Topology-aware Gossip Dissemination for Large-scale Datacenters

(extended abstract of the MSc dissertation)

Miguel Branco
Departamento de Engenharia Informática
Instituto Superior Técnico

Advisor: Professor Luís Rodrigues

***Abstract*—Gossip-based protocols are very robust and are able to distribute the load uniformly among all nodes. Furthermore, gossip-protocols circumvent the oscillatory phenomena that are known to occur with other forms of reliable multicast. As a result, they are excellent candidates to support the dissemination of information in large-scale datacenters. However, in this context, topology oblivious approaches may easily saturate the switches in the highest level of the datacenter network fabric. This document presents and evaluates a novel gossip protocol for datacenters, named Bounded Gossip, that provides an adequate load distribution among the different layers of the switching fabric of the datacenter, avoiding being a source of network bottlenecks.**

Bounded Gossip embodies techniques from previous protocols, such as Hierarchical Gossip and CLON, and combines them with a topology-aware membership maintenance scheme and a topology aware rate-based flow control scheme. The benefits from our solution are illustrated by an experimental evaluation that compares the performance of Bounded Gossip with that of other competing protocols, in terms of imposed load in the routing topology, overall cost of communication, and dissemination latency.

I. Introduction

Gossip-based dissemination protocols have proved to be very effective in supporting reliable dissemination of information in systems with large numbers of participants. Two of the main reasons for their success is their robustness and their ability to distribute the load uniformly among all nodes. This type of protocols has been used for different purposes, including reliable broadcast [1], [2], data aggregation [3], membership maintenance [4], among others [5].

Another important aspect of gossip protocols is that they can avoid the oscillatory phenomena that are known to occur with other forms of reliable multicast (*.e.g*, group communication based on view synchrony) [1]. Also, these protocols are based on point-to-point interactions, not requiring switches to support IP multicast. All these properties make gossip protocols particularly well suited to operate in large-scale datacenters [6].

Unfortunately, in this context, topology-oblivious approaches may easily saturate switches at the highest level of the datacenter network fabric. Therefore, gossip protocols for the datacenter must rely on some form of topology-aware approach. A number of topology-aware gossip protocols have been proposed in the literature, including HiScamp [7] , Hierarchical Gossip [8], and CLON [9]. As we will show, these protocols still exhibit some limitations, namely: i) despite trying to minimize the load imposed on core routers, the resulting load can still be unbounded, ii) resource usage is not very efficient, or, iii) their latency can be significantly larger when compared with flat gossip schemes.

In this document we propose a novel gossip protocol designed to operate on datacenters. Our solution, named Bounded Gossip, is based on a topology-aware approach.

In contrast with previous work, Bounded Gossip leverages a low-cost topology-aware membership maintenance scheme which imbues the overlay network established among nodes with deterministic topology-aware properties. Bounded Gossip leverages this membership service to provide a dissemination scheme that is partially deterministic, topology-aware, and robust. Additionally, we propose a topology-aware rate-based flow control scheme. The dissemination scheme also takes into consideration the freshness of messages during the dissemination process. The combination of these features allow Bounded Gossip to provide an adequate load distribution among the different layers of the switching fabric of the datacenter and to achieve a better resource utilization. The benefits of our solution are illustrated by an experimental evaluation that compares the performance of Bounded Gossip with that of relevant competing protocols.

The rest of this document is organized as follows: Section II discusses gossip-based protocols, and presents several works that have attempted to enrich gossip protocols with topology-awareness. Section III describes current datacenter network topologies. Section IV motivates, presents, and describes Bounded Gossip. In Section V we experimentally evaluate and compare our work to previous solutions found in the literature. Finally, Section VI concludes the document and provides points for future research directions.

II. Related Work

Several past works have proposed variants of gossip to make them topology-aware. In this section we discuss the most relevant examples and briefly compare their design with that of Bounded Gossip. One important aspect to point out, before we present our survey, is that most practical gossip protocols do not operate with full membership. A protocol that has access to full membership selects gossip targets at random from the entire population. However, maintaining full membership is impractical in large-scale

settings. Therefore, gossip protocols are supported by a companion peer-sampling service, that provides to each node a *partial view* of the system. The closure of partial views defines an overlay over which gossip is executed.

Consequently, there are two different ways to make a gossip protocol topology-aware: one consists in acting at the peer-sampling level, by selecting neighbors using some algorithm that takes the underlying network topology into consideration; another is to act at the gossip protocol level (for instance, by biasing the probabilities of selecting some particular members from the partial view). Naturally, some protocols, including our own, act at both levels.

A. Topology-aware Peer-Sampling

Recently, X-BOT [10] was proposed, an adaptive scheme which enables the topology of an unstructured overlay network to continually adapt itself to optimize a performance criteria provided by a companion oracle. X-BOT can be used to maintain a topology-aware unstructured overlay over which a flat gossip protocol can be executed. Unfortunately, X-BOT design can only distinguish between close and distant nodes, being therefore unable to capture more complex topologies, namely the hierarchical topologies with several levels that are usually used to connect nodes in datacenters (we discuss these topologies in the following section), being therefore inadequate to support efficient and topology-aware gossip solutions for datacenters.

The HiScamp protocol [7] is a distributed membership protocol designed on top of the Scamp peer sampling service [11]. HiScamp organizes nodes in clusters according to the underlying network topology. The notion of clusters is employed by the protocol to limit the number of overlay links that connect nodes in different clusters, and consequently, to limit the load imposed on key routing components of the underlying network topology. Unfortunately, and contrary to our work, the resulting overlay topology presents only few, and completely random, links among nodes in distinct clusters. This has a significant negative impact over the latency and reliability of a gossip protocol executed on top of it. The authors of HiScamp argue that their solution can be extended to cope with hierarchical topologies with several layers, however the proposed solution increases the negative effects identified above.

None of the protocols above match well the topologies of datacenters and, to the best of our knowledge, no peer-sampling service has been previously designed with that goal in mind. Furthermore, as it will become clear with the remaining of the document, it is questionable that a satisfactory solution can be achieved acting exclusively at this level. Therefore, Bounded Gossip combines a novel peer-sampling strategy with other mechanisms.

B. Topology-aware Gossip

In contrast with the aforementioned solutions, that work at the peer-sampling level, an alternative approach consists in directly manipulating the patterns of communication exhibited by the gossip dissemination scheme.

A protocol that follows this approach is the Hierarchical Gossip protocol [8]. Hierarchical Gossip operates by relying on a non-uniform selection of nodes when gossiping, in such a way that each node has a higher probability of gossiping with nodes close to the initiator in the underlay. The solution can take into consideration several levels of distance, offering therefore a possibility for capturing the inherent complexity of typical datacenter's topologies. Unfortunately, the peer selection strategy employed by Hierarchical Gossip is still completely random in nature, and contrary to Bounded Gossip, it does not take into consideration the freshness of messages being gossiped when selecting nodes to forward them to. This reduces the efficiency of Hierarchical Gossip's resource usage. As our experimental results demonstrate, this induces excessive load on key routing components of the hierarchical topology. Consequently, the maximum throughput of Hierarchical Gossip is extremely below the one that can be achieved with Bounded Gossip.

C. Hybrid Approaches

Contrary to the works discussed above, our solution relies on an integrated approach, which combines a topology-aware partially deterministic membership service, with a topology-aware partially deterministic gossip dissemination scheme. CLON [9] is a recent work that follows a similar approach to our own, and that, similar to Bounded Gossip, also explicitly tackled the problem of supporting efficient and reliable topology-aware gossip protocols for, and between, datacenters.

In a similar fashion to HiScamp, CLON leverages the Scamp protocol to build a topology-aware overlay network connecting nodes in a datacenter. The design of this membership service promotes the maintenance of distant neighbors by each node in the system. On top of this topology-aware membership service, CLON executes a gossip dissemination scheme that manipulates the probability of selecting an overlay neighbor to receive a gossip message according to the freshness of that message.

However, and contrary to our solution which aims at supporting efficient and reliable gossip inside a single datacenter, CLON is mostly concerned with providing gossip support for services deployed across multiple datacenters. This clearly reflects on the membership service employed by CLON, which can only distinguish between close and distant neighbors, being therefore unable to capture hierarchical topologies with several tiers. This significantly increases the average dissemination latency of the solution. Also, in sharp contrast with our solution, CLON does not apply any form of deterministic biasing to the gossip targets selection and does not rely on a topology-aware flow rate control scheme. This leads CLON to consume high bandwidth when several nodes concurrently inject messages on the gossip protocol.

As the overview above indicates, to the best of our knowledge, Bounded Gossip is the first gossip dissemination solution that combines a partially deterministic topology-aware membership service, a dissemination scheme that not only is topology-aware but also partially deterministic, and

a topology-aware flow rate control mechanism to support an efficient, reliable, and topology-aware gossip solution that is specially tailored for operation on a datacenter network.

III. Network Architecture

In this section we describe the typical topology of a datacenter. Our protocol is designed to operate efficiently on topologies similar to the ones described here. More precisely, we aim at achieving the following relevant characteristics: i) minimize the load imposed on the switches of the network infrastructure; ii) use resources in an efficient fashion, as to maximize the throughput with a maximum acceptable communication load; and iii) minimize the overall latency of the gossip dissemination process. Datacenter network topologies usually follow a three-tiered architecture or a two-tiered architecture, as described below.

A. Three-tiered Architecture

The most common architecture for large datacenter networks is the three-tiered architecture [12]. In this architecture, the network is characterized by three levels of routing equipment[1]. The top tier, usually named the *core tier*, is composed of a single core switch that spawns multiple aggregation switches at the second tier. The aggregation switches connect multiple edge switches that form the edge tier. Those switches are typically top-of-rack switches and connect directly to nodes. In typical configurations, each edge switch connects from 20 to 80 nodes [12]. In the remainder of the document we refer to the set of all nodes connected directly to an edge switch as a *cluster*. To minimize the load imposed over the core switch, some configurations rely on multiple core switches. In this type of architecture each of these core switches owns an independent physical link to each aggregation switch, providing redundant paths between every pair of switches at the aggregation tier[2]. In smaller deployments, the three-tiered architecture is sometimes simplified to a two-tiered architecture. This is achieved by eliminating the aggregation tier and having the edge switches connect directly to the core.

B. Abstracting the Physical Topology

To make our protocol as generic as possible, we make a number of assumptions that allow us to abstract the physical topology, namely by considering that the naming scheme used to identify nodes encodes some information about the location of those nodes in the physical topology. Note that similar assumptions are employed by competing protocols [8].

We assume that all switches in the datacenter are numbered following a hierarchical naming scheme. Furthermore, we treat the identifier space of any set of switches connected to the same switch as a circular space. The child switches of another switch are numbered from 0 to $N-1$, where N is the number of child switches. An example for this numbering scheme is shown in Figure 1.

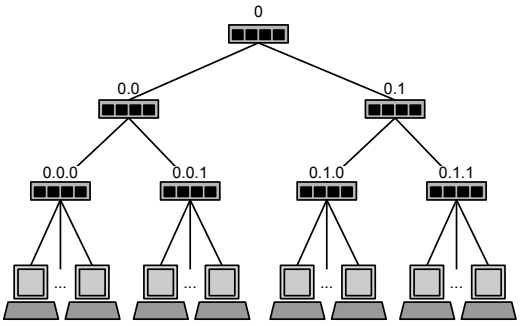


Figure 1. An example of the switch numbering scheme

We further assume that the IP address of a node enables any node to locally determine the identifier of the edge switch to which that node is connected. While the specific strategy to determine the node's location in the network from its IP address is outside the scope of this document, it is important to note that the translation process does not need to be direct, $i.e.$, an IP address of $1.2.3.4$ may not mean that the node is connected to switches 1, 2 and 3 of the datacenter hierarchy. This notion is essential to allow our solution to support network topologies with different numbers of hierarchical levels.

IV. Bounded Gossip

In this section, we describe Bounded Gossip. We start by providing an overview of the Bounded Gossip building blocks and then proceed to make a detailed description of the operation of each of those components.

A. Overview

Our solution follows, and builds upon, the same intuitions behind the design of Hierarchical Gossip [8] and CLON [9], which we refine and extend with new mechanisms.

Bounded Gossip is composed of three main components which complement each other to provide a reliable, efficient, and topology-aware gossip dissemination scheme, that imposes a bounded load on switching components of the datacenter infrastructure. A peer sampling service is responsible for providing to each node in the system a set of partial views. The partial views are managed in such a way that their contents take into consideration the underlying network topology through the use of a limited form of determinism on the contents of those views. On top of this peer sampling service we devised a specially tailored gossip-based dissemination scheme, which leverages the characteristics of the resulting overlay. Our dissemination scheme induces a controlled amount of determinism to the gossip-based message dissemination pattern. This enables to lower the overhead imposed on switches while preserving fault tolerance. This dissemination scheme also takes into

[1] We will use the word switches to refer to both routers and/or switches, as found in most literature [12].

[2] As our solution in no way affects the uniformity provided by consistent hashing over messages employed by Equal-Cost Multi-Path routing, the overall load of the switches is still reduced in those topologies.

consideration the freshness of messages when forwarding them. Finally, the mechanisms above are complemented by a rate-based topology-aware flow control mechanism.

Our scheme can easily be configured to accommodate an arbitrary number of hierarchy levels in the datacenter topology. However, for the sake of clarity of exposition, we have opted to describe the operation of Bounded Gossip considering a three-tier network hierarchy. In the following we describe the operation of the three main components of Bounded Gossip.

B. Peer sampling Service

We rely on a gossip-based membership service that operates in a similar fashion to Cyclon [13]. In our solution however, each node maintains a set of distinct partial views, each view encapsulates information concerning each of the hierarchical levels of the underlying topology. Furthermore, and contrary to previous solutions, our membership service strives to induce a relaxed form of determinism in the way nodes are selected to fill partial views, such that it enables the emergence of topology-aware redundant tree-like topologies connecting all clusters in the datacenter.

Similar to Cyclon, every node periodically exchanges samples of the contents of their partial views with a particular peer. When exchanging these samples, nodes also add their own identifier to the sample sent to its peer. As Cyclon, we also assume that node identifiers stored in partial views are enriched with an age counter, which is increased periodically by nodes to reflect the amount of time that has passed since the creation of that particular identifier.

When a node receives a sample of the system membership from a peer, it uses the enclosed information to update the contents of its local partial views, respecting the constraints that are imposed by our membership service, which we will describe in the next paragraphs. In this process, nodes give preference to identifiers with lower age counters, as this increases the probability of the node that produced that identifier to be still active.

In our system, each individual node maintains L independent partial views of the system, where L is the number of hierarchical levels of the underlying network topology. We named these partial views PV_i where i indicates the hierarchy level encoded in the partial view contents (note that our solution supports an arbitrary number of hierarchy levels). Considering the 3-tier network topology discussed earlier, a node n owns the following 3 partial views:

PV_0 represents the lowest hierarchy level of the topology. This view should contain all the identifiers of nodes in the cluster of n. To optimize the dissemination, we ensure that the identifier of n also appears in the PV_0 of n. Furthermore, the contents of these views are kept sorted considering node identifiers. The size of this view depends on the network topology of the datacenter, having a size equal to the number of nodes in each cluster.

PV_1 contains identifiers of nodes which are reachable by n only by crossing a single aggregation switch. Nodes try to maintain in this view identifiers from K_1 deterministically chosen clusters. The preferred clusters of a node n are selected considering the id of the edge switch to which n is connected. Considering that n is connected to an edge switch with id $c.a.e$, n will give preference to nodes connected to switches with an id between $c.a.(e*K_1+1)$ and $c.a.((e+1)*K_1)$ (notice that we assume that the identifier space of switches is circular at each hierarchy level).

PV_2 captures the highest hierarchy level of the underlying topology. This is achieved by storing identifiers of nodes which are accessible to n only by crossing the core switch. Similarly to PV_1, this partial view is built while trying to keep K_2 identifiers from different deterministically chosen aggregation switches. Considering that n is connected through an aggregation switch with an identifier $c.a$, n will give preference to nodes connected through aggregation switches with identifiers between $c.(a*K_2+1)$ and $c.((a+1)*K_2)$. The edge switch identifiers are not relevant when managing the contents of this partial view. An example of such a bias is represented in Figure 2. In this scenario, we can observe some good properties of our membership scheme which will be leveraged by our dissemination process. If a message is generated in the first core zone (the blue zone), it can be disseminated to the next two zones, depicted in green and red. While the green zone can ensure the rest of the datacenter is infected by the message, the red zone is able to send redundant copies to the previously infected zones, which not only help the system recover from failures but also speeds up the delivery of the message in those zones by having more copies circulating when there are no failures. The core neighbors maintained by the rest of the zones are not depicted in the figure for clarity, but offer similar properties.

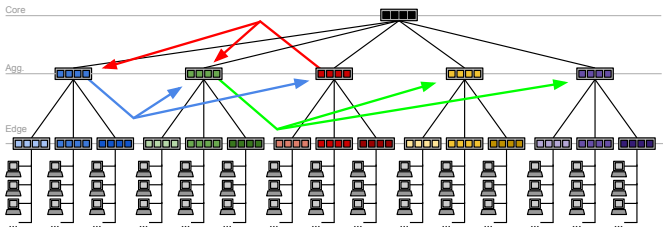


Figure 2. An example of membership bias at the core level

To improve reliability, we rely on the following strategies: i) When exchanging samples of their partial views, nodes include the complete contents of their PV_1 and PV_2 views. ii) They also remove from their partial views nodes that do not reply to previously request to exchange partial view samples. In this case, the membership service assumes those nodes to have failed and removes them from partial views. Additionally, as we discuss further ahead in the text, iii) in our dissemination scheme nodes have specialized *roles*, such as edge, aggregation or core *roles*. When nodes with a role r choose another node with whom they exchange shuffle messages in each round, they bias their selection to promote exchanges with nodes on their PV_r view with probability of

99%.

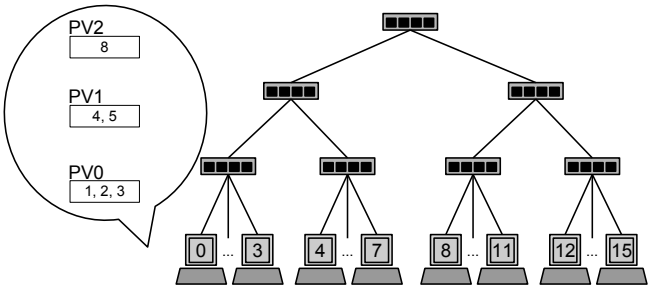


Figure 3. An example membership view of node 0

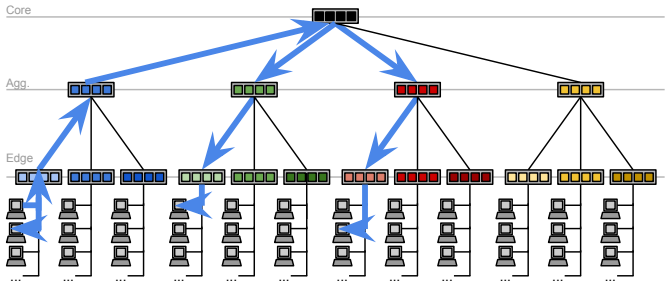
Figure 3 shows an example of how the partial views of a node are filled considering the underlying network topology.

C. Gossip-based Dissemination Scheme

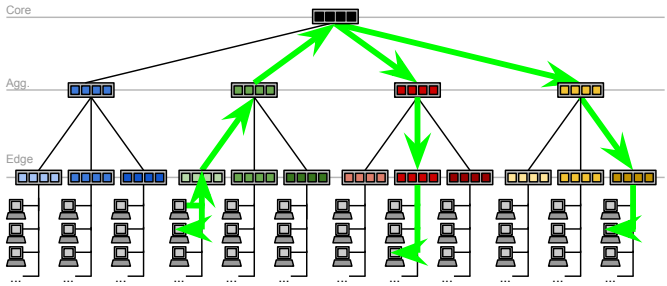
We rely on a gossip-based dissemination scheme that operates in a semi-deterministic fashion by leveraging the underlying topology-aware membership service. The algorithm is based on the principle suggested in a number of previous works, including CLON [9], that in order to minimize latency, messages should be disseminated primarily to remote nodes, and only then at more local levels. However, contrary to what happens in Hierarchical Gossip [8], nodes in our algorithm operate in the *infect and die* [14] model, where each node processes a message only once. When a node processes a message for the first (and single) time, it (re)transmits it to f neighbors (where f is the *fanout* parameter).

To that end, and considering the 3-tier architecture mentioned previously, when an application-level message m is generated in our system, the source node starts to transmit it through the core links of the topology, disseminating message m to all the different zones of the datacenter connected to the core switch. When a source s sends m, it also forwards at least one copy of the message to its local cluster, to ensure the continuation of the dissemination in its own zone. An example of such behavior is present in Figure 4(a). The nodes that received the message sent in the first step will then continue the dissemination, leveraging the properties of the biased membership service. They will not only send the message to uninfected zones in the datacenter but also forward redundant copies of the message to areas that have most likely been previously infected but that can still be oblivious to the message due to node or message failures. Similarly to the first step, each node that forwards a copy of m also sends at least one copy to its cluster. An example of this behavior is shown in Figure 4(b).

When all the zones in the datacenter separated by core links are infected by the message, our system will start saving core switch load by stopping transmission at that level. Instead, nodes will focus on transmitting the message at the aggregation level to infect all the clusters in their core zone. Again, to ensure continued dissemination inside



(a) Step 1



(b) Step 2

Figure 4. Example of dissemination at the core level

their cluster, the nodes will send at least one copy of the message to peers in their cluster. This process is illustrated in Figure 5.

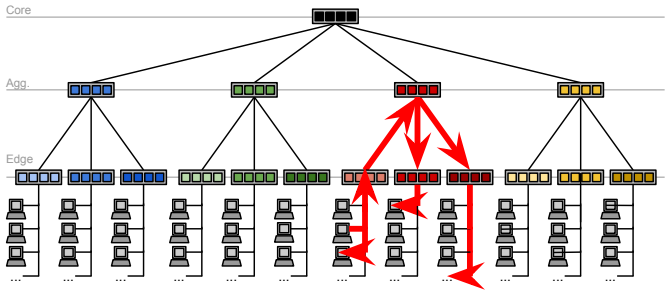


Figure 5. Example of dissemination at the aggregation level

Because nodes in the system are not able to know whether all the different zones in the datacenter are already infected by the message being transmitted, they use preconfigured round values to limit the number of rounds that a message can be disseminated using each of the hierarchy levels. In order to apply this strategy, that takes into account the freshness of messages, messages being disseminated by Bounded Gossip carry a T counter that indicates the number of times that the message has been retransmitted. To take the hierarchical topology into consideration, our dissemination process is controlled by a set of parameters $\pi_i, i \in [0, L[$, which limit the number of times a message can be retransmitted at each hierarchy level of the topology (notice that π_0 behaves as the typical *time to live* parameter of flat gossip

solutions [2]). To know the required hierarchical level for a message, a node finds the maximum i such that $m.T < \pi_i$. The intended level for disseminating the message is thus i. This also means that the message still requires dissemination at all the levels lower than i, which will be considered at a later stage. By transmitting fresh messages through remote links we "parallelize" the message dissemination, avoiding cases where a message is known by all nodes in a cluster before being transmitted to a second cluster, taking therefore double the time to infect nodes in both clusters. These scenarios occur in overlays with a small number of remote links, such as HiScamp and some aggressive configurations of CLON.

However, to ensure that there is a bounded communication cost at each routing element of the datacenter network infrastructure, even when multiple messsages are being generated and thus scheduled for remote transmission, we rely on the PV_0 provided by the membership service to attribute, in a deterministic fashion, specialized dissemination roles across nodes of a cluster.

A replication factor R is used to attribute roles to nodes in the same cluster as follows: the first R nodes in the (sorted) PV_0 of a cluster are chosen to disseminate messages using the highest hierarchical level of the datacenter network topology (they are thus called *core nodes*). The following R nodes in PV_0 are responsible for the dissemination at the next hierarchy level (*aggregation nodes*). Other nodes disseminate information only through the lowest hierarchical level (*edge nodes*). Please recall that PV_0 is a full view of the cluster and that it is sorted by node identifiers, so nodes can keep a consistent view of each member's role.

Algorithms 1 and 2 denote the pseudocode for the dissemination procedure of Bounded Gossip. When a node produces or receives a message for the first time, it stores the message in its local queue (lines $57-60$). Our dissemination scheme is modeled to operate in rounds. Therefore, periodically in each round, each node checks its queue for messages and processes them until its quota is reached (the quota values are defined by the flow control mechanism and will be detailed later). Recall that each processed message is sent to f other nodes.

For each message in a node's queue, there is a specific set of rules for its dissemination. Considering the message's round counter T, the node first discovers at which hierarchical level the message should be transmitted. Let h represent that level and r be the role of the node.

• If the message is already at the edge level (0), the node will forward the message to f nodes in its cluster, increasing the T counter (lines $9-13$).

• Otherwise, if $h < r$ but not 0, or the node has an edge role, the node redirects the message to the R nodes in its cluster responsible for retransmitting the message at level h (without increasing the T counter). Then, the node forwards the message to $f-R$ additional nodes in its cluster, starting by selecting nodes that are responsible for level $h-1$ and so forth, configuring T with appropriate values when forwarding the message for each level (lines $14-26$).

Algorithm 1: Bounded Gossip Dissemination (part 1)

Variable Description

***f*:** fanout parameter; ***knownMessages*:** list of ids of received messages
***queue*:** local message queue; ***quota*:** available message quota
***r*:** hierarchical level of the node's role
***time-to-live*:** messages' retransmission limit

```
1  upon event begin round do
2    quota ← resetQuota()
3    notified ← false
4    while queue ≠ ⊥ and quota > f do
5      msg ← queue.removeNextMessage()
6      h ← level(msg)
7      if msg.T < time-to-live do
8        quota ← quota − f
9        if h = 0 do
10         targets ← membership.getPeersInView(0, f)
11         newMsg ← msg.getCopyWithIncreasedLived()
12         for all peer ∈ targets do
13           trigger SEND(DATA, peer, newMsg)
14       else if h < r or r = 0 do
15         alreadySent ← 0
16         for role in [h to 0] do
17           targets ← membership.getPeersWithRole(role, f−alreadySent)
18           if role = h do
19             newMsg ← msg.getCopyWithSameLived()
20           else
21             newMsg ← msg.getCopyWithLivedFor(role)
22           for all peer ∈ targets do
23             trigger SEND(DATA, peer, newMsg)
24           alreadySent ← alreadySent + targets.size()
25           if alreadySent = f do
26             break for
```

If $h >= r$, the node will start by sending the message to all the nodes in its cluster with roles between h and r (if applicable). Then, the node will perform its role, forwarding the message to all the K_r neighbors in the corresponding view (increasing the T counter), as well as to the other $R-1$ nodes in the cluster also responsible for level r. Furthermore, the node uses the remaining fanout to forward the message to additional nodes in its cluster, starting by selecting nodes that are responsible for level $r-1$ and so forth, configuring T with appropriate values when forwarding the message for each level (lines $27-52$).

However, this step is executed only by one of the R nodes responsible for level r in the cluster, using a deterministic criteria based on the number of the gossip round. This allows Bounded Gossip to avoid redundant transmission of messages. A message that is processed by one of the replicas responsible for level r can be discarded by the remaining replicas of the same level, unless the copy to be discarded required dissemination at a higher hierarchical level than the already processed copy (lines $61-64$). This additional verification is meant to reduce cases where older copies of the message were processed first, which was problematic in scenarios where the first copy required dissemination at the core level and the second copy only required dissemination at the aggregation level. This erroneous behavior decreased the reliability of the protocol because the message would not traverse enough core links. To maintain synchronization between nodes with the same roles, they execute the dissemination strategy associated with their role in a round-

Algorithm 2: Bounded Gossip Dissemination (part 2)

```
27        else if myTurn() = true do
28          alreadySent ← 0
29          for role in [h to 0] do
30            if role = r do
31              targets ← membership.getPeersInView(r)
32              newMsg ← msg.getCopyWithIncreasedLived()
33              for all peer ∈ targets do
34                trigger SEND(DATA, peer, newMsg)
35              alreadySent ← alreadySent + targets.size()
36              targets ← membership.getPeersWithRole(r)
37              newMsg ← msg.getCopy()
38              for all peer ∈ targets do
39                trigger SEND(NOTIFICATION, peer, newMsg)
40              alreadySent ← alreadySent + targets.size()
41              notified ← true
42            else
43              targets ← membership.getPeersWithRole(role, f − alreadySent)
44              if role = h do
45                newMsg ← msg.getCopyWithSameLived()
46              else
47                newMsg ← msg.getCopyWithLivedFor(role)
48              for all peer∈ targets do
49                trigger SEND(DATA, peer, newMsg)
50              alreadySent ← alreadySent + targets.size()
51              if alreadySent = f do
52                break for
53    if notified ≠ true do
54      targets ← membership.getPeersWithRole(role)
55      for all peer ∈ targets do
56        trigger SEND(NOTIFICATION, peer, ⊥)

57  upon delivery DATA(sender, msg) do
58    if msg.id ∉ knownMessages do
59      knownMessages ← knownMessages ∪ msg.id
60      queue ← queue ∪ msg

61  upon delivery NOTIFICATION(sender, msg) do
62    if msg ≠ ⊥ do
63      if msg.id ∈ queue and level(msg) ≥ level(queue.get(id)) do
64        queue ← queue \ msg.id
```

robin fashion, ordered by increasing node identifier (recall that the identifier space of each role is considered circular). When a synchronization error occurs and more than one node transmits at once, the nodes reconfigure the order taking into account the smallest node identifier across those that have sent messages in the previous round. Because the current active node is important for the synchronization, nodes notify their peers with the same role even if they do not send actual messages of that role (lines $53 - 56$).

This procedure allows to effectively propagate a message throughout all nodes in the datacenter in an efficient manner, while still promoting a controlled amount of redundant messages to mask both message omissions and node failures. The amount of redundant traffic is controlled by the parameter R. Our experiments have shown that configuring R with a value of 2 yields high fault-tolerance in our scheme.

D. Flow Control Mechanism

In order to ensure a bounded dissemination traffic generated by Bounded Gossip, we rely on a simple, yet effective, distributed flow rate control mechanism. We remind the reader that each node maintains a queue which contains messages to be disseminated to its peers. To limit the number of messages transmitted per round, we use *quota* values for each node. Each round, nodes extract from its local queue a number m of messages such that $m \times f \leq quota$. Evidently this assumes that $quota \geq f$, $i.e$, the quota value is always greater than the fanout parameter of the protocol. The quota of each node depends on the node's role in the cluster, as there are different quota values for each hierarchical level, allowing for the limits of the dissemination traffic generated by Bounded Gossip to be finely-tuned across all levels of the topology.

Configuring the quota values to achieve a target load limit at each hierarchical level can be done through the expression $q_i = \frac{load_i}{Nclusters} \frac{f}{|PV_i|}$.

E. Fault Tolerance

Because our protocol maintains a notion of node roles, it is crucial to guarantee that each role is fulfilled by at least one node at all times. While this is a simple goal in scenarios where no failures can occur, maintaining the roles in the presence of failures can present a challenge.

Besides the mechanisms described in the previous sections designed to improve reliability, nodes in the same cluster maintain TCP connections between themselves, as a simple failure detection mechanism. When the connection to a node fails, the other members of the cluster remove it from their views and reconfigure their cluster roles, as to ensure R replicas per level, as explained in Section IV-C.

In scenarios with a large number of nodes in each cluster, maintaining TCP connections to every edge neighbor may prove impractical. In those cases, it is possible to save some connections in the nodes with edge role by keeping TCP connections only to nodes which role must be replaced in case of failure. We have also experimented alternatives to the TCP connections such as flooding the PV_0 or gossiping the nodes failures as an internal cluster message. However, the first alternative either greatly reduces the quota values of nodes when failures occur or increases the edge traffic in those cases. The second alternative increases the recovery time and thus offers weaker reliability properties.

V. Evaluation

To evaluate the performance of Bounded Gossip, we have simulated the protocol considering a datacenter network topology consisting of 1 core switch branching into 8 aggregation switches, each with 10 edge switches of clusters of 32 nodes. The total number of nodes in this network is $2,560$. All experiments were conducted using the PeerSim simulator [15], using its event driven engine.

To offer comparative baselines, we have also experimented with other solutions found in the literature over this topology. In particular we have tested: i) a *Flat Gossip* solution operating over a full membership; ii) a flat gossip solution operation on top of *Scamp* [11]; iii) the *Hierarchical Gossip* solution operating with full membership information [8]; and finally, iv) the CLON system [9]. All the protocol implementations were validated experimentally.

We configured every protocol to achieve 100% reliability. We set the f parameter of the Full Flat Gossip protocol to 13, and configured Scamp's redundancy factor C to match

this degree. Due to the number of parameters of the CLON protocol, we decided to conduct the experiments with 3 different configurations: CLON1 adds the nodes to the system randomly, achieving a total number of core connections close to 7.5% all connections; CLON2 uses a redundancy value C large enough to ensure a high degree for every node, so we can limit f and the core round limit; CLON3 uses an external method to add the nodes to the system, using as contact, a node in the closest hierarchical level possible, allowing for a smaller number of remote connections. For the Hierarchical Gossip solution, we manually selected a probability generating value K of 6 to artificially increase the probability of a node using the non-edge links and thus achieve the desired target reliability. We tested each protocol in a cyclic, *infect-and-die*, model, adding quota limits so we could limit the core load equally.

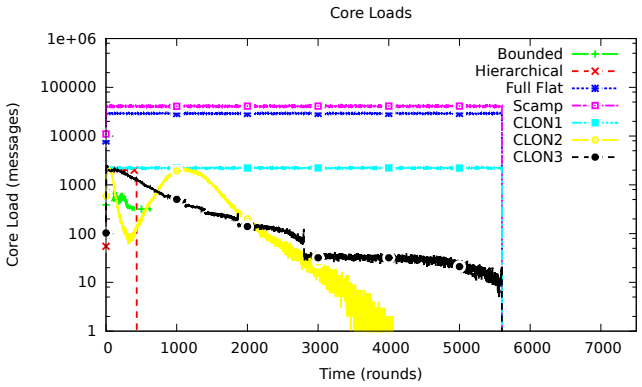


Figure 6. Core switch load for all protocols

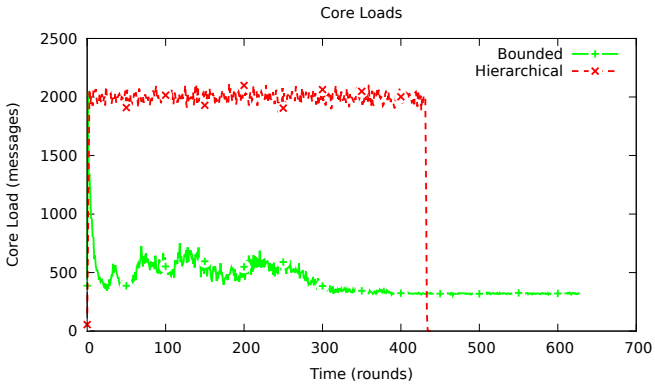


Figure 7. Core switch load (detail)

We can see that even the absolute minimum core switch load produced by Full Flat Gossip and Scamp greatly exceeds the configured quotas of the other protocols. This behavior is achieved by processing only one message at each gossip round, meaning that no less core switch can be produced while still executing the topology-oblivious protocols without some sort of flow control synchronization. The CLON1 configuration was not able to leverage the fanout and core rounds limit (otherwise it would lose reliability due to the reduced number of remote links used for each message) and therefore uses the maximum allowed load during the entire simulation.

Hierarchical Gossip maintained the maximum core switch load during all the simulation, wasting more total resources than Bounded Gossip, which only transmitted young messages through core links. It is also visible that the maximum core load in Bounded Gossip is only achieved in the scenario where 100% of messages are new and must be transmitted through core links. When both new and old messages are being transmitted, the core switch load is diluted through the rounds. A closer look at both protocols' behaviors is provided by Figure 7.

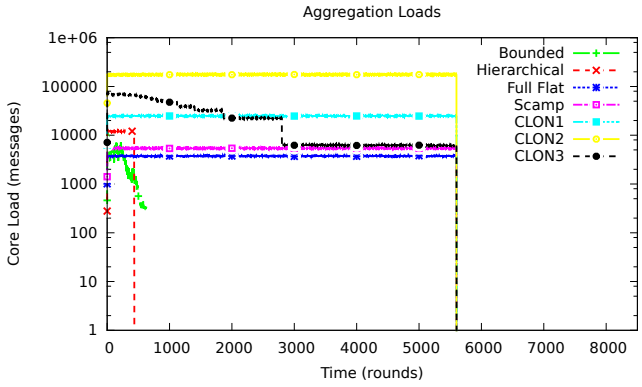


Figure 8. Aggregation switch load for all protocols

We also measured the load in the aggregation-level switches (without counting the load in these switches induced by the messages sent at the core level) in the virtual datacenter. Results are shown in Figure 8. In this case, the topology-oblivious solutions have a reduced load only because the number of connections to core nodes greatly exceeds that of connections to aggregation nodes. Still, Bounded Gossip is on par with these solutions and outperforms Hierarchical Gossip and the different CLON configurations.

It is important to note that Bounded Gossip is the only solution that sends membership messages ($i.e.$, messages that do not contribute to the dissemination of application-level messages but instead focus on maintaining the connectivity of the overlay network and the desired properties of the partial views) during the dissemination process. Scamp and CLON use a subscription mechanism that only sends messages when nodes join the system, which in our simulations is a process that happens exclusively before the generation of the messages. Our implementation of the remaining protocols operates over a full membership view. While this clarifies the contributions of each solution regarding the total number of messages sent (it reduces the likelihood of the results being confused with results of an implementation of any peer sampling service), such deployments of epidemic

protocols are unfeasible in production environments, where systems have to deal with nodes joining and departing the system and maintain an updated membership view.

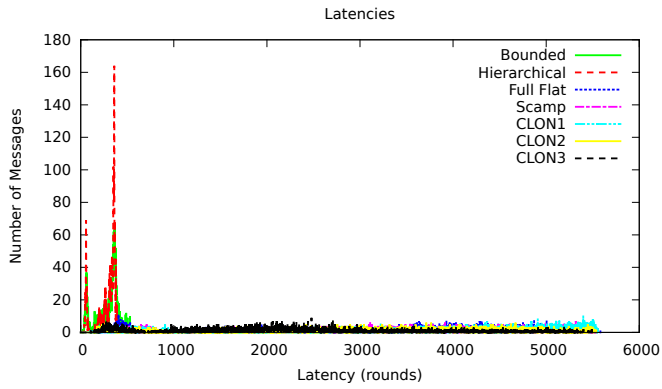


Figure 9. Latency measurements for all protocols

We also measured the latency distribution (in rounds) of all the application-level messages generated in the scenario above, to compare the overall latency of the dissemination process between the various solutions. The results can be seen in Figure 9.

Both topology-oblivious approaches show that due to the quota limitation with the objective of reducing the core load, the small number of messages processed in each round penalizes latency in an unfeasible way, achieving latency values for some messages of over $5,000$ gossip rounds. While the same is true for the two first configurations of CLON, the CLON3 setting is able to reduce the average latency of the messages, having fewer occasions where messages take more than $3,000$ gossip rounds to infect all nodes. The protocols with lower overall latency are Hierarchical Gossip and Bounded Gossip, delivering messages with an average latency of around 300 gossip rounds.

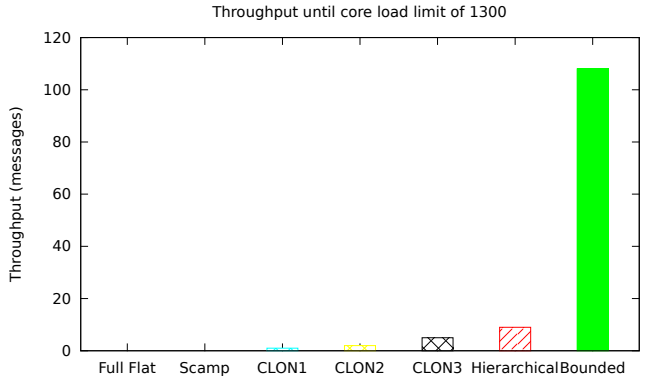


Figure 10. Throughput before core limit is reached

For the throughput experiment, instead of limiting the core switch load per round, we limited the total core switch load induced during the entire dissemination process, to simulate the expected throughput of the different solutions when there is a limit of the core switch load in a given time frame. For each protocol, we then observe the number of application-level messages they can deliver to every participant using only the determined number of core messages. Figure 10 illustrates the results.

It is visible that Bounded Gossip offers the best throughput when we limit the total core load to $1,300$ messages, by a factor greater than 10 over the second best protocol. As expected, considering the results presented above, the topology-oblivious solutions, Flat Gossip and Scamp, cannot effectively disseminate any messages to all participants with such a small core load. Although the first CLON configuration tries to send more core messages for a single application-level message than the ones allowed in total, the $1,300$ that were not dropped allowed that message to reach every node in the network. The second and third CLON configurations achieved slightly better results, due to the core round limits and the fewer core links, respectively. The poor resource utilization of Hierarchical Gossip also induces a high core load even when such messages are not needed, drastically reducing the throughput that the solution can achieve when the core switch load is limited in a given amount of time.

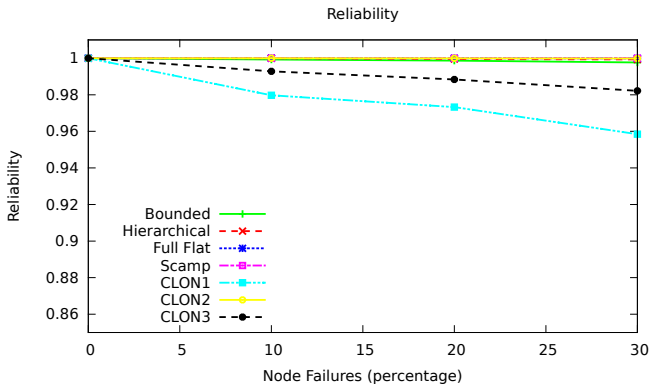


Figure 11. Reliability

In the reliability experiment, we evaluate Bounded Gossip's robustness in the presence of failures. We continuously generate messages up to a total of $5,600$ messages, failing a node uniformly at random in each gossip round. We executed various simulations with different percentages of nodes failing, up to 30% of all nodes in the system. Failed nodes did not join the system again. In the end of the simulation, we counted the number of active nodes that received each message and divided it by the total of active nodes still in the system. We then averaged that number to find the reliability of the protocol in that experiment. To exclude messages that had little chance of being disseminated, we only count messages that are known by at least one active node. Finally, we plotted the results and present them in Figure 11.

All the protocols were able to achieve similar results in

these conditions, although two of the three CLON configurations had weaker results (but still with reliability values of at least 96%). The other protocols are able to achieve reliabilities close to 100% even when 30% of all nodes fail. This demonstrates the inherent robustness of epidemic protocols, and one of the reasons their redundancy properties are important to keep in a system that requires reliability. Another important aspect to consider is that our solution is able to maintain those properties despite the membership and dissemination bias employed, that allowed for a better resource utilization and less core and aggregation switch load as seen in previous experiments.

VI. Conclusions

In this document we proposed Bounded Gossip, a gossip protocol for large-scale data centers. We showed the benefits of adding determinism to epidemic broadcast, creating a protocol that relies on three topology-aware components: a membership service, a dissemination scheme and a rate-based flow control mechanism. We evaluated the performance of our solution against previous works found in the literature and achieved better resource utilization that translates in 10 times message throughput with less switch load per round in the higher levels of the hierarchy and no penalty to overall dissemination latency or reliability. As future work, we plan on extending our solution to operate efficiently across multiple datacenters, leveraging the architecture described in this document that offers support for an arbitrary number of hierarchy levels. Additionally, it would be important to adapt and evaluate Bounded Gossip in more recent datacenter architectures, proposed to improve the conditions set by the current three tier architecture. These proposals include the use of modular switches and redundant links between servers [16] or more significant changes such as deploying the servers in a way that physically translates a structured network [17].

Acknowledgments

This work was partially supported by FCT - Fundação para a Ciência e a Tecnologia under the projects PEst-OE/EEI/LA0021/2011 and HCPI under the grant (PTDC/EIA-EIA/102212/2008). Parts of this work have been performed in collaboration with other members of the Distributed Systems Group at INESC-ID, namely, João Leitão.

References

[1] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM TOCS*, vol. 17, pp. 41–88, May 1999.

[2] J. Leitão, J. Pereira, and L. Rodrigues, "HyParView: A membership protocol for reliable gossip-based broadcast," in *Proc. of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, ser. DSN '07. Edinburgh: IEEE Computer Society, 2007, pp. 419–429.

[3] R. v. Renesse, K. P. Birman, and W. Vogels, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining," *ACM TOCS*, vol. 21, pp. 164–206, May 2003.

[4] A. Lakshman and P. Malik, "Cassandra: A decentralized structured storage system," *SIGOPS Oper. Syst. Rev.*, vol. 44, pp. 35–40, April 2010.

[5] R. v. Renesse, Y. Minsky, and M. Hayden, "A gossip-style failure detection service," Cornell U., Tech. Rep., 1998.

[6] The Amazon S3 Team, "Amazon S3 availability event: July 20, 2008," http://status.aws.amazon.com/s3-20080720.html. [Online]. Available: http://status.aws.amazon.com/s3-20080720.html

[7] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "HiScamp: self-organizing hierarchical membership protocol," in *ACM SIGOPS EW 2002*, Saint-Emilion, France, 2002, pp. 133–139.

[8] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh, "Efficient and adaptive epidemic-style protocols for reliable and scalable multicast," *IEEE TPDS*, vol. 17, no. 7, pp. 593–605, Jul. 2006.

[9] M. Matos, A. Sousa, J. Pereira, R. Oliveira, E. Deliot, and P. Murray, "CLON: Overlay networks and gossip protocols for cloud environments," in *On the Move to Meaningful Internet Systems, International Symposium on Distributed Objects, Middleware, and Applications (DOA)*, ser. Lecture Notes in Computer Science. Springer Verlag, 2009, vol. 5870, pp. 549–566.

[10] J. Leitão, J. Marques, J. Pereira, and L. Rodrigues, "X-BOT: A protocol for resilient optimization of unstructured overlays," in *Proc of the 2009 28th IEEE International Symposium on Reliable Distributed Systems*. Niagara Falls, NY: IEEE Computer Society, 2009, pp. 236–245.

[11] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "SCAMP: Peer-to-peer lightweight membership service for large-scale group communication," in *Networked Group Communication Workshop (NGC)*, ser. Lecture Notes in Computer Science. London, UK: Springer Verlag, 2001, vol. 2233, pp. 44–55.

[12] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *IMC 2010*, 2010, pp. 267–280.

[13] S. Voulgaris, D. Gavidia, and M. v. Steen, "CYCLON: Inexpensive membership management for unstructured P2P overlays," *Journal of Net. and Syst. Man.*, vol. 13, p. 2005, 2005.

[14] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, "From epidemics to distributed computing," *IEEE Computer*, vol. 37, no. 5, pp. 60 – 67, May 2004.

[15] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *P2P 2009*, Seattle, WA, pp. 99–100.

[16] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan, "Scale-out networking in the data center," *IEEE Micro*, vol. 30, pp. 29–41, July 2010.

[17] P. Costa, A. Donnelly, G. O'Shea, and A. Rowstron, "CamCube: A key-based data center," *Technical Report MSR TR-2010-74*, 2010.