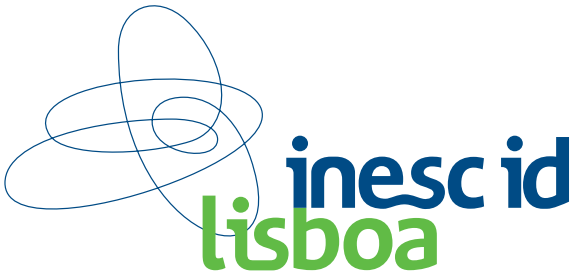

Technical Report RT/26/2012

Bounded Gossip: A Gossip Protocol for Large-Scale Datacenters

Miguel Branco
Instituto Superior Tcnico / IST
miguel.branco@ist.utl.pt

João Leitão
Instituto Superior Tcnico / IST
jleitao@gsd.inesc-id.pt

Luis Rodrigues
Instituto Superior Tcnico / IST
ler@ist.utl.pt

September 2012

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 paper 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.

Keywords: Gossip Protocols, Datacenter Networks, Topology-awareness, Flow Control

Bounded Gossip: A Gossip Protocol for Large-Scale Datacenters

Miguel Branco
Instituto Superior Técnico / IST
miguel.branco@ist.utl.pt

João Leitão
Instituto Superior Técnico / IST
jleitao@gsd.inesc-id.pt

Luís Rodrigues
Instituto Superior Técnico / IST
ler@ist.utl.pt

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 paper 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.

1 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 [2, 8], data aggregation [11], membership maintenance [3], among others [12].

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) [2]. 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 [13].

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 [5] , Hierarchical Gossip [7], 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 paper 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 paper is organized as follows: Section 2 discusses gossip-based protocols, and presents several works that have attempted to enrich gossip protocols with topology-awareness. Section 3 describes current datacenter network topologies. Section 4 motivates, presents, and describes Bounded Gossip. In Section 5 we experimentally evaluate and compare our work to previous solutions found in the literature. Finally, Section 6 concludes the paper.

2 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.

Topology-aware Peer-Sampling. The HiScamp protocol [5] is a distributed membership protocol designed on top of the Scamp peer sampling service [6]. 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.

Topology-aware Gossip. In contrast with the aforementioned solution, that works 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 [7]. 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 makes this solution highly inefficient from the point of view of 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.

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.

3 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.

Three-tiered Architecture. The most common architecture for large datacenter networks is the three-tiered architecture [1]. 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 [1]. In the remainder of the paper 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.

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 [7].

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. 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 (and thus the identity of the switches to which the node is connected indirectly).

4 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.

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

[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.

4.1 Overview

Our solution follows, and builds upon, the same intuitions behind the design of Hierarchical Gossip [7] 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 network 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 over 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 consideration the freshness of messages when forwarding them, as to ensure a low dissemination latency. Finally, the mechanisms above are complemented by a rate-based topology-aware flow control mechanism.

Our solution 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.

4.2 Membership Service

We rely on a gossip-based membership service that operates in a similar fashion to Cyclon [14]. 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.

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 $r > 0$ (edge) choose another node with whom they exchange view samples messages in each round, they bias their selection to promote exchanges with nodes on their PV_r view with a probability of 99%.

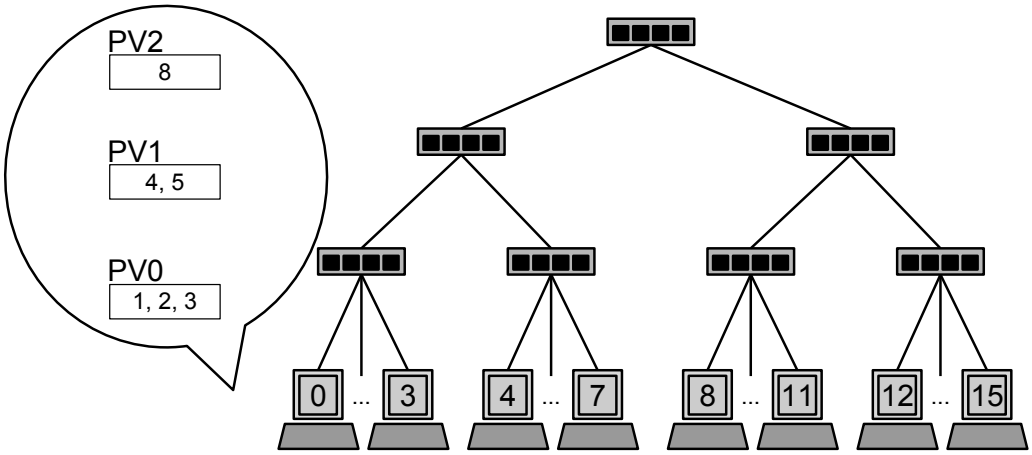

Figure 1. An example membership view of node 0

Figure 1 shows an example of how the partial views of a node are filled considering the underlying network topology.

4.3 Gossip-based Dissemination Scheme

Algorithm 1 denotes the pseudo-code for the dissemination procedure of Bounded Gossip.

We rely on a gossip-based dissemination scheme that operates in a partially 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 to more local levels. However, contrary to the Hierarchical Gossip [7] solution, nodes in our algorithm operate in the *infect and die* [4] 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 a parameter called *fanout*.

In order to apply the bias mentioned above, 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 [8]).

Finally, to ensure that there is a bounded communication overhead at each routing element of the datacenter network infrastructure, 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

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
```

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.

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 hierarchic 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 $14-26$).

• If $h >= r$, the node will start by sending the message to all the intended nodes with roles between h and r (if applicable). Then, the node performs 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 $27-52$).

This step however, is executed only by one of the R nodes responsible for level r in the cluster, using a

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      queue ← queue \ msg
```

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 (lines $61 - 63$). To maintain synchronization between nodes with the same roles, they execute the dissemination strategy associated with their role in a round-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, nodes reorganized among themselves by considering the node with the smaller identifier, across those that have sent messages in the previous round, as the correct sender. Because the current active node is important for the synchronization, nodes notify their peers with the same role even if they do not have any message to disseminate on that role in a round (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.

4.4 Flow-Rate Control

In order to ensure a bounded 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 $m * f \leq quota$. Evidently this assumes that $quota \geq f$. The quota of each node depends on the node's role in the cluster, as there are different quota values for each hierarchical level. At each level i, the configured quota depends on the target load tl_i such that: $q_i = \frac{tl_i}{Nclusters} \frac{f}{|PV_i|}$.

4.5 Fault Tolerance

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 the cluster roles, as to ensure R replicas per level.

5 Evaluation

To extract performance results of our solution we have simulated Bounded Gossip 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 [10], 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* [6]; *iii*) the *Hierarchical Gossip* solution operating with full membership information [7]; 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 a third of 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.

For our first experiment, we injected 800 messages in the system every 10 Gossip rounds, for a total of $5,600$ messages. We measured the core link load at each round and the message latency distribution for each protocol. Figure 2 shows the resulting core load, and Figure 3 shows the latency values. 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 remaining protocols. Moreover, the small quota of messages they are allowed to transmit penalizes latency in an unfeasible way when compared to those of Hierarchical Gossip and Bounded Gossip. The CLON1 configuration was not able to leverage the 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. However, it achieves latencies compared to those of the topology-oblivious solutions, as does the CLON2 configuration. The CLON3 configuration still has a maximum latency value close to the previous configurations but managed to deliver most of the messages in fewer gossip rounds, at the price of an external subscription system for nodes.

Hierarchical Gossip maintained the maximum core switch load during all the simulation, wasting more 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, with no latency penalty. A closer look at both protocols' behaviors is provided in Figure 4.

For the second experiment, we limit the total core switch load (dropping messages that exceed the configured limit). For each protocol, we then observe the maximum throughput without exceeding the load in this switch. Figure 5 illustrates the results.

It is visible that Bounded Gossip offers the best throughput when we limit the total core load, by a factor greater than 10 over the second best protocol. As expected, the results presented above, the flat gossip solution, Scamp and the CLON1 and CLON2 configurations cannot reliably disseminate any messages in this scenario.

In the third and final experiment, we show that despite the reduced core load, our solution does not affect negatively the dissemination reliability, achieving similar results to other systems even with 30% node failures.

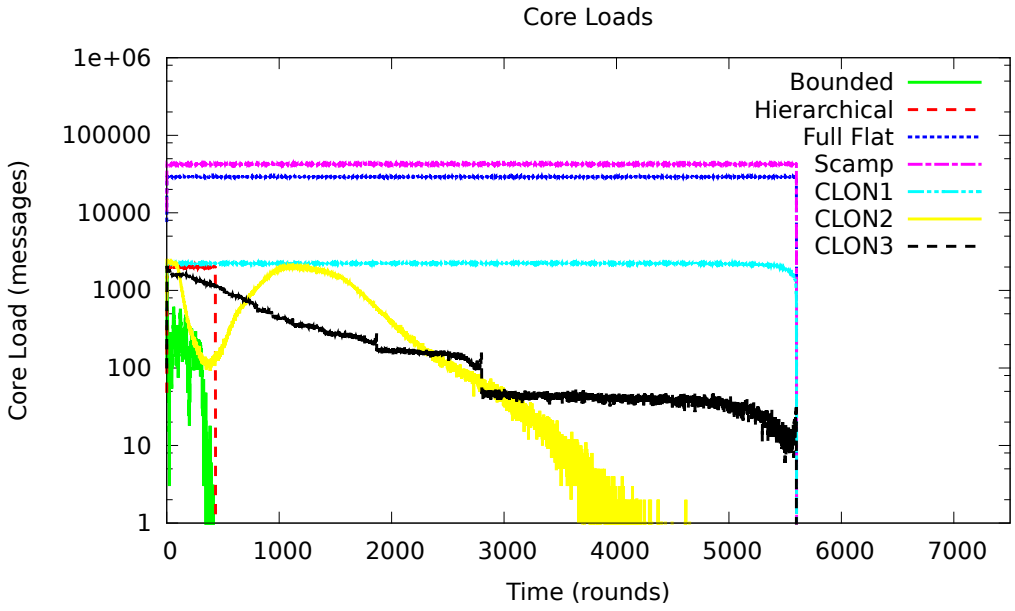

Figure 2. Core switch load for all protocols

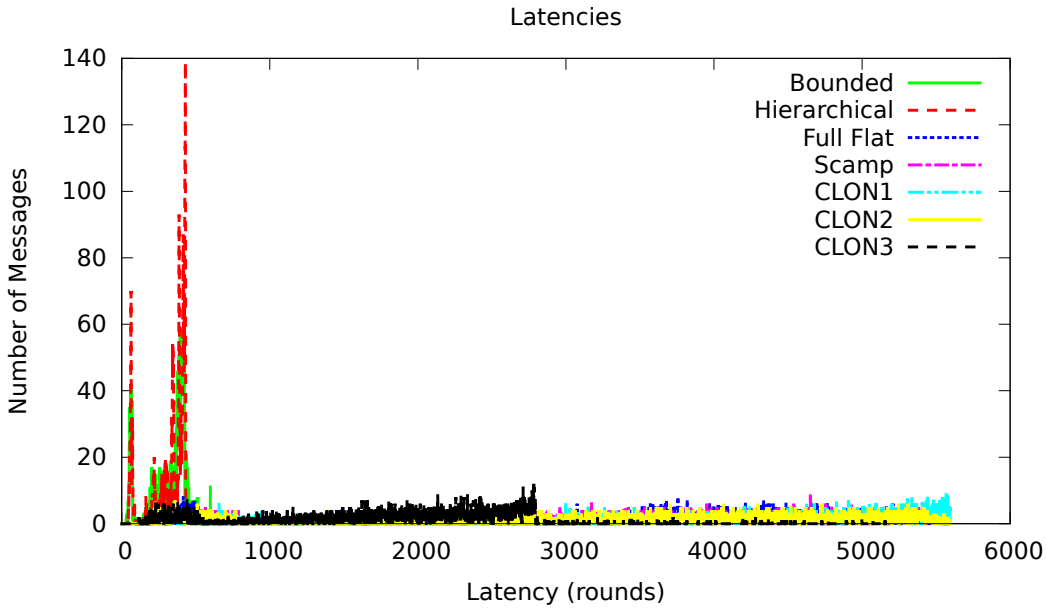

Figure 3. Latency measures for all protocols

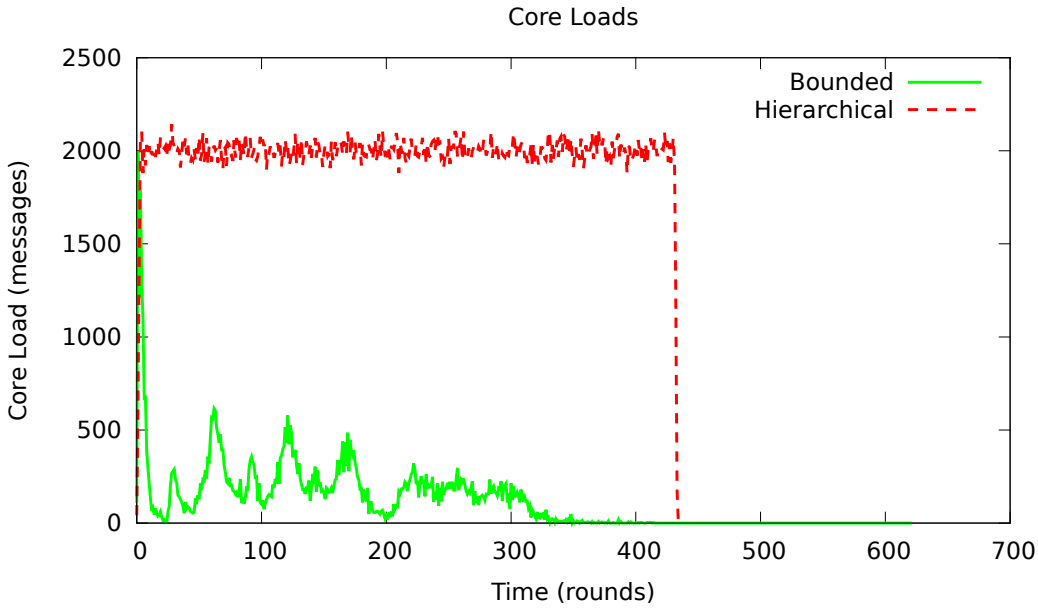

Figure 4. Core switch load (detail)

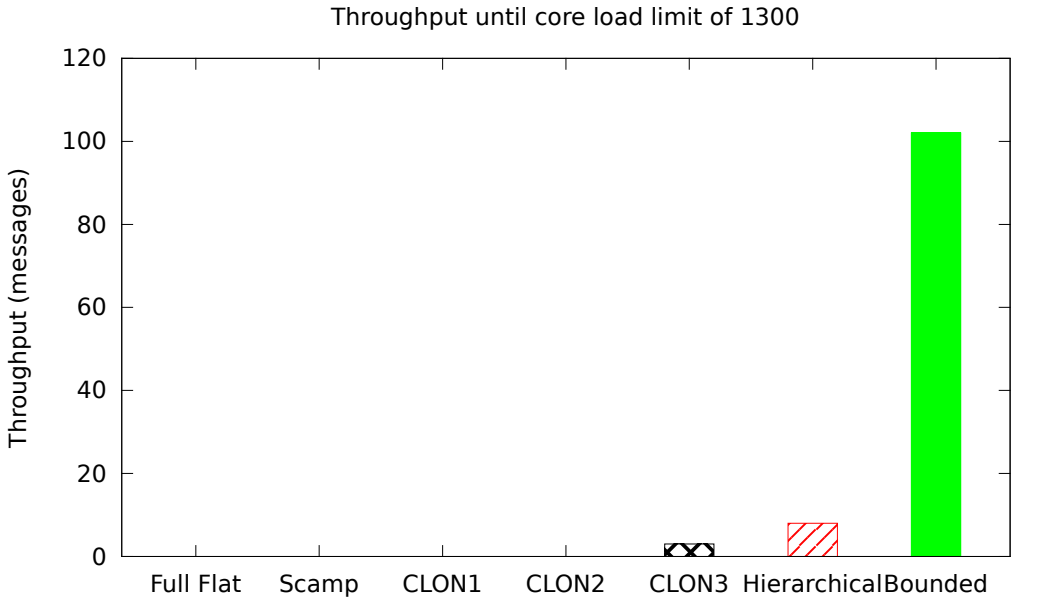

Figure 5. Throughput before core limit is reached

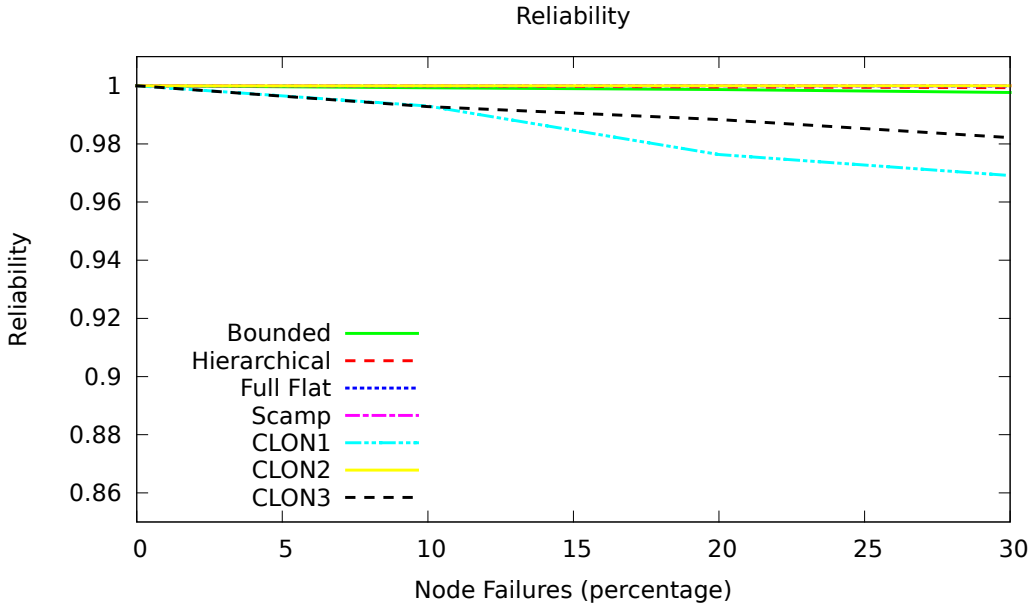

Figure 6. Reliability

We progressively injected messages in the system up to 5,600 messages, while failing a node per round until the desired limit was reached. Results are summarized in Figure 6.

6 Conclusion

In this paper we presented 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 10 times throughput with less core switch load per round and no penalty to overall latency or reliability.

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).

References

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

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

[3] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. *SIGOPS OSR*, 41:205–220, October 2007.

[4] P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie. From epidemics to distributed computing. *IEEE Computer*, 37(5):60 – 67, May 2004.

[5] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. HiScamp: self-organizing hierarchical membership protocol. In *ACM SIGOPS EW 2002*, pages 133–139.

[6] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. pages 44–55. Springer-Verlag, 2001.

[7] I. Gupta, A.-M. Kermarrec, and A. J. Ganesh. Efficient and adaptive epidemic-style protocols for reliable and scalable multicast. *IEEE TPDS*, 17(7):593–605, July 2006.

[8] J. Leitão, J. Pereira, and L. Rodrigues. HyParView: A membership protocol for reliable gossip-based broadcast. In *DSN 2007*, pages 419–429.

[9] M. Matos, A. Sousa, J. Pereira, R. Oliveira, E. Deliot, and P. Murray. CLON: Overlay networks and gossip protocols for cloud environments. In *DOA 2009*, MSP+09, pages 549–566.

[10] A. Montresor and M. Jelasity. PeerSim: A scalable P2P simulator. In *P2P 2009*, pages 99–100, Seattle, WA.

[11] 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*, 21:164–206, May 2003.

[12] R. v. Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. Technical report, Ithaca, NY, USA, 1998.

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

[14] S. Voulgaris, D. Gavidia, and M. v. Steen. CYCLON: Inexpensive membership management for unstructured P2P overlays. *Journal of Network and Systems Management*, 13:2005, 2005.