CLON: Overlay Networks and Gossip Protocols for Cloud Environments

Miguel Matos, António Sousa, José Pereira, Rui Oliveira, Eric Deliot and Paul Murray

`miguelmatos@di.uminho.pt`

Techn. Report DI-CCTC-09-13

2009, April

DI-CCTC-09-13

CLON: Overlay Networks and Gossip Protocols for Cloud Environments

by Miguel Matos, António Sousa, José Pereira, Rui Oliveira, Eric Deliot and Paul Murray

Abstract

Although epidemic or gossip-based multicast is a robust and scalable approach to reliable data dissemination in distributed systems, its inherent redundancy may result in high resource consumption both on links and nodes. This problem is aggravated in settings that have costlier or resource constrained links, as happens in Cloud Computing infrastructures composed by several interconnected large data centers across the globe.

The goal of this work is therefore to improve the efficiency of gossip-based reliable multicast used in infrastructure management systems by reducing the load imposed on those constrained links. In detail, the proposed CLON protocol combines an overlay that gives preference to local links and a dissemination strategy that takes into account locality. Extensive experimental evaluation using a very large number of simulated nodes shows that this results in a reduction of traffic in constrained links by an order of magnitude, while at the same time preserving the resilience properties that make gossip-based protocols so attractive.

Keywords: Cloud Computing, Overlay Networks, Epidemic Dissemination Protocols.

CLON: Overlay Networks and Gossip Protocols for Cloud Environments*

Miguel Matos[1], António Sousa[1], José Pereira[1], Rui Oliveira[1], Eric Deliot[2], and Paul Murray[2]

[1] Universidade do Minho, Braga, Portugal
miguelmatos@di.uminho.pt{miguelmatos,als,jop,rco}@di.uminho.pt
[2] HP Labs, Bristol, United Kingdom
miguelmatos@di.uminho.pt{eric.deliot,pmurray}@hp.com

Abstract. Although epidemic or gossip-based multicast is a robust and scalable approach to reliable data dissemination in distributed systems, its inherent redundancy may result in high resource consumption both on links and nodes. This problem is aggravated in settings that have costlier or resource constrained links, as happens in Cloud Computing infrastructures composed by several interconnected large data centers across the globe.
The goal of this work is therefore to improve the efficiency of gossip-based reliable multicast used in infrastructure management systems by reducing the load imposed on those constrained links. In detail, the proposed CLON protocol combines an overlay that gives preference to local links and a dissemination strategy that takes into account locality. Extensive experimental evaluation using a very large number of simulated nodes shows that this results in a reduction of traffic in constrained links by an order of magnitude, while at the same time preserving the resilience properties that make gossip-based protocols so attractive.

1 Introduction

Reliable Multicast is an important building block in distributed systems, as it offers a strong abstraction to a set of processes that need to communicate reliably. The concept of overlay is fundamental to Reliable Multicast as it abstracts the details of the underlying physical network by building a virtual network on top of it that can be seen as a graph where vertices represent the nodes and edges map to the links. Therefore, the overlay represents the 'who knows who' relationship among nodes.

To construct those overlays two main approaches exist: structured and unstructured protocols. The former is frugal in resource consumption of both nodes and links, but is highly sensitive to churn and failures. This is because the overlay is built as a spanning tree that takes into account optimization metrics such as latency or bandwidth. The dissemination tree is built before-hand and thus can take advantage of nodes and links with higher capacity. However, due to this pre-building, upon failures the tree must be rebuilt, precluding the dissemination while this process takes place. As such, in highly dynamic environments where the churn rate is considerable, the cost of constantly rebuilding the tree may become unbearable. Furthermore, nodes closer to the root of the tree handle most of the load of the dissemination, thus impairing the scalability of the approach. On the other hand, in the unstructured approach links are established more or less randomly among the nodes, without any efficiency criteria. Thus, to ensure that all nodes are reachable, links need to be established with enough redundancy, which has a significant impact on the overlay. First, as the overlay is redundant each node receives multiple copies of a given message through its different neighbors. This is due to the existence of multiple implicit dissemination trees, and as such any given message will be relayed through some of them. While this is undesirable from an efficient resource usage point of view, it yields strong properties: reliability, resilience and scalability. The first two come naturally due to the inherent redundancy in the establishment of links. As there are multiple dissemination paths available, failure does not impair the successful

* This work is supported by HP Labs Innovation Research Award, project DC2MS (IRA/CW118736).

delivery of a given message as it will be routed by some other available path. Furthermore, as there is no implicit structure on the overlay the churn effect is mitigated as there is no need to global coordination or rebuilding of the overlay. By requiring each node to know only a small subset of neighbors the load imposed on each one in the maintenance of the overlay and in the dissemination is minimized, which allows those protocols to scale considerably.

The key overlay properties, according to are [8]: Connectivity, that indicates node reachability and attests the overlay robustness; Average Path Length, that is the average number of hops separating any two nodes and is related to the overlay diameter; Degree Distribution, which is the number of neighbors of each node, and measures a node reachability and its contribution to the connectivity; and Clustering Coefficient, which measures the closeness of neighbor relations, and is related to robustness and redundancy.

The mechanism used to construct the overlay in the unstructured approach is known as the Peer Sampling Service [8], and several works before have focused on building such a service in a fully decentralized fashion [6,13,15,11,12,7]. With the abstraction provided by the Peer Sampling Service, peers wishing to disseminate messages, simply consult the service to obtain a subset of known neighbors, and forward those messages to them.

Dissemination on top of unstructured overlays typically use the epidemic or gossip-based approach. This approach relies on the mathematical models of epidemics [2]: If each infected element spreads its infection to a number of random elements in the universe, then all the population will be infected with high probability. The amount of elements that need to be infected by a given element - the fanout - is a fundamental parameter of the model, below that value the dissemination will reach almost none of the population, and above it it will reach almost all members, thus showing bimodal behaviour [3].

The actual gossip process, i.e. the decision of when and how to send the message payload to the neighbors may follow several approaches [10], which we describe next. The most common gossiping strategy is eager push, in which peers relay a message as soon as received to a number of targets for a given number of rounds, and is used by several well known protocols [5,9,14]. The major drawback of this strategy is the amount of bandwidth required, as multiple message copies are received by nodes. Oppositely, in lazy push, peers forward an advertisement of the message instead of the full payload. Peers receiving an advertisement of a new message could then ask the source for the payload, and achieve exactly once message payload delivery. Assuming that the message payload tends to be much larger than an advertisement with its id, this strategy drastically reduces the bandwidth requirement of the previous strategy but, increases the latency of the dissemination process, as three communication steps are needed to obtain the actual message content. Furthermore, this also has an impact on reliability as the additional communication steps increases the time window to network and node faults.

A different strategy relies on pulling, where nodes periodically ask neighbors for new messages. In the eager variant, when a node asks for news to a neighbor, the latter will send all new known messages to the petitioner. On contrast, in the lazy approach, also known as two-phase pull, the node that receives the news request only sends new message ids. The petitioner would then be able to selectively pull messages of interest.

The eager versus lazy strategy is clearly a trade-off between bandwidth and latency, while the difference between a push and pull scheme is more subtle. In push, nodes behave reactively to message exchanges, while on pull nodes behave in a proactive fashion by periodically asking for news. Thus, in an environment where messages are sparingly, a push strategy has no communication overhead, while the pull approach presents a constant noise due to the periodically check for news.

This paper comes in the context of the DC2MS project [1] whose goal is to provide a management service for Cloud Computing environments. The Cloud infrastructure is composed by several data centers spread worldwide and organized in a federation. The members of the federation are interconnected by long-distance expensive WAN links with high aggregate bandwidth demand, while the links that internally connect its components typically have less stringent requirements. The adequate management of such infrastructure is crucial to the success of Cloud Computing as the Cloud provider needs to be able to properly meter, bill, and abide by the Service Level Agreements of its customers among other essential management operations. Reliable Multicast appears as an important building block to the management of the infrastructure as it offers strong abstractions on top of which other essential services could leverage such as data aggregation, consensus and the actual dissemination of customer-related information.

The goal of this paper is therefore to build a reliable multicast service that is able to cope with the requirements of a cloud environment, namely its massive scale, the dynamics of the infrastructure where nodes constantly join and leave the system, the inherently federated infrastructure where the aggregate bandwidth requirements vary considerably, while offering strong reliability even in the presence of massive amounts of failures, as demanded by an infrastructure that needs to run 24/7. This is addressed at two distinct levels: the Peer Sampling Service which follows a flat approach that does not rely on special nodes or global knowledge but instead takes into account locality at construction time; and the dissemination protocol, which is also locality aware and can be configured to clearly distinguish between transmission to remote or local neighbors. By disseminating on top of the right overlay, and carefully choosing which strategy to use on a per node basis, we are able to reduce bandwidth consumption on undesirable links without impairing the resilience and reliability of both the overlay and the dissemination.

The rest of this paper is organized as follows: Section 2 describes existing protocols, how they relate to our work and why they fail do meet the requirements pointed above; Section 3 presents our proposal to address the aforementioned problems; Section 4 describes the experimental evaluation conducted to assess the effectiveness of the devised proposal and finally Section 5 concludes the paper.

2 Related Work

In this section we will briefly describe several overlay construction algorithms, and analyse how they relate to our work.

Scamp [6] is a peer-to-peer membership service with the interesting property that the average view size converges naturally to the adequate value by using local knowledge only. This is achieved by integrating nodes in the local view with a probability inversely proportional to the view size. With this mechanism and by sending several subscription requests for each node that joins the overlay, the protocol ensures that the view size converges to $(c + 1)log(N)$, where c is a protocol parameter related to fault tolerance and N is the system size. As pointed by its authors, Scamp is oblivious to locality and is a reactive protocol as it does not do any effort on the evolution of the overlay.

Cyclon [15] is a scalable and robust peer-to-peer overlay manager, which relies on a shuffling mechanism to promote link renewal among neighbors. In opposition to Scamp, Cyclon is a proactive protocol that continuously tries to enhance the overlay by means of periodic executions of the shuffle mechanism. The shuffle operation is very simple: each peer selects a random subset of peers in its local view and chooses an additional peer to which it will send this set. The receiving node also selects a subset of known neighbors and sends it to the initial node. After the exchange, each node discards set entries pointing to themselves and includes the remaining peers on their views, discarding sent entries if necessary. By including links in the exchange set accordingly to their age, the protocol is able to provide an upper bound on the time taken to eliminate links to dead nodes.

HyParView [11] also uses shuffling to build the overlay. However, each node maintains two views: a small active view with stable size used for message exchange; and a larger passive view maintained by shuffling and used to restore the active view on the presence of failures. By relying on a large passive view, the protocol is able to cope with massive failures, and by using a small sized active view the redundancy of message transmissions is reduced.

The Directional Gossip Protocol [12], aims at providing dissemination guarantees in a WAN scenario. To accomplish this, the authors adopt a two-level gossip hierarchy: the lower level runs a traditional gossip protocol in the LAN, and the other level is responsible for gossiping among LANs, through WAN links. The latter is achieved by using gossip servers, for each LAN there is a selected gossip server that is internally seen as yet another process. When a server receives a message from its LAN, it sends the message to the known gossip servers of the other LANs. When receiving an external message, it disseminates the message internally using traditional gossip protocols. While this protocol achieves good results in the amount of messages that cross WAN links, it relies on the undesirable selection of nodes with special roles, the gossip servers.

The Localizer algorithm [13], builds on the work done in Scamp by constantly trying to optimize the resulting overlay according to some proximity criteria. Periodically, each node chooses two nodes randomly, computes the respective link cost and sends those values to both. The receivers reply with their respective degrees and additionally, one of the nodes sends the estimate cost of establishing a link with the other. The initiator locally computes the gain of exchanging one of its links with one between the other nodes and, if desirable, the exchange is performed with a probability p, given by a function which weights the trade-off between the closeness to an optimal configuration and the speed of convergence. Localizer has not been deeply studied in presence of high churn rates and requires the interaction among three nodes to work properly. Furthermore, if the link exchange is successful, the initiator node behaves in a self-sacrificing manner as it loses one of its links.

HiScamp [7], is a hierarchical protocol that leverages on the work done in Scamp, by aggregating nodes into clusters according to a distance function. Joining nodes contact a nearby node and, based on the distance function, either join an existing cluster or start a new one. The protocol uses two views, an inView which is used to handle subscriptions, and a hView used to gossip. The hView has as many levels as the hierarchy, where the lowest level contains gossip targets in the same cluster, and the other levels contain targets on each hierarchy level. The inView has one lesser level than the hierarchy, is common to all nodes in the same level, and contains all nodes belonging to that level. Each cluster is seen as an individual abstract node on the next level, and in each level runs a Scamp instance that manages its overlay. To avoid the single point of failure of having a single node representing a cluster, an algorithm is run periodically to ensure that a given cluster is represented by more than one node. HiScamp effectively reduces the stress imposed on long distance links, but at the cost of decreased reliability. For instance, in the results presented by the authors, with more than 20% node failures the number of reachable nodes drops below 90%.

Scamp, Cyclon and HyParView are flat protocols that do not take into account locality and therefore fail to cope with the requisite of distinguishing links characteristics. This is important as we want to reduce the load imposed on the long-distance links that connect the members of the federation to increase the aggregate bandwidth available to them. Nonetheless they are highly resilient to churn and failures of links and nodes, and address our reliability concerns. On the other hand, Localizer, Directional Gossip and HiScamp are protocols that take into account locality and therefore are able to reduce the stress imposed on the long-distance links but unfortunately they are sensible to churn and failures as the experimental evaluation of the respective papers attest. This weakness precludes their use in scenario with requirements such as ours, and is due to the reliance on nodes with a special rolte to handle locality. Upon failure of those nodes, new nodes need to be selected for the special role to guarantee the connectivity of the members of the federation. However, this is hard to achieve in a fully distributed and dynamic environment as it requires some sort of distributed agreement to elect which nodes are special. Even if the special nodes are chosen in a probabilistic fashion, it is not clear how to get them to know each other and how to properly handle their failures, as it will imply some a priori-knowledge of which nodes are on the other locations of the federation to establish the long-distance links with them.

3 Algorithm Description

In this Section, we describe our reliable multicast service, whose goal is to provide the reliabiliy and resilience of flat protocols with the bandwidth efficiency of hierarchical approach to meet all the requirements presented in Section 1.

This is achieved at two distinct levels, the peer sampling service and the dissemination process. First, the peer sampling service builds an overlay that mimics the structure of the underlying network but without relying on special nodes as in the approaches presented previously or in any type

```
1  upon init
2    contact = getContactNode()
3    send(contact,handleSubscription(myself))
4
5  proc handleSubscription(nodeId)
6    for n ∈ view
7       send(n,handleJoin(nodeId))
8    for (i=0; i < c; i++)
9       n = randomNode(view)
10      send(n,handleJoin(nodeId))
11
12 proc handleJoin(nodeId)
13   keep = randomFloat(0,1)
14   keep = Math.Floor(localityBias(viewSize,nodeId)) * keep)
15   if (keep == 0) and nodeId ∉ view
16    view.Add(nodeId)
17   else
18      n = randomNode(view)
19      send(n,handleJoin(nodeId))
```

Listing 1.1: Clon protocol

of global knowledge. With this approach our protocol is able to tolerate considerable ammounts of failures and be resilient to churn as the traditional flat protocols. Finally, the dissemination protocol builds atop the peer sampling service and is responsible by the actual exchange of messages between peers. This protocol supports different dissemination strategies, that could be used to achieve different latency versus bandwidth trade-offs without endangering correctness. Additionaly, the dissemination also takes into account locality further reducing the load imposed on the costlier links.

3.1 Peer Sampling Service

The peer sampling service uses the same philosophy of the Scamp [6] protocol, namely the probabilistic integration of nodes and the injection of several subscription requests, which allows the average node degree to adjust automatically with the system size. However, instead of relying only on the view size of the node integrating the joiner, the protocol relies on an oracle to manipulate the view size perceived by the integration routine. Thus, nodes could be integrated with different probabilities based on the locality of the joiner, but nonetheless maintain the convergence and adaptability to varying system sizes. In detail, the oracle should provide higher virtual view sizes to remote nodes and therefore reduce their probability of integration, or lower virtual view sizes to achieve the opposite result. With an adequate configuration it is possible to manipulate the views of the nodes in order to have them know mostly local nodes and some remote nodes and thus bias the overlay to the underlying network topology. As all the nodes contribute to the network awareness, the protocol retains its reliability in face of faults as it does not depend on special nodes to handle it, while at the same time reducing the load imposed on the long-distance links, as fewer links to remote nodes are established when comparing to the traditional flat approaches.

In the rest of this Section we describe the Clon protocol, that can be observed in Listing 1.1.

Upon boot, a node obtains a contact node from an external mechanism and sends a *handleSubscription* to it (lines 1 to 3).[1] The receiver of the subscription creates a join request and forwards it to all nodes in its view, and to c additional random nodes (lines 5 to 10). A node receiving a join request (lines 12 to 19) generates a random seed and weights it with the value returned by the *localityOracle*. The oracle receives the view size and the id of the node joining the overlay and should return a value indicating the preference that should be given to the integration of the joiner, as pointed previously. If the calculation in line 14 yields zero, the joiner is integrated into the view of the node, otherwise the subscription is sent to a random node.

Finally, this service offers two calls to the dissemination protocol *PeerSampleLocal* and *PeerSampleRemote* that return a set of local and remote nodes, respectively.

[1] In fact the problem of how to know the initial contact node is still an open issue.

3.2 Dissemination Protocol

The Peer Sampling Service presented above addresses the requirements presented in Section 1, namely with respect to resilience, reliability and the reduction of the load imposed on the long-distance links. In this Section, we present a locality aware dissemination protocol with the goal of further reducing the load imposed on the long-distance links.

Instead or relying on a pure flood gossip protocol to relay messages on top of the overlay, the protocol takes advantage of the different dissemination strategies presented in Section 1 and the trade-offs that each one offers. Furthermore we also introduce the notion of locality in the dissemination process by using distinct rounds to disseminate message locally and remotely. The rationale behind this is that the number of nodes in a given local area and the number of local areas in the system will likely differ by some orders of magnitude (for example a scenario with 5 local areas and 200 nodes on each area), and therefore the number of rounds necessary to infect each one of those entities is quite different. Instead of coping with the necessity of reaching all the nodes with a higher global round number, we split that in a local round, that is increased when the message is disseminated locally, and a remote round that is increased when the message is disseminated to remote nodes. As such it is possible to infect some nodes on the remote areas and stop gossiping to them (as the round number as already expired) and let the local dissemination infect the remaining local nodes, thus reducing the number of messages that traverse the long-distance links. With this flexibility, we could disseminate the message only to a given portion of the population, lets say local, without wasting resources to send it to the other portion, lets say remote.

A dissemination protocol that combines the eager and lazy push strategies to achieve desirable bandwidth/latency trade-offs in a single protocol without endangering correctness [4]. In this work we take advantage of this properties of this protocol and extend it with the locality awareness concept presented above. The protocol is divided in two main components, one responsible for the selection of the communication targets, and the other by the actual point-to-point communication and selection of the transmission strategy.

The peer selection algorithm is show in Listing 1.2. Initially, the algorithm is started with an empty set of known messages, that is used to avoid delivering duplicates to the application via the *Deliver* upcall. An application wishing to send a message, calls the *Multicast* primitive on line 4 that generates a unique message id, initializes the rounds to zero and *Forward* the message. In *Forward*, the message id is added to the known set of messages, and the protocol enters the peer selection phase, from line 11 to 14, where the distinction between remote and local peers is made. The amount of peers specified by the respective fanouts is collected independently, if the respective round number has not expired, and then the $L - Send$ primitive of the point-to-point communication strategy selection is invoked for all the collected peers. When the point-to-point communication layer delivers a message to this level, via the $L - Receive$ upcall, the message id is checked against the known set of ids and, if the message is new, it is forwarded. The last important remark is the reset to the local message round if the message comes from a remote origin (lines 20 and 21). This is because messages being received remotely have a local round count that is meaningless to this local area and therefore must be reset to zero for dissemination to be successful locally. The *isExternal* oracle must then indicate whether the origin of the message is considered to be local or not.

In Listing 1.2 it is possible to observe the point-to-point communication part of the protocol. The $L - Send$ function called by the previous layer queries the strategy oracle *isEager* to infer whether the message should be sent in a eager or lazy approach. If the message should be sent eagerly, then a MSG is sent with the actual payload, otherwise an advertisement with the id is sent via $IHAVE$. Messages sent lazily may then be retrieved with the $IWANT$ call that will sent the actual payload to the requester (lines 23 to 25). If an advertisement is received (lines 13 to 15), the message is queued to be retrieved in a point in the future via the $ScheduleNext$ function. The requests are added to the retrieval queue in an order that puts request to local nodes first. Thus, if a request is already scheduled to retrieval from a remote node and the incoming advertisement is from a local node, the requests order is swapped (lines 31 to 38). This simple procedure further

```
initially
 K = ∅ /*known messages*/

proc Multicast(d)
 Forward(mkdId(),d,0,0)

proc Forward(i,d,ri,re)
 Deliver(d)
 K = K ∪ {i}
 P = ∅
 if re < maxRe
   P = P ∪ PeerSampleRemote(remoteFanout)
 if ri < maxRi
   P = P ∪ PeerSampleLocal(localFanout)
 for each p ∈ P
   L-Send(i,d,ri+1,re+1,p)

upon L-Receive(i,d,ri,re,s)
   if i ∉ K
     if isExternal(s)
        ri = 0
     Forward(i,d,r)
```

Listing 1.2: Dissemination Protocol: Peer Selection

reduces the transmissions on remote links, as the payload retrieval is first attempted on local nodes. The *isCloser* oracle simply compares the distance between the two potential sources and can be built upon the *isExternal* oracle defined previously. Upon reception of a new message (lines 17 to 21), the message id is added to the set of known messages, any pending requests on the message are cleared, and the payload is delivered to the peer selection layer via the $L - Receive$ upcall.

Finally, this layer of the CLON service will offer a *Multicast* primitive which application could use to disseminate their messages, and a *Deliver* upcall which will be used to deliver disseminated messages to the application.

4 Evaluation

On this section we will describe the experimental evaluation conducted in order to verify that the devised protocols address the requirements presented in Section 1.

All experiments have been run on a simple round-based simulator and assuming 1000 nodes distributed evenly among 5 local areas, that are connected to each other by long-distance links, i.e. links were we want to reduce the bandwidth consumption whithout impairing reliability.

4.1 Peer Sampling Service

In the first experiment, depicted in Figure 1, we analyse the properties of the overlays generated by Scamp and CLON , and the impact of each one in the reduction of the load imposed on the long-distance links. To access the reliability of both protocols in the presence of failures we devised three drop strategies that randomly remove nodes from the overlay, without healing, from 0% to 100%, in steps of 10%. The strategy $UniformDrop$ drops nodes from the universe of nodes in a random fashion. Additionally, *OneAreaDrops*/*TwoAreaDrops* remove nodes from one/two preselected local areas to access the contribution of a particular local area to the overall connectivity. Both protocols are configured with the $c = 7$ which makes the average view size 9. After observing the connectivity level of Scamp with this configuration, we configured the *localityOracle* of CLON such that the protocol provides the same reliability level as Scamp. It is important to notice that the oracle configuration should take into account the way the contact node is chosen. In our experiments the contact is chosen randomly across the set of existing nodes, and thus nodes will receive four more times subscriptions from remote nodes than local ones, as in this scenario we have four times more nodes than local ones. Therefore if we want to have the same amount of local and remote nodes in a view, the oracle should increase by four times the virtual view sizes when receiving subscriptions for remote nodes, to compensate for the greater amount of remote subscriptions received, and therefore match the desired ratio of remote and local nodes in a view.

Figure 1a depicts the evolution of both protocols when applying the dropping strategies presented above. As Scamp is not locality-aware, its views are composed, on average, by 6.8 and 2.2

```
initially
  ∀i: C[i] = ⊥
  R = ∅

proc L-Send(i,d,ri,re,p)
   if isEager(i,d,ri,re,p)
      send(p,MSG(i,d,ri,re))
   else
      C[i] = (d,ri,re)
      send(p,IHAVE(i))
  R = R ∪ {i}

upon receive(IHAVE(i),s)
   if i ∉ R
     QueueMsg(i,s)

upon receive(MSG(i,d,ri,re),s)
    if i ∉ R
      R = R ∪ {i}
      Clear(i)
   L-Receive(i,d,ri,re,s)

upon receive(IWANT(i),s)
   (d,ri,re) = C[i]
   send(s,MSG(i,d,ri,re))

forever
   (i,s) = ScheduleNext()
    send(s,IWANT(i))

proc QueueMsg(i,s)
   if i ∉ Queue
     Queue.add(i,s)
   else
     (i,source) = Queue.get(i)
      if isCloser(s,source)
        Queue.add(i,s)
        Queue.swap(s,source)
```

Listing 1.3: Dissemination Protocol:P2P Communication

remote and local nodes respectively, which reflects the fact that we have much more remote nodes than local ones. On the other hand, we observed that to maintain the same reliability level in CLON , it is only necessary to have views with 2.7 and 6.3 remote and local nodes respectively. It is important to note that the composition of the view is an essential metric to obtain the desired reliability level while at the same time reduce the load imposed on the long distance links. In fact, simply by changing the ratio of remote/local nodes in the view of the nodes, it is possible to directly affect the number of messages that traverse the long distance links, and consequently the load imposed on them. As it is possible to observe, with this configuration both protocols are able to tolerate up to 60% node drops, in the *UniformDrop* strategy without compromising the reachability of the alive nodes, being almost indistinguishable up to that amount of failures. Nonetheless after 70% failures both protocols fail to achieve the desired reachability level. The impact of localized failures in a given local area has no practical impact on the reachability, which means that local areas are inter-connected with enough redundancy to tolerate those localized failures. The lines depicting the connectivity of *OneAreaDrop* and *TwoAreasDrop* only go to 20% and 40% respectively, because that is the amount of global failures that a complete failure of one/two local areas represent.

On Figure 1b it is possible to observe the evolution of the clustering coefficient in presence of increasing drop rates. As expected, the clustering coefficient of CLON is a little higher than that of Scamp, because CLON tends to mimic the underlying network structure, which is inherently clustered among the local areas.

The average path length of the overlay could be observed in Figure 1c, and shows that CLON has a slightly larger average path length than Scamp. This comes directly from the fact that some nodes on CLON do not know remote nodes, although most of them do, as could be attested by the reachability results depicted in Figure1a. Due to this, some nodes have to do some extra hops to reach the entire overlay. However, as we will show latter, with the help of an adequate dissemination strategy the impact of this could be reduced.

Finally, Figure 1d depicts the number of messages exchanged when disseminating on top of both overlays, before applying any drop strategy. To this end, we used a naive eager push dissemination protocol, that just floods all its known neighbors, in an infect and die fashion. Every alive node multicast exactly one new message and after all messages have been delivered we count the number

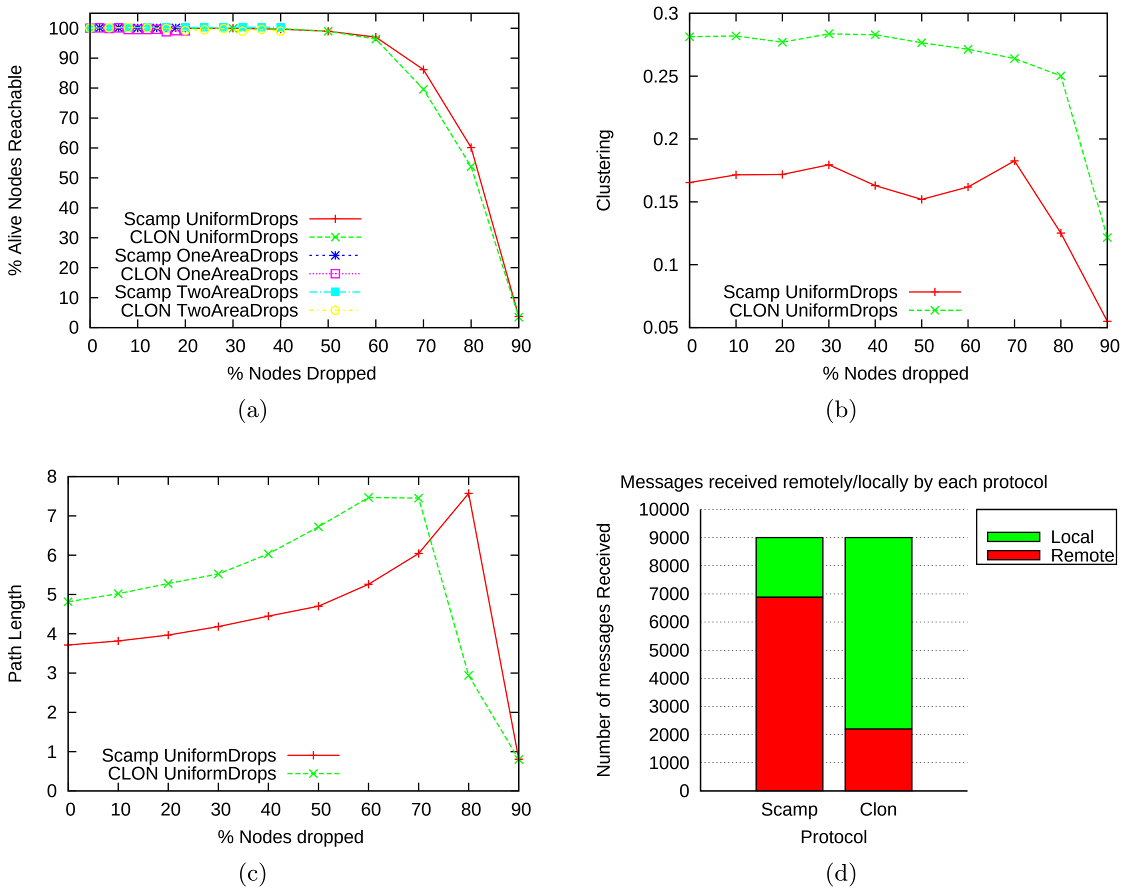
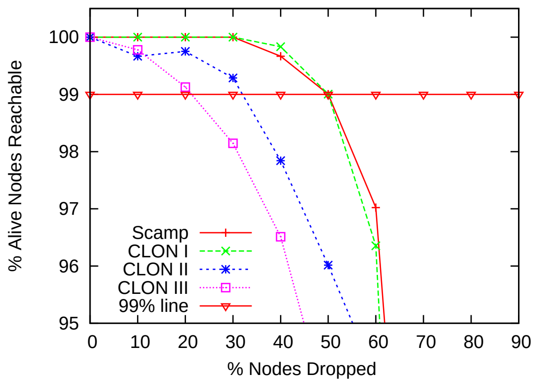


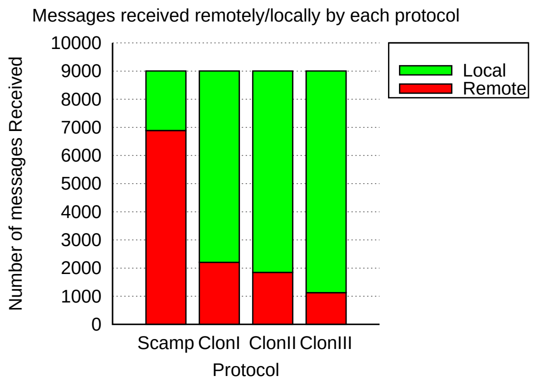
Fig. 1: Overlay properties and impact on the dissemination with a flooding protocol.

of messages received through remote and local nodes. The amount of total messages received in both protocols is around 9000, which reflects the injection of 1000 new messages on the system, one by each node, and the fact that they are sent on average 9 times by each node, as it is the average view size. As it is possible to observe, Scamp receives around 7000 messages through remote nodes, which reflects the average number of remote neighbors each node has. On the contrary, due to the biasing to the underlying network that gives preference to local links over remote ones, Clon receives around 2000 messages via remote neighbors, thus being able to achieve a reduction of more than 70% on the amount of messages that traverse the long-distance links, while tolerating the same amount of failures.

In the next experiment, depicted in Figure 2, we made the biasing to the underlying network more aggressive, by further reducing the number of remote nodes known, on average. While previously, the goal has to achieve the reliability of Scamp, which tolerates around 60% global node failures, here we intended to tolerate up to 30% and 40% failure rates, while guarantying that more than 99% of the alive nodes are reachable. On the left we have the reachability, as in Figure 1a, and in the right the respective amount of messages received locally and remotely. The labels Scamp and Clon refer to the previous configuration for reference purposes, while ClonII and ClonIII are configured to tolerate up to 40% and 30% global failures respectively. As it is possible to observe, ClonII (blue line on the left plot) tolerates up to 30% global failures with more than 99% confidence with a slight reduction on the number of remotely received messages (right plot). On the other hand, the configuration ClonIII tolerates up to 20% total failures with the same confidence level but further reduces the number of messages received remotely to slightly more than 1000. This experiment shows that with the adequate oracle configuration Clon is able to offer different reliability guarantees that directly impact the load imposed on the long-distance links. This trade-off is related to the amount of remote neighbors a given node has, with smaller values the reliability decreases as a local areas are more prone to become disconnected from each other but due the redundant nature of gossip protocols this also reduces the number of messages that traverse the long-distance links. Nonetheless, with a reliability level equivalent to standard flat overlay management protocols such as Scamp, Clon is able to significantly reduce the load imposed on the long-distance links, and contribute to increase the aggregate bandwidth available on them, as per the requirements presented in Section 1.



(a)



(b)

Fig. 2: Different oracle configurations.

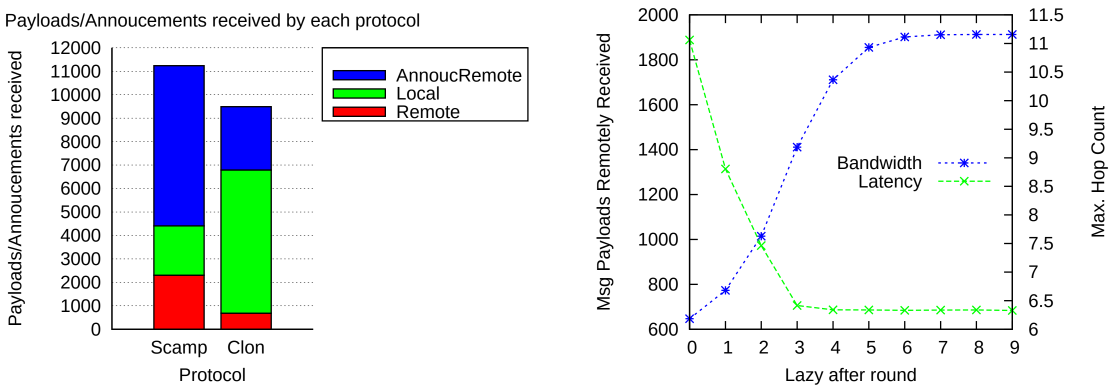
4.2 Dissemination Protocol

In this Section, we analyze the dissemination protocol proposed in Section 3. The protocol is run atop both Peer Sampling Services with the same configuration as the first experiment of the previous Section. To fully understand the advantages of the strategy combination, it is important to assume that message advertisements are considerable smaller than the actual payload, which holds true on many real scenarios. Anyway, if this assumption does not hold, the oracle could just be configured to a pure eager strategy as in this case the lazy strategy has no advantage over the eager one.

The results obtained are depicted in Figure 3. On the left the dissemination protocol is configured with a simple strategy: it uses eager gossip to local nodes and lazy gossip to remote ones. The rationale behind this strategy is to disseminate very fast on local areas, by means of the eager strategy, and send advertisements to remote areas. When a given local area lazily receives the payload of a new message, it will spread it quickly inside it. As more local nodes receive the payload eagerly transmitted, the queued requests for those payloads via remote nodes are dropped and therefore a considerable amount of message transmissions is avoided. Although the latency is considerable in this configuration due to the lazy pushing to remote nodes, the objective is to observe the lower bound on the number of messages that traverse the long-distance links. The improvement bring by this configuration of the dissemination protocol is clear as in both Scamp and CLON the number of messages received through remote neighbors (red bar) significantly decreases. In Scamp this value goes down from around 7000 with the flooding protocol to slightly more than 2000, whereas in CLON the value goes down to around 600 messages received remotely, an improvement of an order of magnitude if we consider the combination of Scamp with a flooding gossip protocol. As expected the number of messages received locally (green bar) is much higher than Scamp, due to the biasing to the underlying network, which gives preference to the local links, over the remote ones. Finally, the number of advertisements received is considerably higher in Scamp than in CLON , again due to Scamp having much more remote neighbors on average. The reduced number of announcements on CLON is due to two facts: first the average number of known remote nodes is smaller than Scamp and therefore fewer announcements are generated; and secondly, only a fraction of payloads are lazily pushed (and therefore the IWANT announcements sent) due to the prior infection by local nodes.

Finally, in Figure 3b, we analyse the bandwidth/latency trade offs offered by the dissemination protocol. The goal is to observe the impact of the chosen payload transmission strategy (by means of the *isEager* oracle, see Listing 1.3) on the latency and bandwidth consumption of the dissemination process. To this end we run a set of experiments where the *isEager* oracle returns False if and only if the target node is external and the external round is below a given threshold. The rationale is to transmit the message payloads eagerly for a certain number of rounds and then fall back to lazy strategy. In the experiment we varied the TTL value from 0 to 9, and for each value we run the dissemination protocol on top of the same overlay of the first experiment of the previous Section. On the X axis it is possible to observe the different TTL used for each run. As such on the leftmost part of the axis we have a completely lazy strategy that becomes gradually eager as we move to the right. On the left Y axis we measure the bandwidth consumption, blue line, with respect to the number of message payloads transmitted over the long distance links. On the right Y axis we measure the latency of the dissemination, green line, in the number of hops necessary to infect all nodes in the overlay. For instance, in the completely lazy strategy, i.e. when lazy after the round zero, nodes receive on average slightly more than 600 messages through remote links, which in fact is the experiment of the left With this configuration the latency to infect all nodes is 11 hops. As expected, the bandwidth increases with the eagerness to transmit the payloads, as more redundant messages are sent, while the latency decreases, as messages reach all nodes quicker, without the additional roundtrips of a lazy strategy. It is interesting to notice that in this scenario the latency reaches its minimum after 4 eager rounds, when it becomes close to the overlay diameter. On the other hand, the bandwidth tens to stabilize only around the 7^{th} round. Therefore, in this scenario using a eager strategy for more than four rounds will only waste bandwidth without bringing any improvement on the latency of the dissemination process.

The point where the two lines intersect presents an interesting trade off as it is when the bandwidth overhead required for the dissemination is small, with a moderate latency penalty. This account for around 1000 message payloads received remotely which is half of the value obtainable with a flooding protocol as the one used in Figure 1d. In fact, even if the latency should be reduced to a minimum, for instance by switching to the lazy push strategy after the third round, the impact on the bandwidth consumption on the long-distance links is still attractive as only around 1400 remote messages are received.



(a) (b)

Fig. 3: Latency/Bandwidth trade-off.

To finalize, we showed that by combining eager and lazy push strategies, it is possible to considerably reduce the stress put on resource constrained links. For instance, from the initial setting of having a flooding protocol on top of Scamp, to a combined dissemination protocol on top of Clon, it is possible to achieve a reduction on the number of payloads transmitted from more than 7000 messages to around 1400, while offering an attractive latency value. Furthermore, the excess of locally received messages on Clon, could have been mitigated by also using the eager/lazy strategy combination.

5 Conclusion

On this work we presented Clon, a Reliable Multicast Service that aims at reducing the bandwidth consumption on undesirable costlier links, such as inter data center links, while coping with the reliability and resilience required in such environments. We addressed the problem at two different levels: the peer sampling service which is related to the construction and maintenance of the overlay network, and the dissemination protocol which disseminates messages on top of the built overlay.

By taking into account locality at construction time, and by refusing to rely on nodes with special roles to handle locality, we obtained an overlay that is biased to the network topology, without compromising the key properties that ensure the reliability and robustness of unstructured protocols. On the dissemination protocol, we also take into account locality by having different rounds for remote and local nodes, and by offering to the programmer different latency/bandwidth trade-offs on a single protocol, just by the configuring an oracle to behave accordingly to the desired policy.

By relying on oracles to configure the protocol different trade-offs, and abstracting those particular configurations out of the model, we obtained a highly configurable service that can be used on a wide range of scenarios, from the low level of infrastructure management in a Cloud environment, to a added-value service offered to client applications.

The experimental results obtained are promising, as we achieved a reduction of an order of magnitude on the number of message payloads that traverse the long distance links.

References

1. DC2MS: Dependable Cloud Computing Management Services. http://gsd.di.uminho.pt/projects/projects/DC2MS, 2008.
2. N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications.* Hafner Press, second edition edition, 1975.
3. K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2):41–88, 1999.
4. N. Carvalho, J. Pereira, R. Oliveira, and L. Rodrigues. Emergent structure in unstructured epidemic multicast. In *Proc. of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 481–490. IEEE Computer Society, 2007.
5. P. T. Eugster, R. Guerraoui, S. B. Handurukande, P. Kouznetsov, and A.-M. Kermarrec. Lightweight probabilistic broadcast. *ACM Trans. Comput. Syst.*, 21(4):341–374, 2003.
6. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulie. SCAMP: Peer-to-peer lightweight membership service for large-scale group communication. In *Networked Group Communication*, pages 44–55, 2001.
7. A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié. Hiscamp: self-organizing hierarchical membership protocol. In *EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop*, pages 133–139. ACM, 2002.
8. M. Jelasity, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. The peer sampling service: experimental evaluation of unstructured gossip-based implementations. In *Proc. of the 5th ACM/IFIP/USENIX Intl. Conf. on Middleware*, pages 79–98. Springer-Verlag New York, Inc., 2004.
9. B. Kaldehofe. Buffer management in probabilistic peer-to-peer communication protocols. pages 76–85, Oct. 2003.
10. R. Karp, C. Schindelhauer, S. Shenker, and B. Vocking. Randomized rumor spreading. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, page 565. IEEE Computer Society, 2000.
11. 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*, pages 419–428. IEEE Computer Society, 2007.
12. M. J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. In J. Hlavicka, E. Maehle, and A. Pataricza, editors, *Third European Dependable Computing Conference*, volume 1667 of *Lecture Notes in Computer Science*, pages 364–379. Springer, 1999.
13. L. Massouli, A. marie Kermarrec, and A. J. Ganesh. Network awareness and failure resilience in self-organising overlay networks. In *In Proceedings of the 22nd Symposium on Reliable Distributed Systems*, pages 47–55, 2003.
14. J. Pereira, L. Rodrigues, R. Oliveira, and A. Kermarrec. Neem: Network-friendly epidemic multicast. In *In Proceedings of the 22nd Symposium on Reliable Distributed Systems*, pages 15–24. IEEE, 2003.
15. S. Voulgaris, D. Gavidia, and M. Steen. Cyclon: Inexpensive membership management for unstructured p2p overlays. *Journal of Network and Systems Management*, 13(2):197–217, June 2005.