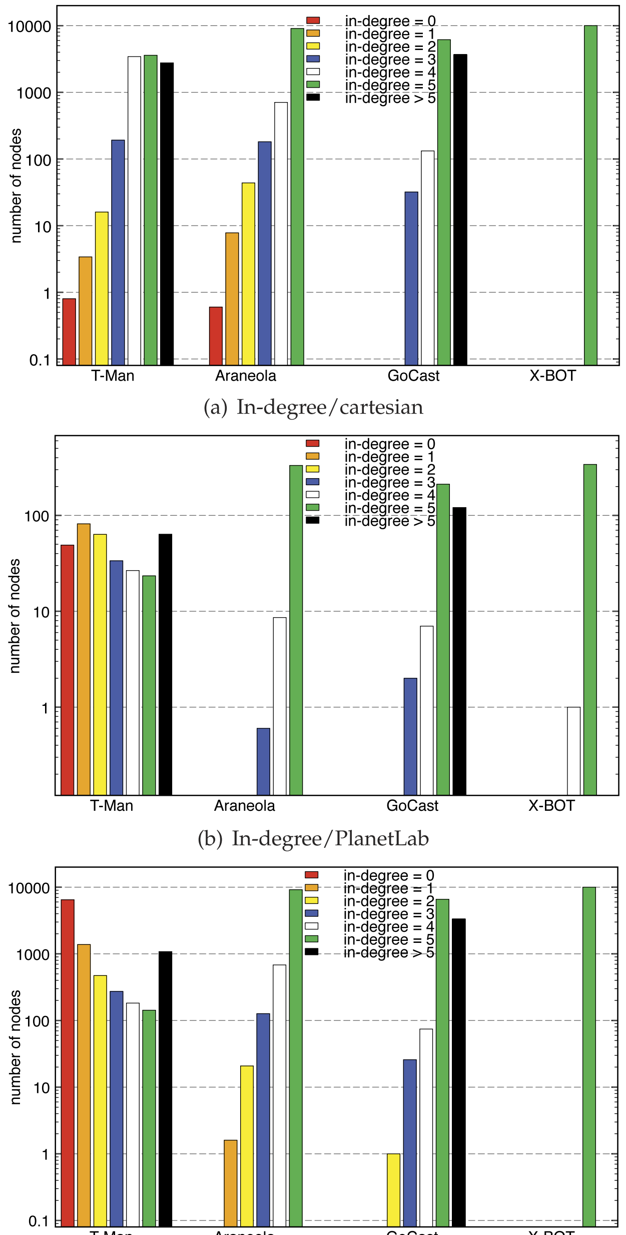
Fig. 3. Cost of the overlay created by each protocol for the three experimental settings.

scenarios where only one unbiased/random neighbor is maintained by each of the protocols.

We consider that the most relevant metric is the overlay cost, as it allows to assert the benefits of employing each of the protocols. Figs. 3a, 3b, and 3c depict, respectively, the overlay cost for the cartesian, PlanetLab, and inet-3.0 scenarios for all protocols. Compared with both Araneola and GoCast, *X*-BOT achieves a lower overlay cost. This can be explained as follows: Araneola is a reactive protocol, in the sense that once the partial view of a node stabilizes (i.e., by matching all protocol requirements) it will never be updated again until some external event happens (e.g., a neighbor fails). Therefore, Araneola does not explore the full optimization potential of the environment. GoCast, on the other hand, is able to iteratively improve the overlay topology, unfortunately the protocol does not ensure a constant degree of nodes (see Figs. 4a. 4b, and 4c), which results in the creation of additional links that increase the overlay cost. Moreover, considering that the average cost of each link maintained by GoCast is higher than the cost of the links maintained by *X*-BOT, it is possible to observe that our 4-node coordination optimization strategy offers better results than the GoCast strategy.

T-Man achieves a performance that is similar to *X*-BOT in the PlanetLab scenario and can even achieve a more efficient overlay than *X*-BOT in the cartesian and inet-3.0 scenarios. This is due to its aggressive optimization strategy. Unfortunately, this aggressive strategy has severe drawbacks, namely it has a negative impact in the overlay connectivity. Figs. 4a. 4b, and 4c depict the in-degree distribution obtained with each protocol. T-Man generates overlays where several nodes exhibit an in-degree of 0 (which is specially noticeable in the inet-3.0 scenario where many nodes are distant from the core of the network) while other nodes have a very a high in-degree (as high as 120 neighbors in the PlanetLab scenario, and 1,277 neighbors in the inet-3.0 setting). As a result, T-Man is unable to preserve the connectivity of the overlay, which has a negative impact on applications and services running on top of the overlay (such as gossip-based broadcast protocols). Notice that in this scenario, and for load balancing during broadcast message dissemination, ideally each peer should have the same in-degree value, as this ensures that each peer has to send and receive (on average) a similar number of messages.

Table 1 shows the resulting clustering coefficient (CC) and average shortest path (ASP) for all protocols in all



(a) In-degree/cartesian

(b) In-degree/PlanetLab

(c) In-degree/inet-3.0

Fig. 4. In-degree of the overlay created by each protocol for the three experimental settings.

TABLE 1
Comparison of Overlay Properties for All Tested Protocols

	Cartesian		PlanetLab		Inet-3.0	
	CC	ASP	CC	ASP	CC	ASP
T-Man	0.271	∞	0.350	∞	0.197	∞
Araneola	0.193	9.808	0.070	4.235	0.1048	9.66
GoCast	0.0009	6.021	0.024	3.793	0.0004	6.03
X-BOT	0.117	7.506	0.098	4.230	0.0023	6.38

scenarios. Notice that T-Man does not present a value for average shortest path, as none of the executions of the protocol was able to construct a connected overlay. *X*-BOT and GoCast offer the best results, although GoCast achieves these results at the expense of maintaining several nodes with a node degree higher than the target value.

Despite the fact that *X*-BOT presents a somewhat higher clustering coefficient when compared with GoCast for all scenarios, in terms of average shortest path *X*-BOT has values only slightly above those of GoCast, a fact which is specially noticeable in the inet-3.0 scenario (which is the scenario that betters model the operation of protocols over a large-scale network with properties similar to those of Internet). This happens because contrary to GoCast, *X*-BOT implicitly bias unbiased neighbors to promote distant links, resulting in an overlay with lower diameter. This allows *X*-BOT to better support several peer-to-peer services, such as gossip-based broadcast protocols as we will show further ahead in the text.

5.2.3 Scenario with Several Internet Service Providers

In order to illustrate the flexibility of *X*-BOT, we evaluated the performance of the protocol in an additional setting. This setting is similar to the cartesian scenario with the difference that we associate to each node in the system one internet service provider. The assignment has been performed such that the number of nodes belonging to each internet service provider is approximately the same.

We rely on an implementation of an Internet Service Provider Oracle that operates by providing a cost of 0 units to any node belonging to the same internet service provider, and a large constant value (in our case 1,000) to all remaining nodes. We performed experiments with *X*-BOT in scenarios where a different number (ranging from 2 to 10) of internet service providers coexist. In order to extract comparative measures, we performed similar experiments with T-Man, Araneola, and GoCast.

In such a specific scenario we rely on a metric that better captures the efficiency of the resulting overlay networks. This metric, that we dubbed Inter-ISP link density is defined as the ratio between the number of overlay links that connect nodes in two distinct ISPs and all existing links in the overlay having a value between 0 and 1.

Intuitively, an efficient overlay in this scenario should present a low inter-ISP link density. However, it should be above zero to ensure that the overlay remains connected. In a scenario where nodes are associated to more than one ISP, a value of inter-ISP link density of 0 is only possible to achieve if the overlay becomes partitioned. Our experimental results show that a value between 0.1 and 0.2 offers good connectivity while minimizing the number of links that connect nodes in distinct ISPs.

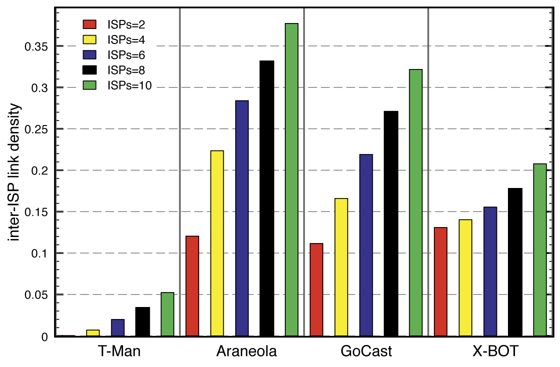


Fig. 5. Fraction of suboptimal links for different protocols and different number of ISPs.

Fig. 5 summarizes the results obtained in this scenario, for all protocols, and for several numbers of coexisting internet service providers. With the exception of T-Man, *X*-BOT is the protocol that achieves lower values for inter-ISP link density. Moreover, for all protocols the inter-ISP link density rises with the number of coexisting ISPs. This is expected as the number of nodes in each ISP becomes lower with the addition of other ISPs leading to a situation where it becomes harder for a node to find peers that are also connected through its ISP. Notice that this effect is less visible in *X*-BOT than Araneola or GoCast. This results from the *X*-BOT strategy that periodically employs its 4-node coordinated optimization technique to continually evolve the overlay topology to a better configuration.

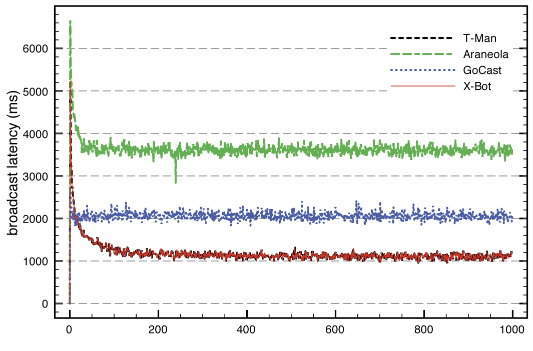
T-Man is able to obtain lower inter-ISP link density values by sacrificing the overlay connectivity. The percentage of nodes in the largest connected cluster in the resulting overlay network is depicted in Table 2. Notice that the percentage of nodes in T-Man is always far below that of 100 percent, and this value lowers with the number of ISPs. In fact this happens because the aggressive strategy of T-Man leads to a situation where nodes associated to a given ISP i, are only connected to other nodes in i, or nodes in other ISPs which neighbors are all associated to that same ISP.

5.3 Message Dissemination

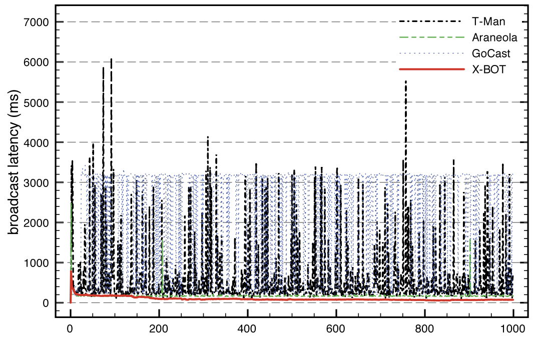
In this section, we evaluate the performance of a gossip-based broadcast protocol [3] operating on top of the overlays resulting from the operation of the different protocols. Considering that the target node degree in the overlay is 5 we set the fanout value of the gossip protocol to

TABLE 2
Percentage of Nodes in the Largest Connected Cluster for Each Protocol for Different Numbers of ISPs

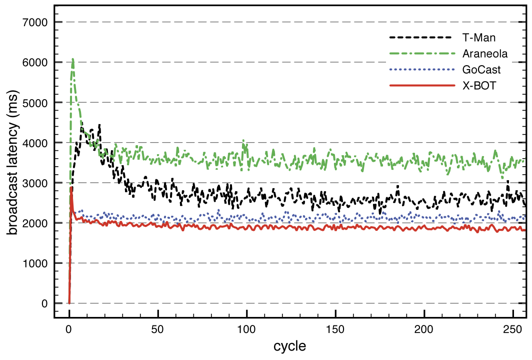
Number of ISPs	T-Man	Araneola	GoCast	*X*-BOT
2	50.011	99.994	100.0	100.0
3	30.380	99.994	100.0	100.0
4	25.050	99.992	100.0	100.0
5	20.082	99.996	100.0	100.0
6	16.780	99.996	100.0	100.0
7	14.184	99.998	100.0	100.0
8	12.664	99.994	100.0	100.0
9	11.282	99.998	100.0	100.0
10	10.156	99.998	100.0	100.0



(a) Cartesian scenario



(b) PlanetLab scenario



(c) Inet-3.0 scenario

Fig. 6. Message dissemination latency using the overlays generated by each protocol for each scenario.

4 (i.e., the largest fanout that prevents a message from being sent more than once on any given link). As before, the values presented are an average of five independent experiments. Link latency is captured by the link cost. The event-based engine of PeerSim was used to implement the broadcast protocol and introduce latency to message dissemination based on that metric.

Figs. 6a, 6b, and 6c depict the broadcast latency (i.e., the amount of time required to deliver a message to the maximum of participants) for each protocol.[9] Only T-Man is able to provide better latency than *X*-BOT. This only happens because T-Man is not able to provide a broadcast reliability of 100 percent as its overlay becomes disconnected. This is notorious in the PlanetLab and inet-3.0 scenarios, which have a nongaussian link cost distribution (see Table 3). The good performance of *X*-BOT is due to its

9. In the supplementary document [16], the interested reader can find a breakout of Fig. 6b, for better readability.

TABLE 3
Comparision of Broadcast Latency and Reliability Using Overlays Generated by the Different Protocols

Cartesian Scenario				
	T-Man	Araneola	GoCast	X-BOT
Latency (ms)	1238.0	3701.8	2032.8	1165.2
Reliability (%)	99.328	99.994	100	100
PlanetLab Scenario				
	T-Man	Araneola	GoCast	X-BOT
Latency (ms)	711.8	162.8	3180.8	72.0
Reliability (%)	49.0322	100	100	100
Inet-3.0 Scenario				
	T-Man	Araneola	GoCast	X-BOT
Latency (ms)	2545.2	3517.0	2108.0	1879.8
Reliability (%)	13.806	100	99.99996	100

capacity to improve the overlay efficiency while preserving the in-degree distribution and connectivity (as shown in the Section 5.2.2). Notice that in the PlanetLab scenario, GoCast and T-Man latency exhibits several spikes. This is due to adaptations of the overlay affecting node degrees, which can increase the overlay diameter resulting in additional latency.

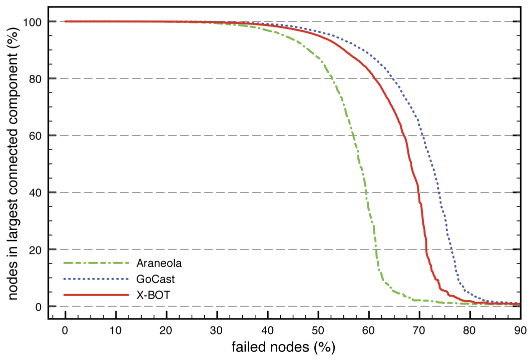
To provide a better comparison among protocols, Table 3 shows both the latency and the reliability values of the message dissemination runs. In all scenarios, *X*-BOT offers better reliability with a lower latency. This clearly shows that *X*-BOT, when equipped with a latency oracle, offers a better support to implement gossip-based broadcast protocols. Notice that as described above, T-Man is unable to achieve a reliability of 100 percent due to the fact that the overlay becomes disconnected as a result of the protocol operation.

5.4 Fault Tolerance

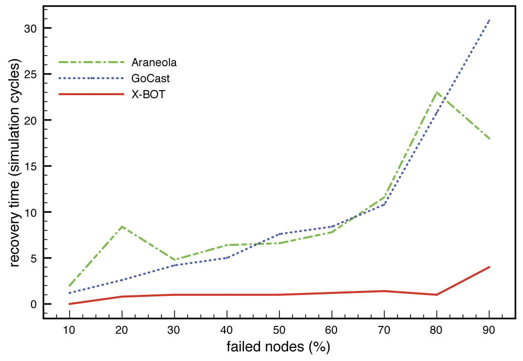
In this section, we evaluate the resilience and healing capabilities exhibited by each protocol in face of node failures. We assume that nodes can fail by crashing, and that TCP enables a node to detect these failures after a small amount of time (the following simulation cycle). Due to space limitations, we only present results for the cartesian scenario. Moreover, we omit T-Man given that the overlay becomes disconnect in steady state.

Fig. 7a plots the percentage of correct nodes in the largest connected overlay component as nodes fail one by one. In these simulations protocols were not allowed to take any corrective measures. *X*-BOT offers better connectivity in face of failures when compared with Araneola. However, GoCast connectivity surpasses that of *X*-BOT. This is not surprising, giving that GoCast maintains a significant number of nodes with degree above 5 (unfortunately, as discussed previously, this feature has a negative impact on the performance of gossip-based broadcast protocols).

We then evaluated the time required by each protocol to recover from massive failures that range from 10 to 90 percent of simultaneous nodes crashes. To this end, we measured the number of simulation cycles required, in average, for the broadcast protocol to regain its previous (or maximum) reliability values after the induction of failures. Whereas in general *X*-BOT always regains a reliability of 100 percent, the same is not true for other protocols. Results are depicted in Fig. 7b. *X*-BOT is able to recover from failures much faster. This is due to the design of *X*-BOT which, unlike GoCast or



(a) Connectivity



(b) Recovery Time

Fig. 7. Overlay resilience to node failures in terms of connectivity and time required to recover.

Araneola, promotes connectivity, by avoiding optimization rounds when nodes do not have a full active view.

6 Conclusions

In this paper, we proposed and evaluated *X*-BOT, a new protocol that allows an unstructured overlay network to bias its topology according to a target efficiency criteria. The challenge addressed in the present paper was to improve the overlay topology by selecting better links without loosing the relevant properties of the original overlay (such as the low clustering coefficient, in-degree distribution, and high-failure resiliency and recovery).

Experimental results show that *X*-BOT is able to improve the overlay topology to more efficient configurations than previous approaches in several distinct scenarios. A significant feature of *X*-BOT is its ability to promote overlay connectivity, by preserving node degrees. As a result, *X*-BOT is able to support efficient and resilient gossip-based broadcast solutions when equipped with an appropriate oracle (e.g., a latency oracle). Moreover, *X*-BOT is able to recover from failures faster than previous proposed solutions.

Acknowledgments

The authors wish to express their gratitude to the anonymous reviewers for the useful comments and suggestions. This work was partially supported by Fundação para a Ciência e Tecnologia (FCT) under project (PEst-OE/EEI/LA0021/2011) and via the project "HCPI" (PTDC/EIA-EIA/102212/2008).

References

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

[2] A.-M. Kermarrec, L. Massoulié, and A. Ganesh, "Probabilistic Reliable Dissemination in Large-Scale Systems," *IEEE Trans. Parallel and Distributed Systems*, vol. 14, no. 3, pp. 248-258, Mar. 2003.

[3] J. Leitão, J. Pereira, and L. Rodrigues, "HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast," *Proc. IEEE/IFIP 37th Ann. Int'l Conf. Dependable Systems and Networks (DSN '07)*, pp. 419-429, 2007.

[4] M.-J. Lin and K. Marzullo, "Directional Gossip: Gossip in a Wide Area Network," *Proc. Third European Dependable Computing Conf. (EDCC)*, pp. 364-379, 1999.

[5] M. Jelasity and A. Montresor, "Epidemic-style Proactive Aggregation in Large Overlay Networks," *Proc. 24th Int'l Conf. Distributed Computing Systems (ICDCS '04)*, pp. 102-109, 2004.

[6] P. Eugster and R. Guerraoui, "Probabilistic Multicast," *Proc. IEEE/IFIP Ann. Int'l Conf. Dependable Systems and Networks (DSN)*, 2002.

[7] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "From Epidemics to Distributed Computing," *Computer*, vol. 37, no. 5, pp. 60-67, May 2004.

[8] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Epidemic Algorithms for Replicated Database Maintenance," *Proc. Sixth Ann. ACM SIGACT-SIGOPS Symp. Principles of Distributed Computing (PODC)*, pp. 1-12, 1987.

[9] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling Churn in a Dht," *Proc. USENIX Ann. Technical Conf.*, pp. 10-10, 2004.

[10] M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen, "The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations," *Proc. Fifth IFIP/ACM Int'l Conf. Distributed Systems Platforms (Middleware)*, pp. 79-98, 2004.

[11] A. Ganesh, A.-M. Kermarrec, and L. Massoulié, "SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication," *Proc. Third Int'l COST264 Workshop Networked Group Comm.*, pp. 44-55, 2001.

[12] S. Voulgaris, D. Gavidia, and M. Steen, "Cyclon: Inexpensive Membership Management for Unstructured P2P Overlays," *J. Network and Systems Management*, vol. 13, no. 2, pp. 197-217, June 2005.

[13] K. Birman, "The Promise, and Limitations, of Gossip Protocols," *SIGOPS Operating Systems Rev.*, vol. 41, no. 5, pp. 8-13, 2007.

[14] S. Girdzijauskas, A. Datta, and K. Aberer, "Oscar: Small-World Overlay for Realistic Key Distributions," *Proc. Int'l Conf. Databases, Information Systems, and Peer-to-Peer Computing (DBISP2P)*, pp. 247-258, 2007.

[15] H. Rostami and J. Habibi, "Topology Awareness of Overlay P2P Networks," *Concurrency and Computation: Practice & Experience-Autonomous Grid*, vol. 19, pp. 999-1021, May 2007.

[16] J. Leitão, J.P. Marques, J.O. Pereira, and L. Rodrigues, "X-BOT: A Protocol for Resilient Optimization of Unstructured Overlay Networks (Supplementary Document)," http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.29, 2011.

[17] Y.-H. Chu, S. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," *IEEE J. Selected Areas in Comm.*, vol. 20, no. 8, pp. 1456-1471, Oct. 2002.

[18] I. Gupta, A.-M. Kermarrec, and A. Ganesh, "Efficient and Adaptive Epidemic-Style Protocols for Reliable and Scalable Multicast," *IEEE Trans. Parallel Distributed Systems*, vol. 17, no. 7, pp. 593-605, July 2006.

[19] L. Massoulie, A.-M. Kermarrec, and A.J. Ganesh, "Network Awareness and Failure Resilience in Self-Organising Overlay Networks," *Proc. 22nd Symp. Relizable Distributed Systems (SRDS)*, pp. 47-55, 2003.

[20] C. Tang and C. Ward, "GoCast: Gossip-Enhanced Overlay Multicast for Fast and Dependable Group Communication," *Proc. IEEE/IFIP Ann. Int'l Conf. Dependable Systems and Networks (DSN)*, pp. 140-149, 2005.

[21] R. Melamed and I. Keidar, "Araneola: A Scalable Reliable Multicast System for Dynamic Environments," *Proc. IEEE Third Int'l Symp. Network Computing and Applications (NCA)*, pp. 5-14, 2004.

[22] M. Jelasity and O. Babaoglu, "T-Man: Gossip-Based Overlay Topology Management," *The Fourth Int'l Workshop Eng. Self-Organizing Applications*, May 2006.

[23] J. Leitão, N.A. Carvalho, J. Pereira, R. Oliveira, and L. Rodrigues, *Handbook of Peer-to-Peer Networking*, ch. 10, Springer, 2010.

[24] P. Karwaczyński, D. Konieczny, J. Moçnik, and M. Novak, “Dual Proximity Neighbour Selection Method for Peer-to-Peer Based Discovery Service,” *Proc. 22nd ACM Symp. Applied Computing (SAC)*, pp. 590-591, 2007.
[25] M. Jelasity, A. Montresor, G.P. Jesi, and S. Voulgaris, “The Peersim Simulator,” http://peersim.sf.net, 2012.
[26] J. Winick and S. Jamin, “Inet-3.0: Internet Topology Generator,” Technical Report UM-CSE-TR-456-02, EECS, Univ. of Michigan, 2002.

João Leitão received the graduate degree in 2006 and the master’s degree in computer engineering in 2007 from the Faculty of Sciences of the University of Lisbon. He is currently working toward the PhD degree in computer engineering at Instituto Superior Técnico of the Technical University of Lisbon, and conducts his research in the context of the Distributed Systems Group of INESC-ID Lisboa. His research interests include design of fault-tolerant and efficient large-scale and peer-to-peer systems, in particular, in the design of overlay networks. He is a member of the ACM and the IEEE.

João Pedro Marques is an undergraduate student of computer engineering at the Instituto Superior Técnico of the Technical University of Lisbon. He also was a junior researcher in the Distributed Systems Group of the INESC-ID Lisboa.

José Pereira received the graduate degree in 1995 in computer engineering and informatics, the master’s degree in 1998 and the PhD degree in 2002 in computer science from the University of Minho. He is now an assistant professor at the Computer Science Department of the University of Minho. He has been doing research in reliable distributed systems, in particular, in large-scale group communication and dependable data management. He is a member of the IEEE.

Luís Rodrigues is a professor Catedrático at the Departamento de Engenharia Informática, Instituto Superior Técnico, Universidade Técnica de Lisboa and a senior researcher at the Distributed Systems Group of INESC-ID Lisboa. His current research interests include distributed fault-tolerance, replicated data management, peer-to-peer computing, and mobile computing. He has more than 150 publications in these areas. He is coauthor of two books on distributed computing. He is a senior member of the ACM and the IEEE.