GoDaCen: Optimizing Gossip for Use in Datacenters

Miguel Jorge Cardoso Branco
miguel.branco@ist.utl.pt

Instituto Superior Técnico
(Advisor: Professor Luís Rodrigues)

Abstract. Gossip, protocols have emerged to serve as building blocks to applications with high reliability requirements. However, a generic gossip solution may not be sufficient to ensure competitive performance in specific topologies such as in datacenter networks. This work addresses the problem of optimizing gossip-based protocols for use in datacenters. It starts by making a survey of different techniques that have been proposed to exploit the topology of the underlying network when optimizing the operation of gossip protocols. Then the topology of datacenter networks is studied and abstracted. Based on this analysis, the report discusses why existing gossip optimization techniques may provide unsatisfactory results when running on datacenter networks and identifies some directions of research to improve the state-of-art in this respect.

1 Introduction

Gossip, or epidemic, protocols are based on the periodic exchange of information between pairs of nodes [1]. Originally proposed for database replication [1, 2], gossip protocols have proven to be an effective building block in several other applications, such as reliable message dissemination [3, 4], data aggregation [5, 6], failure detection [7], among others [8]. One of the most attractive properties of epidemic protocols is their inherent scalability, as the load imposed on each individual node is typically logarithmic with respect to the total number of nodes in the system [9].

Gossip based protocols have been used in the implementation of several practical systems, including systems that support a very large number of users such as Amazon's Dynamo [10] and Facebook's Cassandra [11]. In this context, gossip is used to maintain the system membership, detect failures, and replicate control state. These systems operate on large datacenters, which comprise large numbers of off-the-shelf machines that are prone to failures. Gossip based protocols are appealing in this context not only due to their scalability properties, but also because they are very robust to failures of individual nodes.

Part of the robustness of gossip-based protocols derives from redundancy. In each gossip round, each node only interacts with a small number of other nodes, typically selected at random. However, the total number of interactions in the system is usually very large, and a given node (or link) is not unlikely to receive (transport) the same gossip message multiple times. Therefore, gossip protocols may stress the network in an undesirable manner. This problem may be exacerbated if the gossip protocol is oblivious to the underlying network topology. In fact, if the network is abstracted by a fully connected *clique* (*i.e.*, a single group of fully connected nodes that are able to communicate directly between themselves), gossip exchanges among disjoint sets of nodes may appear completely independent. However, different overlay links may use the same equipment at the underlay level.

One way to mitigate the potential negative effect of gossip on the underlying network is to design gossip based protocols that are *network aware*, *i.e.*, where gossip exchanges are explicitly biased in an attempt to ensure better network utilization. Although different techniques to take into account the network topology in the operation of gossip-based protocols have been proposed [12–14], it is unclear which of them are more suitable to optimize gossip operation on top of the network topologies used in current and future datacenters. For that reason, we propose to study both network aware gossip protocols and datacenter topologies with the aim of identifying which techniques are more appropriate for this particular setting. Our goal is to design, implement, and evaluate a gossip-based protocol optimized for the network topologies of datacenters.

The rest of the report is organized as follows. Section 2 briefly summarizes the goals and expected results of our work. In Section 3 we present the fundamental concepts and survey previous work which is closer to our own. We analyze current and proposed datacenter topologies in Section 4. Section 5 describes the proposed solution to be developed and implemented, whereas Section 6 describes how we plan to evaluate our solution and validate it. Finally, Section 7 presents the schedule of future work and Section 8 concludes the report.

2 Goals

This work addresses the problem of optimizing the operation of gossip protocols for datacenter topologies. Our goal is to devise a protocol that can serve as a building block to support gossip-based services in datacenters with the assurance that they will behave in a scalable, efficient, robust, and predictable fashion, while exploiting the properties of the underlying network. Specifically, the goals are:

Goals: Research, implement, and test a novel gossip protocol that is optimized for both current and future datacenter topologies.

To that end, we need to compare the abstractions currently employed to model existing networks and, if necessary, define a new network model that captures the specific nature of datacenter networks.

The project will produce the following expected results:

Expected results: The work will produce i) an abstract model for datacenter topologies that can be used to optimize the operation of gossip-based protocols; ii) a specification of a gossip protocol optimized for datacenter networks; iii) an implementation of the designed protocol, and iv) an extensive experimental evaluation using simulations.

3 Related Work

In this section we survey the main techniques proposed in the literature to render the operation of gossip-based protocols *network aware*, *i.e.*, to exhibit communication patterns that better match the topology of the underlying network. The section starts by describing the characteristics that define a gossip protocol that is oblivious to the underlying network topology, showing the general applications of such protocols. Then, we present useful metrics to evaluate the performance of gossip-based systems, before identifying the properties of the underlying network that are relevant for the operation of epidemic protocols. After that, we identify two general strategies to optimize gossip-based solutions with regard to the network properties. Finally, we describe a number of existing gossip systems, and discuss how these systems take the previously identified network properties into consideration.

3.1 Network Oblivious Gossip

We will now introduce the most common characteristics of gossip protocols that do not take the underlying network topology into consideration. We will start by defining a simple gossip protocol (named *Flat Gossip*) before we incrementally present typical strategies employed to increase the benefits of epidemic protocols in more complex scenarios. Finally, we will show how to leverage the introduced properties to build applications that use gossip as a building block.

3.1.1 Flat Gossip

Gossip (or epidemic) protocols are named after the social or natural process they mimic in order to disseminate information in a network of computer nodes: their operation is inspired by the way a rumor, or a disease, spreads in a population. Typically, these protocols present a periodic behavior, which can be modeled as a sequence of *gossip rounds*. In each gossip round, a node exchanges information with a predefined number f of other nodes selected uniformly at random from the entire system population (the parameter f is called the fanout). Therefore, if a node has some novel information that needs to be disseminated, after one gossip round this information is also known by other f nodes. Since all nodes engage in gossip rounds periodically, the number of nodes that know the information (sometimes referred as *infected* nodes) grows exponentially with the number of executed rounds. As a result, in this setting, the time required for a piece of information to be propagated to all nodes in the system is, on average, $O(log(N))$, where N is the total number of nodes executing the protocol [9].

This protocol is known as flat gossip exactly because gossip exchanges are performed with nodes selected uniformly at random, with no concern for their location in the underlying topology. Implicitly, the protocol assumes that each node has access to the full system membership (such that it can select gossip peers uniformly). In very large dynamic systems, to maintain full membership at every node becomes prohibitively expensive, given that every join and leave of individual nodes would need to be propagated globally. Therefore, many practical flat gossip protocols operate based on partial membership information, *i.e.*, they rely on the availability of a companion *partial membership service*, that offers to each node a sample view of the full membership [15, 16, 4]. Protocols based on partial membership exhibit the same properties of gossip protocols based on full membership as long as local views include a uniform sample of the full membership [17].

3.1.2 Partial Membership Services

In the context of gossip-based protocols, the goal of a companion partial membership service is to provide each node with a sample of the entire system membership. The main purpose of using such a service is to increase the scalability of the system. In fact, any large-scale system is faced with system dynamics, *i.e.*, some nodes depart from the system (for instance, due to failures, maintenance, etc.) and new nodes are added to the system. This phenomenon is known as *churn* [18]. To propagate these changes to all other nodes in a timely manner would consume a significant amount of resources. By providing nodes with partial views, the effects of churn can be localized.

A partial membership service, also called a *peer sampling service* [19], aims at providing each node with a uniform sample of the entire system membership, called a *local* or *partial* view. Usually, the size of the local view is larger than the fanout value f. Thus, in its operation, the gossip protocol selects f nodes at random from its local view, instead of selecting f nodes at random from the entire membership. Therefore, when a partial membership protocol is used, gossip no longer operates on an overlay that is a fully connected clique; instead, it operates on an overlay network with topological properties that derive from the way partial views are built.

There are two main approaches for maintaining partial views. The *reactive* approach only changes the partial view of a node in response to a change in the membership of the entire system (i.e., when nodes leave or join) [12]. A *cyclic* approach continuously shuffles the partial membership information maintained by the nodes [16]. Finally, there are also protocols that combine features of the two previous approaches [4].

3.1.3 Gossip Strategies

In the previous sections we have stated that gossip rounds happen periodically and that in each round a node exchanges information with other nodes. Obviously, this is a simplified and abstract description of how gossip may operate, and does not capture all the alternative implementations that have been described in the literature. A more detailed description of the possible strategies for information exchange can be made by considering the gossip exchange unidirectional and by distinguishing the node that triggers the exchange (the *initiator*) from the node that is contacted (the *target*).

In this more refined setting, if the information flows from the initiator to the target we say that the gossip protocol operates in *push* mode. Otherwise, we say that gossip operates in *pull* mode. Naturally, some protocols exchange information in both directions, thus combining push and pull in each round, in what is called a *push-pull* approach.

Furthermore, when the initiator sends new information to the target without first inquiring which information the target already knows, we state that the protocol operates in *eager push* mode. If the initiator first requests the target to return the list of missing information, and only later sends the actual data, we state that the protocol operates in *lazy push.* Lazy push may be useful if the information to be transmitted is large and its availability at the target can be checked by exchanging first small identifiers. In this case, lazy push saves network bandwidth in exchange for a longer latency in the information dissemination.

Finally, a push protocol can also operate in pure reactive mode, i.e., a node may initiate a gossip round as soon as it receives new information, instead of waiting for a predefined interval.

3.1.4 Practical Convergence Techniques

Another practical challenge that arises from a simplistic view of gossiping local information with peers is how to manage a large amount of information, as is the case with a broadcast service that was used to disseminate many messages, or a replication system with many objects suffering constant updates. To achieve convergence (*i.e.*, ensuring that every node has the same information), one has to take into account that it may be impractical for a node to transmit all the information it has received since the startup of the system.

Epidemic convergence approaches are typically divided into two separate categories [20]: *anti-entropy* and *rumor mongering.* In anti-entropy systems, nodes gossip the most recent changes to their state and merge any differences found, whereas in rumor mongering systems, nodes gossip new information for a limited number of gossip rounds. A key aspect to take into consideration is that while nodes in an anti-entropy system continuously gossip their state, nodes in a rumor mongering system stop gossiping in the absence of new information.

3.1.5 Common Gossip Applications

After introducing the general flat gossip protocol and some additional features that are usually employed to increase scalability in practical scenarios, we will now demonstrate how gossip can be used as a building block for two example applications. The proposed examples are a reliable broadcast application and an anti-entropy state reconciliation application.

Reliable Broadcast An epidemic protocol to disseminate information reliably could work using a reactive push strategy as described above. To account for peers that did not receive the message in the push dissemination, the nodes can maintain a cyclic lazy push strategy with fanout f, periodically sending the IDs of messages they know to f neighbors, which in turn request the messages that they did not yet receive.

As the ID summaries grow with the number of messages, it is useful to limit the number of rounds a message is kept in the lazy push summary before its memory can be reclaimed, similarly to a rumor mongering state reconciliation strategy. Such an approach can be seen in [3].

Anti-entropy State Reconciliation An example of an anti-entropy state reconciliation protocol is proposed in [20], a system in which nodes keep replicas of each others' state. To update their vision of the state of a peer p, a node gossips with another (not necessarily p), exchanging the maximum version number they have for objects of p. Then, should the version numbers differ, the node with higher version number for p sends the other as many updates for p objects as it can (*i.e.*, with regard to the maximum gossip message size), in chronological order.

The chronological order and the limit of the number of updates mitigate the exchange of outdated information that could occur when the two gossiping nodes have different versions but neither of them is up-to-date. Because both nodes will have to gossip with an up-to-date node to receive the latest state, they should not be wasting resources exchanging their outdated values.

3.2 Metrics for Evaluating Gossip Protocols

To evaluate the performance of an epidemic protocol, one needs metrics that can grade the system on its key features. Such metrics are useful not only to know what could be improved in the solution being studied but also to compare it to other *baseline* solutions, to gain some insight on whether the system improved the desired properties or not.

We now present some metrics to evaluate key properties of gossip protocols, noting the distinction of their implementation in broadcast systems and state reconciliation systems.

Reliability The robustness of the protocol is one of the key properties of these systems, so it is commonly evaluated. In broadcast protocols, robustness is typically measured in terms of *reliability*, defined as the percentage of nodes which receive a given message (*i.e.*, the number of infected nodes). In state reconciliation protocols, reliability can be measured in terms of the percentage of nodes that converged to a final state.

Reliability is closely tied to the number of peers to which nodes disseminate each piece of information. Therefore, the fanout parameter of an epidemic protocol is a fundamental factor when considering the reliability of the system. Since gossip protocols follow a *bimodal behavior*, either only a negligible subset of nodes receives the information or almost all nodes do [21]. Reliability should then be tested in both stable scenarios and in the presence of multiple faults, when the number of nodes that receive the information decreases to values that could be potentially insufficient to ensure full dissemination.

Latency The *latency* of messages is also an interesting metric for these systems. It is typically defined in broadcast systems by the amount of time that messages take to infect all the nodes in the network. As a simplification, latency can also be measured in terms of the *last delivery hop* (LDH) [14] of a message. In epidemic broadcast protocols, LDH is the number of gossip rounds that a message took to be delivered to all nodes. Please note that this metric can easily be compared to the latency (in time units) by multiplying LDH for the gossip round time duration. As for state reconciliation protocols, the latency of the system can be measured as the time it takes for all nodes to converge on a final state.

Redundancy As gossip protocols tend to generate redundant information to mask node failures, it is also useful to measure the link stress in terms of messages per link, to determine the overhead on the network.

Another strategy to identify the redundancy penalty specifically is counting the redundant messages that are transfered for each new piece of information. If we divide the number of redundant messages by the number of nodes in the system, we discover how many useless messages each node received, on average. An issue with such an approach can arise when not all the nodes receive the information, skewing our metric into displaying a false average penalty. For that reason, the *relative message redundancy* (RMR) was proposed in [14]. RMR is given by the expression $\frac{m}{n-1}-1$, where m denotes the number of message payloads sent during the dissemination of one message and n denotes the number of nodes that actually received the message. This metric measures the average number of message copies (besides the first) that each node received and thus the overhead of the dissemination process, accounting for scenarios where not all the nodes received the information.

3.3 Modeling Heterogeneity

When using flat gossip, an initiator selects gossip targets at random, with no regard for the heterogeneity that may exist among the nodes and the links that connect them. Naturally, it is unlikely that in practice all gossip exchanges can be executed with the same efficiency, as different nodes and different links may have different properties. Therefore, flat gossip may ignore fundamental aspects that can affect the performance of the protocol in a realistic setting.

In the following paragraphs, we identify the relevant properties that may affect the execution of gossip protocols. We will divide these properties into three groups: *Link Properties*, *Node Properties* and *Overlay Topology Properties*.

3.3.1 Link Properties

A link at the overlay level is implemented at the underlay level by one or more paths that are materialized by physical media and network equipment such as network cards, routers, etc. Given the diversity of technologies available to implement a network link, the physical distance among the nodes, the number and profile of other applications that share the same paths, etc, the observed behavior of the overlay links may vary substantially. The most important differences may be abstracted as follows:

Latency The latency can be defined as the elapsed time between the beginning of a transmission by the sender and the receiver completely receiving the information. Link latency is often the dominating factor in the time required to finish a distributed computation, given that in most cases processing time is negligible when compared with latency. This is especially true when links connect nodes placed in geographically remote locations. A particular case of a distributed computation is request-reply operation. In this case, it is often relevant to measure the *round trip time*, which is twice the latency as defined above.

In reactive push based gossip-protocols, by tuning the operation of a gossip protocol to use more frequently links with low latency, it may be possible to speed up the dissemination of information in certain scenarios [13, 22].

Loss Rate Loss rate is defined as the fraction of messages sent through the link that are not successfully transmitted. If the gossip protocol is implemented on top of a reliable transport protocol such as TCP, the loss rate is masked at the transport protocol and transformed into latency (as the information is automatically retransmitted at the transport level). If the gossip protocol is implemented on top of an unreliable transport protocol such as UDP, gossip must be configured such that the redundancy in the gossip operation is enough to mask the loss rate and maintain the reliability values. The operation of the gossip protocol can be tuned to take heterogeneous loss rates into consideration, for instance by changing the frequency of gossip on lossy links [23].

Link Capacity The capacity of a link is defined as the number of bits that it can transmit per unit of time. If at a given point in time the information that needs to be sent over a link exceeds its capacity, the information needs to be temporarily queued, introducing additional latency in the communication. If the application keeps producing new information at a pace that cannot be satisfied by the capacity of the links, one needs to resort to some form of flow-control mechanisms, to throttle the application rate. Typically, the capacity of a network is limited by the aggregate capacity of the links that constitute its minimum cut. In some cases, better performance can be obtained if the gossip rate is limited to match the aggregate capacity of the network [20, 24]. Note that the link capacity as observed by the gossip protocol may be smaller than the actual capacity at the physical level. This happens because physical links are often shared by many different applications and users. Therefore the available capacity needs to the split among the applications that use the link.

Link Independence Typically, it is simpler to consider that two different overlay links are independent from each other. If links are independent this means that the way the protocol uses a given link does not affect the properties of another link. For instance, sending messages on a link does not change the capacity of the remaining links. However, it is often the case that different links at the overlay level share physical components at the physical level. In this case, links are not independent. In this scenario, the aggregate capacity of a set of overlay links may be fixed and, by sending messages on one link, the available capacity of the other links is reduced, increasing the risk of queuing messages at the underlay link's routing equipment. The knowledge about link dependencies may be used by the protocol to reduce the likelihood that related links are used simultaneously [12, 23].

3.3.2 Node Properties

Node properties (and therefore the differences among nodes) also have an impact on the system's behavior. We now discuss two characteristics of nodes that are relevant in the context of gossip protocols, *capacity* and *expected uptime*.

Node Capacity In most practical cases, not all nodes that participate in a gossip protocol have the same hardware configuration and run the same software. Differences in CPU power, available memory, among others, make that some nodes are able to engage in more gossip exchanges than others. In an abstract manner, the capacity of a node may be measured by the number of gossip exchanges which that node is able to perform with other nodes in a single gossip round. The node capacity may be taken into consideration, for instance, by assigning different fanout values to different nodes, or employing an eager push strategy more often when nodes with higher capacity are involved [25].

Expected Uptime The nodes that engage in a gossip protocol may abandon the system due to crashes, software failures, maintenance activities, or simply because their users decide to unplug the node. Often, not all nodes have the same probability of leaving the system, as some hardware may be less reliable, some software more prone to bugs, etc. If these factors can be estimated during the execution, the gossip protocol may bias its operation to favor more reliable nodes. At the overlay level, node failures can be difficult to distinguish from link losses, and therefore some solutions like [23] work for both.

3.3.3 Overlay Topology Properties

As we have noted before, when the gossip protocol operates with full membership, the overlay is a fully connected clique. Therefore, the topology has no particular impact on the way the information is spread. On the other hand, when using partial membership, the overlay defined by partial views may have some topological properties that should be considered when executing the gossip protocol.

Heterogeneous Degrees In graph theory, the number of links that a node owns to other nodes is called the *degree* of that node. If the graph is directed, which in gossip protocols would translate to the neighbor set being asymmetrical (*i.e.*, node A having node B as a neighbor does not ensure that node B will have node A as his neighbor), the *indegree* of a node is the number of incoming links to it, and the *outdegree* is the number of links it owns to other nodes.
If the node degree is not maintained, some nodes may become more easily disconnected from the network in the presence of faults, due to lack of neighbors (and, conversely, nodes to which they are neighbors) [4, 22].

Heterogeneous Connectivity In a clique, the number of distinct paths between any pair of nodes is exactly the same. Therefore, if all nodes use the same protocol, the probability of information reaching a node in a given round is also the same for every node.
When a different topology is used, this is no longer true. In fact, the number of distinct paths among different pairs of nodes may vary greatly. Consider the example in Figure 1: all paths from nodes in the clique to node 1 include node 3. However, if node 3 selects gossip targets at random, the probability of new information, generated in the clique, to remain in the clique, is larger than the probability of this information being disseminated to node 1. The gossip protocol can bias the target selection process to ensure a more uniform spread of the information, despite asymmetries in the overlay [23].

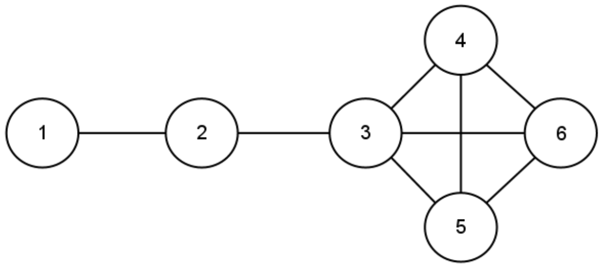

Fig. 1. An example of a topology with heterogeneous connectivity

Hierarchical Overlay In some cases, the overlay defined by the partial views may be organized in a hierarchical manner. This happens when the views reflect the administrative structure of the system where gossip operates or are correlated with the hierarchical nature of the underlying network. In this case, the overlay is organized in horizontal layers, and nodes in a given layer can only communicate with other nodes in the same layer or in adjacent layers. Figure 2 illustrates such an example. Hierarchical topologies are normally used for data aggregation in systems such as the one presented in [6].

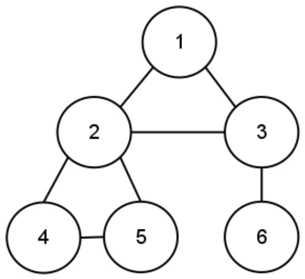

Fig. 2. An example of a hierarchical topology

3.4 Taking Heterogeneity into Account

As we have seen, the heterogeneous properties of links and nodes, as well as the properties of the overlay topology, may affect the operation of the gossip protocol. There are mainly two ways of addressing this heterogeneity, namely, by acting at the level of the gossip protocol, biasing the gossip targets; or by acting at the level of the companioning membership service, biasing the composition of partial views. Each of these alternatives is discussed in the next paragraphs.

3.4.1 Biasing the Gossip Target

A common approach to integrate topology awareness in epidemic protocols is to associate weights to neighbors and use those weights to bias the probability with which a node chooses each neighbor as a gossip target [23, 25, 14]. The weight can capture a *utility function*, in which case the probability of choosing a given neighbor for gossiping is directly proportional to the weight; or a *cost function*, in which case the probability will be inversely proportional to the weight.

Typically, the computation of the utility/cost function is encapsulated within an architectural component commonly named *Oracle*. For instance, an oracle that gives information about the latency of the links could exchange PING messages with neighbors and register the observed round trip time so as to estimate link latency. Alternatively, another implementation of an oracle could measure the hops from a node to another using TRACEROUTE and thus provide a cost function in terms of hops instead of time units.

3.4.2 Biasing the Partial Views

Instead of biasing the gossip target selection, a different approach is to change the membership service in order to bias the local views provided to each node [12, 13, 22]. This can affect the overlay topology in different manners: for instance, higher capacity nodes may be assigned a higher number of neighbors than low capacity nodes; neighbors can be selected such that overlay links have a lower average cost, etc.

It is worth noting that the biasing of the overlay topology can bring both benefits and disadvantages. If performed in the wrong manner, biasing the partial view may introduce undesirable features. One can easily introduce a *clustering* effect in the neighboring relations. This might happen because nodes usually try to bias the neighboring selection using transitive properties. For instance, taking into consideration the network latency between nodes. This can lead nodes to organize themselves in a clustered way, where small and highly connected groups of nodes become weakly connected among them. It has been shown that clustering can have a negative effect on the time required for information to spread across the whole system [23], as well as on the connectivity of the system, increasing the probability of network partitions [26], especially in the presence of node failures.

3.5 Existing Systems

We now survey a number of relevant gossip-based protocols that illustrate several of the aspects that have been identified in the previous paragraphs.

3.5.1 Directional Gossip

Directional Gossip [23] is a reliable multicast gossip system that tries to improve Wide Area Network (WAN) gossip in two separate ways.

First, it distinguishes WAN gossip from Local Area Network (LAN) gossip, as to not stress routers that connect the different LANs. The distinction is made by employing a hierarchical structure of gossip levels as introduced in section 3.3.3. At the WAN level, Directional Gossip runs a single, but optionally replicated, gossip server, instead of delegating to the LAN gossip nodes the task of disseminating messages to the other LANs.

Second, it addresses the issue of poorly connected nodes that we also discussed in section 3.3.3. It does so by leveraging a neighbor bias strategy, assigning weights to each neighbor, where each weight is the number of link-disjoint paths known to exist between the node and the neighbor. Two paths are link-disjoint if they do not share any link at the overlay level.

It then selects the gossip target with probability inversely proportional to the weight of the neighbor. To increase reliability, Directional Gossip also floods (*i.e.* sends a received message without waiting for the gossip round time) the neighbors with weight lesser than a preset value k.

Additionally, the protocol also considers techniques to deal with heterogeneous link loss rates, proposing an alternative weight function that takes the loss rates of the links into consideration and gives less importance to links that fail with high probability.

The reactive flooding of each message presents an overhead in link stress when compared to cyclic gossip solutions. However, the overhead is still smaller than using a pure flooding solution, where each node sends the message to its entire neighbor list instead of identifying critical neighbors.

3.5.2 HiScamp

HiScamp [12] is a partial membership service proposed to alleviate stress in core routers, using a hierarchical structure. The system organizes nodes into clusters according to a proximity measure. Conceptually, each cluster represents a single node in a higher level gossip layer.

It then restricts the partial views of nodes such that, at the higher level, each cluster maintains at most one single link to each of the other clusters. This translates to the lower level in the following manner: for any two clusters of nodes, for instance cluster a and cluster b, there is only one node in cluster a with a link to a member of cluster b. This way, there is less traffic between clusters, as the cluster members divide among themselves the responsibility of sending information to other clusters. These levels form a hierarchy of arbitrary depth, repeating the responsibility division at each level.

It is interesting to note that the degree of the nodes updates dynamically with the number of nodes and clusters in the system, using a subscription process on node join and an unsubscription process on node departure. Although nodes can leave the system without executing the unsubscription process, the authors claim that a subscription lease system can be implemented as proposed in [15].

This approach reduces network load on high latency links but requires a good proximity metric as soon as the time of joining the system. Additionally, the proximity metric is not dynamic.

The inherent tradeoff in this solution is that although the link stress between clusters is reduced, the average latency grows with the number of hierarchy levels and the message broadcast displays less reliability than a flat gossip protocol.

3.5.3 GoCast

GoCast [13] is a reliable multicast gossip system that builds a multicast tree to disseminate messages quickly through the links with lower latency, using an eager push approach. The system uses a membership restriction strategy to build neighbor lists comprised of C_{near} nearby neighbors and (fewer) C_{rand} random neighbors. Although a node has $C_{near} + C_{rand}$ neighbors in the overlay, only a subset of those links are used as multicast tree links.

If a branch of the tree fails, nodes can still receive messages through a periodic lazy push gossip by their other overlay neighbors (peers in the neighbor list that are not connected by tree links, only by overlay links). A node gossips the message IDs to each of its purely overlay neighbors using a rumor mongering approach with the duration of only one round.

Based on results from experiments, the authors conclude that the optimal values for the neighbor parameters are $C_{near} = 5$ and $C_{rand} = 1$, claiming that one random neighbor is sufficient to provide almost the same connectivity guarantees as larger random membership views. The degree of the nodes is aggressively maintained and, thus, homogeneous. The degree maintenance process ensures symmetrical neighboring sets, and the same multicast tree is used for every dissemination, regardless of the source.

3.5.4 Payload Scheduler

Nuno Carvalho et al. [25] propose a lower layer to multicast gossip protocols, called Payload Scheduler, to approximate gossip multicast to that of a multicast tree, despite not building an explicit tree. The proposed Payload Scheduler behaves like a gossip target biasing solution. It assigns weights to neighbors according to a number of different strategies but, instead of using the weights to select an appropriate gossip partner, they use them to decide whether to send the message using an eager or a lazy push approach.

This way, they try to minimize the redundancy of messages that normal epidemic protocols produce, by approximating the gossip dissemination to a multicast tree. This approach creates a tradeoff between bandwidth and latency: the fewer eager push transmissions that are made (and thus less bandwidth wasted on message payloads), the more likely it is that each node receives the message id before the message payload (resulting in increased latency due to the additional request for the payload).

Some of the proposed strategies are of particular interest to us. The *Time-To-Live* (TTL) strategy uses an eager push approach only for the first rounds of the dissemination of each message. This follows the intuition that as the number of rounds increases, so does the number of nodes that have already received the message, therefore decreasing the probability of the gossip target peer needing the message payload. The *Radius* strategy tries to optimize the latency of messages using an eager approach only on nearby neighbors, according to an arbitrary measure of distance. The *Ranked* strategy always sends the full message payload to and from nodes that have higher capacity. This approach reduces the latency penalty in gossips with those nodes, since they are the least likely to be affected by the bandwidth overhead.

By combining the three above strategies, such as the authors propose in the *Hybrid* strategy, it is possible to decrease the radius threshold according to the number of rounds elapsed since the origin of the message. This can be useful to emphasize long links in the first hops of a message, distributing it evenly throughout the network, and then use only low latency links to disseminate it quickly around the holders of the message, simulating a message with many sources. The inclusion of the Ranked strategy prevents an unoptimized usage of high capacity nodes, applying the above safeguards only for nodes with potential bandwidth issues.

3.5.5 Plumtree

Plumtree [14], like GoCast, creates a multicast tree to disseminate messages in an eager push approach. It maintains two distinct sets of peers as well, *eagerPushPeers* and *lazyPushPeers*. Contrary to GoCast, however, Plumtree does not restrict the membership of each node to a majority of low latency neighbors. Instead, it biases a random peer sampling, selecting the closer neighbors to form a tree.

A neighbor is deemed close (moved to the eagerPushPeers set and thus contributing a link to the tree) when a previously unknown message is received from that neighbor. In turn, neighbors that deliver already known messages are grouped into the *lazyPushPeers* set. Communication to this set is made by lazy push gossip.

Because the tree branches are created following the paths of the first broadcast message, it is inherently optimized for the sender of that specific message. Messages sent by different sources experience a higher latency unless multiple trees are built and maintained.

To overcome the latency penalty for multiple broadcast sources that results from the tree creation process, Plumtree includes an optimization that continuously updates the tree by promoting lazy links to eager links, when the former deliver messages with less hopcount than the latter.

It is possible to define Plumtree as a Payload Scheduler strategy (similar to the Radius strategy described above), where the probability of employing an eager push approach is 1 for neighbors with low latency links and 0 for the others. The difference is that Payload Scheduler modifies the behavior of a cyclic strategy, reducing the load on high latency links, whereas Plumtree implements a pure reactive strategy, only disseminating information the moment it is received, without waiting for gossip rounds.

3.5.6 X-BOT

X-BOT [22] is a partial membership service that employs a membership restriction strategy to optimize an arbitrary cost function (for instance, latency) of neighboring links. Each node maintains a small symmetrical list of neighbors, called the *Active View*, containing both optimized gossip targets and some other non-optimized (or *unbiased*) targets to maintain connectivity and thus guarantee reliability. TCP connections to these neighbors are used as a simple failure detector, to minimize the impact of having a reduced neighbor list.

Each node has also a larger secondary view, named *Passive View*, that contains only random nodes. This view is periodically refreshed to reflect the arrival and departure of nodes in the network. Peers in this list are not considered when selecting gossip targets. Instead, the *Passive View* provides potential new additions to the *Active View* if more suitable neighbors are found. This process ensures a dynamic biased neighbor list, that converges to the best possible list instead of stopping at local minima.

Due to the symmetrical list of neighbors, the optimization process cannot be executed by each node locally. The authors propose then a four node optimization process, which, despite introducing some overhead in the protocol's optimization steps, still maintains locality to the nodes and their neighbors, not affecting the rest of the network.

System	**Approach**	**Addressed Properties**								
		Link Properties				Node Properties		Overlay Properties		
		Latency	Loss Rate	Link Cap.	Link Independence	Node Cap.	Expected Uptime	Het. Degrees	Het. Connectivity	Hierarchical Overlay
Directional Gossip	Target Bias	-	✓	-	✓	-	✓	-	✓	✓
HiScamp	Membership Bias	-	-	✓	✓	-	-	-	-	✓
GoCast	Membership Bias	✓	-	-	-	-	-	✓	✓	-
Payload Scheduler	Target Bias	-	-	✓	-	✓	-	-	-	-
Plumtree	Target Bias	✓	-	-	-	-	-	-	-	-
X-BOT	Membership Bias	✓	-	-	-	-	-	✓	-	-

Table 1. Summary of the system's analysis

3.6 Comparison of the Analyzed Systems

Now that we have presented each system individually, we will compare their characteristics, identifying the tradeoffs made by their approaches when optimizing the protocol operation. For reference, we present a brief summary of the analysis in Table 1.

Generally, systems that take latency into account do so by selecting in each node an optimized set of peers to which they send messages immediately at receiving time. Because of this initial flooding and the overhead of maintaining optimized neighbor sets, the average load on links is greater than in systems with a cyclic gossip strategy. A common strategy to ensure fast delivery of messages is to create a multicast tree with the lowest latency links, combining the reliable gossip approach with previous tree multicast solutions.

Additionally, latency optimized systems also show an inherent tradeoff between the latency of broadcast messages and catering for link independence. The intuition behind this tradeoff is that in order to disseminate information quickly, the first nodes should send the information to other clusters. However, if the intercluster communication can be made at will by any node (*i.e.*, the responsibility of sending the information to other clusters is not divided explicitly between same cluster nodes), links that connect different clusters have to handle additional load due to information redundancy.

The inverse is also true: systems that instead define link independence and link capacity as the key properties to be taken into account introduce a latency penalty when compared to other solutions that do not.

4 Datacenter Network Topologies

We will now study datacenter network topologies as to better understand which techniques would have greater potential to improve epidemic protocols in such networks. We will start by analyzing the datacenter topologies that are currently being employed in nowadays datacenters. However, restricting our solution to those particular topologies would be insufficient, as alternative infrastructures are being proposed to replace them. Therefore, we also need to take the proposed topologies into consideration in order to design a solution that offers optimized gossip operation in both.

After analyzing these different datacenter topologies, we will identify their most relevant properties and propose some guidelines for the design of a gossip protocol that favors datacenter usage.

4.1 Current Topologies

Current datacenters employ a three-tiered network architecture in the form of a tree [27]. At the root of the tree, there is a core tier responsible for routing information to and from the second level, the aggregation tier. The aggregation tier is in turn responsible for routing information to and from the edge tier, that connects to the servers, the leaves of the tree. An example is shown in Figure 3.

The core switches[1] need to handle potentially more data than the aggregation and edge switches. Thus, they need to be of highly specialized and expensive hardware.

One important aspect to notice is the relative properties of communication between different servers. For instance, sending data to a server handled by the same edge switch has less latency and loss rate than sending the same data to a server handled by a different edge switch but the same aggregation switch. Similarly, it has even less latency and loss rate than sending the data to a server handled by a different aggregation switch. This fact implies a ranking of the neighbors in terms of total cost of communication that should be explored in the final solution.

[1] We will use the word switches to refer to routing equipment such as routers and switches with no differentiation, as to comply with the suggested bibliography.

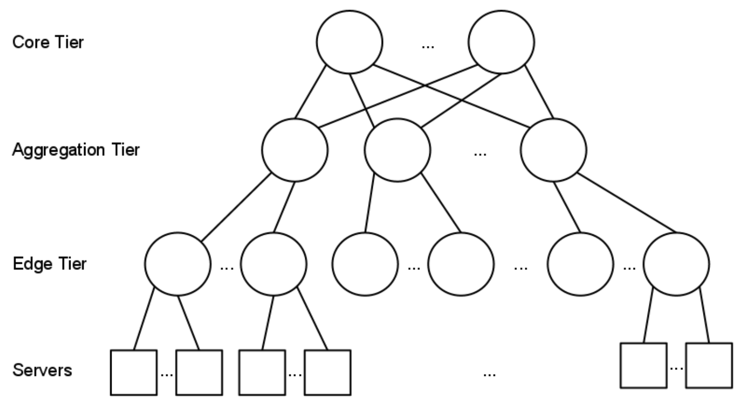

Fig. 3. A three-tiered datacenter architecture

Another relevant fact to consider is that different patterns of communication have a different impact in the regions of the network. If there is a lot of communication between same-edge servers, the aggregation and core switches will have less load, but if there is a lot of communication between servers handled by different aggregation switches, the edge switches still have to deliver the messages, although their total load is divided more evenly between them in this case.

In the worst case scenario, where all the nodes want to communicate with a single node, the switches closer to that node will have to handle all the traffic in the network, generated by all the nodes at once. With high probability, a lot of messages will block while waiting to be delivered. Although this is an unlikely scenario, it is not difficult to imagine close variants with gossip protocols that do not employ a uniform peer sampling service.

4.2 Proposed Topologies

Recent proposals for datacenter topologies try to address the different cost of communication for different peers and the monetary cost of the hardware used in these systems. We will now show some examples of state-of-the-art proposals, identifying the network properties that change from the currently used topologies to these proposed ones.

4.2.1 Triton

Vahdat et al. [28] propose a novel architecture for datacenters based on modular groups of commodity switches, called *pods*. Each pod consists of multiple aggregation and edge switches, and is connected to the core switches through multiple aggregation switches. This ensures a better load balancing scheme due to the multiple paths connecting any two nodes.

They guarantee 100 percent throughput with this model, while reducing the hardware cost by employing *merchant silicon*, creating a single piece of hardware comprised of multiple chips of commodity switches. The cost is reduced not only due to the commodity switches but the absence of cables connecting the switches that share a pod. The core switches are also bound together in a core switch array.

To take advantage of the multiple paths between servers, they use both *Equal-Cost Multi-Path* (ECMP) forwarding and *Hedera*. ECMP is a standard multipath forwarding mechanism in which some of the packet header's fields are hashed. The resulting hash is used as an input to a function that decides the next switch in the path (when there is a choice), ensuring uniform load distribution while maintaining the same path for a unique data flow.

Hedera is their contribution to solve the problem of certain communication patterns that exhibit suboptimal performance on ECMP. When multiple large flows are assigned the same path, the corresponding link has to support all the load, whether the alternative links are stressed or not. For that reason, Hedera monitors the network for large flows and when it finds cases like the one mentioned above, reassigns some of the flows to different paths.

Since Hedera only addresses long communication flows and ECMP is oblivious to communication patterns, blocking may still occur when the same route is used for a number of nodes that wish to gossip with another group of peers that share links in the path. We believe more could be done to accommodate this scenario and we plan on testing it to understand how likely it is to happen and what we can do to avoid it.

4.2.2 CamCube

Given that many services running on datacenters are essentially key-value storage systems, Costa et al. [29] propose a physical Distributed Hash Table architecture for the datacenter hardware itself. Each server can be directly connected to a group of other servers, mimicking the links in the structured overlay (in their case a 3D torus).

To optimize the information flow, they realize that different services want to achieve different properties, which may not be guaranteed by the default routing method [30]. They state that a transparent, customizable routing method can increase the performance of the overall network when multiple services are used. To demonstrate this, they implement specific routing for a number of services (TCP/IP Support, Virtual Machine Distribution, Cache Service and Aggregation Service for MapReduce systems) and show that each service's overhead decreases, while not straining each link with too many services.

Although this topology maximizes the synergy between the physical and logical networks, it is designed with static structured networks in mind. Still, the authors recognize that different services have different routing requirements.

We have seen that broadcast services, for instance, can benefit from an epidemic approach to information dissemination. Therefore, it could be beneficial to those services if we included in our solution the ability to take advantage of these types of synergy between physical and logical networks when selecting gossip targets.

5 Architecture

As stated before, we want to optimize gossip for use in datacenters. To that end, we need to consider a network model that captures the properties of both current datacenters and possible future ones. In this section, we will formalize the characteristics of our proposed model and present the guidelines we believe are critical for the design of our optimized gossip protocol.

5.1 Modeling Datacenter Networks

As we have shown, an important characteristic of most datacenter topologies is that they are often shaped like a tree of connections where the leaves are the nodes, creating a noticeable difference in communication time between distinct pairs of nodes. In our abstraction, we should model this latency penalty so it can contribute to the selection of neighbors.

Because we are taking advantage of the underlying network when building our overlay (increasing the synergy between the two), the division of nodes in regions (where applicable) should also be taken into account, in terms of both link independence (to reduce load on links that connect different regions) and heterogeneous connectivity (to ensure connectivity of the regions).

However, we have seen in the analyzed systems that latency optimization is typically tackled by a reactive push gossip strategy. This increases the average load of the system, creating an overhead that could stress links that connect regions. Since this conflicts with our objective of optimizing the load on these links, it is crucial to adopt more sophisticated strategies to reduce the latency penalty. Alternatively, it is fundamental to define tunable parameters that can be used to give more importance to each side of the tradeoff, letting administrators or even service designers specify which properties are more important for their applications.

The relevant properties to be considered in our solution are then:

- Latency (as the number of hops between two nodes)
- Link independence
- Heterogeneous Connectivity

Since heterogeneous connectivity is a property of the overlay, this means that our protocol will need to have access to two properties of the underlay: hop count between the nodes and the overlay-to-underlay link mapping. To include the hop count in the network model, we propose an undirected, fully connected, weighted graph. The weights are the number of hops between the nodes.

Because link independence is difficult to represent in a model, we will assume that our protocol can have access to either a diagram of the underlay network or a way of inferring the positions of nodes in the physical topology, given a numeric division of clusters. To identify the cluster of each node, its node id is prefixed with its cluster number.

An example is shown in Figure 4. For clarity, only links pertaining to node 1 are fully represented.

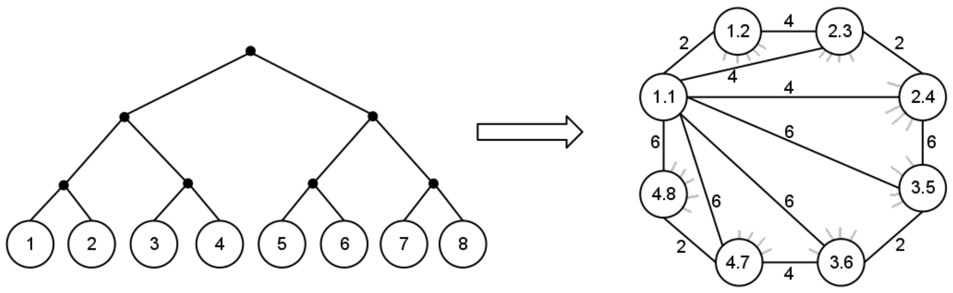

Fig. 4. Overlay model of a datacenter network

It is important to note that this model does not assume a full membership view in the gossip protocol. To adapt the model to a specific scenario that uses partial views and asymmetrical neighbor lists, we need only to omit the nonexistent or unknown edges from the model and replace the undirected edges with directed edges for the asymmetrical neighbor relations.

5.2 Designing a Datacenter Gossip Protocol

We will now translate the model into potential guidelines for our proposed solution.

We have seen in Directional Gossip and HiScamp that a potential strategy to deal with a hierarchy of communication costs is to adopt that same hierarchy in the system structure. Current and future datacenter topologies differ in the number of levels in the hierarchy. For instance, current datacenters have three levels of switches that affect the communication cost: core, aggregation and edge. The Triton architecture reduces that number to two, by merging aggregation and edge switches into pods. With this in mind, our protocol should create a gossip layer hierarchy of variable depth, similarly to HiScamp, featuring a different number of levels for each topology (*e.g.* three levels of gossip for current datacenters, two levels for the proposed Triton architecture, and a single level for the CamCube architecture).

However, HiScamp maintains only a single link between clusters, which reduces the reliability of the system, a key feature in epidemic protocols. Furthermore, each HiScamp level introduces a penalty in latency, which we are trying to optimize. For those reasons, we plan on employing a neighbor bias strategy instead of HiScamp's membership restriction strategy. A possible approach to that type of solution would be integrating some of the ideas present in Payload Scheduler when choosing weights for each neighbor. If we applied the Radius and Time-To-Live strategy, nodes could send the intended information to inter-cluster links with inverse probability to the number of hops that the message took until the present time. This would ensure fast dissemination to all the clusters in the first hops and parallelized intra-cluster information sharing in the following hops.

Inside the lower level clusters (or the whole system in solutions like CamCube), latency should be similar between any pair of nodes. However, there are still possible optimizations to the neighbor selection. In fact, since in most solutions the clusters are evenly divided and contain only a fraction of the nodes of the whole system, it may be feasible to manage a full cluster membership view such as in HiScamp and organize the nodes with the objective of minimizing intra-cluster information redundancy. In already structured solutions like CamCube, the organization is aided by the coordinates system.

6 Evaluation

We will conduct an experimental evaluation of the protocol, through extensive simulations using the Peersim Simulator [31].

6.1 Metrics

As explained previously, we want to improve the average load on the links, especially the ones that connect different regions of the network. However, the impact on the latency of the messages must be minimized. Finally, we need to ensure that the robust properties of gossip are still maintained. For those reasons, we will evaluate the protocol with the following metrics, as defined in 3.2:

- **Link Stress** (measured by the number of messages that go through each link in each gossip round) and **Relative Message Redundancy** to assess the load on the links and if it can be reduced
- **Last Delivery Hop** to estimate latency in an easily comparable fashion. For state reconciliation systems, a similar metric will be used to count the number of rounds that the protocol takes to converge.
- **Reliability** to ensure that the gossip robustness still holds

6.2 Methodology

We will evaluate the above metrics using implementations of the other available solutions as a baseline for comparison. The evaluation steps will be:

- Build two prototype PeerSim applications that use the designed protocol as a building block for reliable broadcast and state reconciliation

- Evaluate the benefits of the optimized protocol with the above metrics in a simulation of a current datacenter topology in stable conditions
- Evaluate the benefits of the optimized protocol with the above metrics in a simulation of a current datacenter topology in the presence of node failures (with various percentages of failures)
- Repeat the simulations with the proposed topologies Triton and CamCube

7 Scheduling of Future Work

Future work is scheduled as follows:

- January 9 - March 29, 2012: Detailed design and implementation of the proposed architecture, including preliminary tests.
- March 30 - May 3: Perform the complete experimental evaluation of the results.
- May 4 - May 23: Write a paper describing the project.
- May 24 - June 15: Finish the writing of the dissertation.
- June 15: Deliver the MSc dissertation.

8 Conclusions

In this report, we have presented a survey of the different techniques that have been proposed to exploit topology of the underlying network when optimizing the operation of gossip protocols. We divided the solutions in two major categories, *biasing the gossip target* and *biasing the partial views*, and explained both.

We have also studied the topologies of datacenter networks, proposing a possible abstraction that can address the most relevant properties of these topologies.

Finally, following this model, we have identified some directions of research to improve the state-of-art in this respect, including guidelines for the design of a novel gossip protocol adapted to datacenter networks.

Acknowledgments

We are grateful to João Leitão, João Paiva and Sérgio Almeida for the fruitful discussions and comments during the preparation of this report. This work was partially supported by Fundação para a Ciência e Tecnologia (FCT) via the INESC-ID multi-annual funding through the PIDDAC Program fund grant and via the project "HPCI" (PTDC/EIA-EIA/102212/2008).

References

1. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., Terry, D.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing. PODC '87, New York, NY, USA, ACM (1987) 1–12

2. Agrawal, D., El Abbadi, A., Steinke, R.C.: Epidemic algorithms in replicated databases (extended abstract). In: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems. PODS '97, New York, NY, USA, ACM (1997) 161–172
3. Birman, K.P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y.: Bimodal multicast. ACM Trans. Comput. Syst. **17** (May 1999) 41–88
4. Leitão, J., Pereira, J., Rodrigues, L.: HyParView: A membership protocol for reliable gossip-based broadcast. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DSN '07, Washington, DC, USA, IEEE Computer Society (2007) 419–429
5. Gupta, I., Renesse, R.v., Birman, K.P.: Scalable fault-tolerant aggregation in large process groups. In: Proceedings of the 2001 International Conference on Dependable Systems and Networks (formerly: FTCS). DSN '01, Washington, DC, USA, IEEE Computer Society (2001) 433–442
6. Renesse, R.v., Birman, K.P., Vogels, W.: Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. ACM Trans. Comput. Syst. **21** (May 2003) 164–206
7. Renesse, R.v., Minsky, Y., Hayden, M.: A gossip-style failure detection service. Technical report, Ithaca, NY, USA (1998)
8. Eugster, P.T., Guerraoui, R.: Probabilistic multicast. In: Proceedings of the 2002 International Conference on Dependable Systems and Networks. DSN '02, Washington, DC, USA, IEEE Computer Society (2002) 313–324
9. Karp, R., Schindelhauer, C., Shenker, S., Vocking, B.: Randomized rumor spreading. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, Washington, DC, USA, IEEE Computer Society (2000) 565–
10. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., Vogels, W.: Dynamo: Amazon's highly available key-value store. SIGOPS Oper. Syst. Rev. **41** (October 2007) 205–220
11. Lakshman, A., Malik, P.: Cassandra: A decentralized structured storage system. SIGOPS Oper. Syst. Rev. **44** (April 2010) 35–40
12. Ganesh, A.J., Kermarrec, A.M., Massoulié, L.: HiScamp: self-organizing hierarchical membership protocol. In: Proceedings of the 10th workshop on ACM SIGOPS European workshop. EW 10, New York, NY, USA, ACM (2002) 133–139
13. Tang, C., Ward, C.: GoCast: Gossip-enhanced overlay multicast for fast and dependable group communication. In: Proceedings of the 2005 International Conference on Dependable Systems and Networks. DSN '05, Washington, DC, USA, IEEE Computer Society (2005) 140–149
14. Leitão, J., Pereira, J., Rodrigues, L.: Epidemic broadcast trees. In: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems. SRDS '07, Washington, DC, USA, IEEE Computer Society (2007) 301–310
15. Ganesh, A.J., Kermarrec, A.M., Massoulié, L.: SCAMP: Peer-to-peer lightweight membership service for large-scale group communication, Springer-Verlag (2001) 44–55
16. Voulgaris, S., Gavidia, D., Steen, M.v.: CYCLON: Inexpensive membership management for unstructured p2p overlays. Journal of Network and Systems Management **13** (2005) 2005
17. Eugster, P.T., Guerraoui, R., Handurukande, S.B., Kouznetsov, P., Kermarrec, A.M.: Lightweight probabilistic broadcast. ACM Trans. Comput. Syst. **21** (November 2003) 341–374

18. Rhea, S., Geels, D., Roscoe, T., Kubiatowicz, J.: Handling churn in a DHT. In: Proceedings of the annual conference on USENIX Annual Technical Conference. ATEC '04, Berkeley, CA, USA, USENIX Association (2004) 10–10
19. Jelasity, M., Guerraoui, R., Kermarrec, A.M., Steen, M.v.: The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware. Middleware '04, New York, NY, USA, Springer-Verlag New York, Inc. (2004) 79–98
20. Renesse, R.v., Dumitriu, D., Gough, V., Thomas, C.: Efficient reconciliation and flow control for anti-entropy protocols. In: Proceedings of the 2nd Workshop on Large-Scale Distributed Systems and Middleware. LADIS '08, New York, NY, USA, ACM (2008) 6:1–6:7
21. Eugster, P.T., Guerraoui, R., Kermarrec, A.M., Massoulié, L.: Epidemic information dissemination in distributed systems. Computer **37** (May 2004) 60–67
22. Leitão, J., Marques, J., Pereira, J., Rodrigues, L.: X-BOT: A protocol for resilient optimization of unstructured overlays. In: Proceedings of the 2009 28th IEEE International Symposium on Reliable Distributed Systems, Washington, DC, USA, IEEE Computer Society (2009) 236–245
23. Lin, M.J., Marzullo, K.: Directional gossip: Gossip in a wide area network. In: Proceedings of the Third European Dependable Computing Conference on Dependable Computing. EDCC-3, London, UK, Springer-Verlag (1999) 364–379
24. Vigfusson, Y., Birman, K., Huang, Q., Nataraj, D.P.: Optimizing information flow in the gossip objects platform. SIGOPS Oper. Syst. Rev. **44** (April 2010) 71–76
25. Carvalho, N., Pereira, J., Oliveira, R., Rodrigues, L.: Emergent structure in unstructured epidemic multicast. In: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. DSN '07, Washington, DC, USA, IEEE Computer Society (2007) 481–490
26. Kermarrec, A.M., Steen, M.v.: Gossiping in distributed systems. SIGOPS Oper. Syst. Rev. **41** (October 2007) 2–7
27. Al-Fares, M., Loukissas, A., Vahdat, A.: A scalable, commodity data center network architecture. SIGCOMM Comput. Commun. Rev. **38** (August 2008) 63–74
28. Vahdat, A., Al-Fares, M., Farrington, N., Mysore, R.N., Porter, G., Radhakrishnan, S.: Scale-out networking in the data center. IEEE Micro **30** (July 2010) 29–41
29. Costa, P., Donnelly, A., O'Shea, G., Rowstron, A.: CamCube: A key-based data center. Technical Report MSR TR-2010-74 (2010)
30. Abu-Libdeh, H., Costa, P., Rowstron, A., O'Shea, G., Donnelly, A.: Symbiotic routing in future data centers. SIGCOMM Comput. Commun. Rev. **41** (August 2010) 51–62
31. Montresor, A., Jelasity, M.: PeerSim: A scalable P2P simulator. In: Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09), Seattle, WA (September 2009) 99–100