Low Latency Probabilistic Broadcast in Wide Area Networks*

J. Pereira
U. Minho
jop@di.uminho.pt

L. Rodrigues
U. Lisboa
ler@di.fc.ul.pt

A. Pinto
U. Lisboa
apinto@di.fc.ul.pt

R. Oliveira
U. Minho
rco@di.uminho.pt

Abstract

In this paper we propose a novel probabilistic broadcast protocol that reduces the average end-to-end latency by dynamically adapting to network topology and traffic conditions. It does so by using an unique strategy that consists in adjusting the fanout and preferred targets for different gossip rounds as a function of the properties of each node. Node classification is light-weight and integrated in the protocol membership management. Furthermore, each node is not required to have full knowledge of the group membership or of the network topology. The paper shows how the protocol can be configured and evaluates its performance with a detailed simulation model.

1 Introduction

Probabilistic broadcast protocols, also called gossip-based or epidemic protocols, have been shown to scale to very large number of participants while

*Sections of this report will be published in the Proceedings of the 23rd Symposium on Reliable Distributed Systems, Florianopolis, Brazil, October 2004. The work was partially supported by LASIGE and FCT project RUMOR (POSI/40088/CHS/2001) via POSI and FEDER funds.

at the same time ensuring probabilistic reliability guarantees despite process failures and network omissions. By relying also on gossip based membership management, the scalability of such protocol can be realized in practice. Probabilistic protocols have been applied to a variety of different uses, from monitoring, management and data mining [15] to the support of large number of spectators in multi-user games [14].

Probabilistic broadcast protocols disseminate messages by gossiping: Upon reception, a message is relayed to a small random subset of participants. Therefore, to most recipients, delivery happens only after a message has been relayed a number of times by participant nodes. In consequence, although gossiping is highly resilient, it makes end-to-end latency be several times larger than the latency between any pair of nodes.

In heterogeneous environments, such as wide area settings, the high latency of probabilistic broadcast in exacerbated by the fact that most existing approaches are strictly symmetric, i.e., the traffic is approximately the same in every node. Therefore they do not exploit the fact that some nodes can sustain higher bandwidth than others. The gossip path between the sender and a given recipient may include bottleneck hops which add-up queuing delays.

The work reported in this paper is based on the hypothesis that the latency of probabilistic protocols may be improved if nodes with higher capacity (namely in terms of available bandwidth) offer a greater contribution to the message dissemination than those nodes with smaller capacity. The proposed protocol uses a novel strategy that consists in adjusting both the fanout and the preferred targets for each gossip round according to the properties of each node, thus reducing end-to-end latency both by reducing the number of rounds to delivery and the average hop latency. The resulting protocol, called *Lo*w *La*tency probabilistic broadcast ("LoLa pbcast", for short), is fully dynamic: it automatically adapts to network topology and traffic conditions.

Notice that blindly increasing the fanout or biasing the gossip procedure may lead to undesirable effects: The excessive traffic in preferred links, nodes or even in all the network may lead to congestion and additional queuing delays. It can also negatively impact the reliability of the protocol and harm other traffic sharing the network. LoLa does not suffer from these problems, as it includes feedback and adaptive mechanisms that prevent resources from being overloaded. The scalability of the protocol is also ensured by relying only on locally available information, exchanged using the gossip protocol

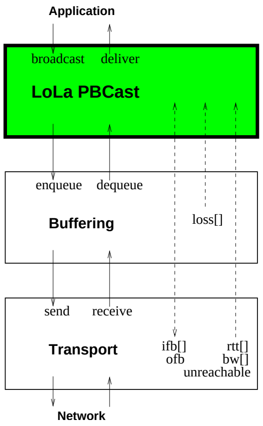


Figure 1: Architecture of the LoLa pbcast.

itself.

The paper shows how the protocol can be configured and evaluates its performance with a detailed simulation model. We show that the protocol effectively avoids the undesirable effect of having a good path to become overloaded because it is chosen so often, resulting in queuing delays or messages dropped. We also show that the biasing of gossip towards low latency does not negatively impact reliability guarantees, which rest on the assumption of uniformly distributed faults.

The remaining of the paper is structured as follows: Section 2 presents the protocol. Section 3 evaluates the proposed protocol with a detailed simulation model. Section 4 discusses related work and Section 5 concludes the paper.

2 Low Latency Probabilistic Broadcast

A probabilistic broadcast protocol [7, 1, 3] disseminates messages by gossiping. In each round, a message is relayed to a random subset of known participants. Each message is relayed a bounded number of times. It has been shown that such protocol results in probabilistic reliability: Each message is delivered with a high probability to almost all or almost none of the participants. By evenly spreading the load, such protocols are also scalable to large numbers of participants. The inherent redundancy makes the protocol tolerate network omissions and process failures.

LoLa differs from other gossiping protocols by dynamically tuning the contribution of each peer to gossiping according to its specific characteristics, in terms of available bandwidth and position in the network. This is achieved by combining several complementary mechanisms: *(i)* Use of TCP/IP connections among peers to promote a good usage of network resources, leveraging the architecture of NeEM [14]. *(ii)* Dynamic evaluation of each node's capacity to gossip a message with low latency. As a result of this procedure, some nodes are marked as *preferred* nodes. *(iii)* Weighted selection of membership, ensuring that each node keeps a subset of preferred nodes in its local membership (i.e., as potential targets for gossip). *(iv)* Weighted early gossip rounds, ensuring that a message is relayed to the preferred nodes in the first rounds of dissemination. *(v)* Adaptive fanout, allowing a node to augment its fanout value in early rounds, but dynamically adapting to network conditions to avoid congestion.

As it will be seen, the combination of these mechanisms makes LoLa operate as a self-organizing hierarchical epidemic broadcast protocol, in which high capacity nodes automatically form a network backbone responsible for message dissemination. This hierarchical structure emerges automatically from an unstructured peer-to-peer network: Messages in the early rounds are directed preferably towards preferred nodes and in later rounds are then disseminated outwards to leaf nodes.

In the following subsections, we start by describing the layered architecture of the protocol and the gossip procedure. Then we describe each of the previous mechanisms in isolation with more detail and we will later conclude the section by highlighting the interactions among them.

2.1 Protocol Overview

LoLa pbcast leverages the architecture of the NeEM protocol [14]. NeEM is a network friendly epidemic protocol that combines a standard gossip protocol with a specialized buffering layer and a transport layer that makes use of TCP/IP connections among peers to promote a good usage of network resources. NeEM makes no effort to reduce the average latency of the broadcast and instead improves throughput by taking advantage of message semantics.

In this paper we take the approach one step ahead and replace the standard gossip protocol used in NeEM with the LoLa pbcast, which uses feedback provided by the flow control mechanisms of TCP/IP, combined with other metrics such as round-trip delays, to assess the capacity of peers. This information is later used to bias the gossip procedure. LoLa pbcast also embodies the idea of maintaining only a partial membership of the system at each node [3]. However, the contents of the membership kept at each nodes is also biased according to the node's properties.

Following these principles, the architecture of LoLa pbcast is presented in Figure 1. At the bottom layers, TCP/IP is used to exchange messages with peers. Since each node only maintains a small fraction of the global system membership, the number of connections that need to be maintained open is small. LoLa also relies on additional information provided by the transport layer about the available network resources, as discussed in the next section.

On top of the TCP/IP layer, a buffering layer is used to accumulate messages when the network or the remote peer is congested. For each outgoing connection, a buffer queue is used. Upon overflow, messages are discarded according to their age [11] or, when all messages have the same age, at random. LoLa uses information about the amount of messages being discarded to feed the adaptive mechanisms.

Finally, on top of the queuing layer, a probabilistic protocol that implements a biased gossip is implemented. The gossiping protocol uses information provided by the buffering and transport layers about the status of the network resources and is presented in Figure 2. As usual in epidemic protocols, a message being broadcast (line 3) or locally received (line 10) for the first time is gossiped to a subset of known nodes (line 19). Each message is gossiped a finite number of times, by incrementing the age field and verifying it (lines 20 and 15). Duplicates are detected, and thus ignored, by keeping a set of ids of recently seen messages in variable *ids*. The gossip procedure uses the following configuration parameters:

- The default fanout value f_{min} and the maximum age age_{max} should be configured as in any other gossip protocol [1, 9].
- The maximum number of stored message identifiers $|ids|_{max}$ and garbage collection period T_1 should be set to avoid duplicate deliveries [3, 10].

Whenever node selection is required, either a uniformly random selection function, *uniform*, or a weighted random selection function, *weighted*, is used. Both functions receive the number of nodes to choose, and the set to choose from. The weighted function receieves in addition the weights to influence its choice. Note that the *ENQUEUE* and *DEQUEUE* functionality is that offered by the underlying buffering layer.

```
Initially:
   f= f_min, ids = ∅, peers = ∅, seen = ⊥, weight[]

upon BROADCAST(data)
   e.id ← unique id
   e.data ← data
   e.age ← 0
   DELIVER(e.data)
   ids ← ids ∪ {e.id}
   gossip(e)

upon DEQUEUE(e):
   { Deliver and gossip message }
   if e.id ∉ ids then
      DELIVER(e.data)
      ids ← ids ∪ {e.id}
      if e.age <age_max then
         gossip(e)
   { Store peer id }
   seen = e.peer

proc gossip(e):
   e.age ← e.age +1
   e.peer ← uniform(1, peers∪self)
   if e.age <age_min then
      dest ← weighted(f, peers, weight)
   else
      dest ← uniform(f_min, peers)
   for all peer ∈ dest do
      ENQUEUE(peer,e)

every T_1:
   { Garbage collect }
   if |ids| > |ids|_max
      ids ← ids \ oldest(ids)

every T_2:
   { Update weights }
   for all peer ∈ peers do
      weight[peer] ← ifb[peer]^p/rtt[peer]^q
   { Update feedback value }
   ofb ← ∑_peer bw[peer]

every T_3:
   { Update membership }
   peers ← peers\unreachable
   if seen ∉ peers ∧ seen ≠ self then
      if |peers| = |peers|_max then
         save ← weighted(|peers|_min, peers, weight)
         peers ← peers \ uniform(1, peers\save)
      peers ← peers ∪{seen}
   if f> |peers| then
      f ← |peers|

every T_4:
   { Update effective fanout }
   if loss = 0 then
      f ← f+1;
   else if loss_low < loss < loss_high then
      f ← f−1;
   else if loss > loss_high then
      f ← f_min;
```

Figure 2: Low latency probabilistic broadcast.

2.2 Node Evaluation and Preference

Central to the proposal is that nodes with different capacities should offer a different contribution to the gossip procedure. Low latency can be achieved by: *(i)* using short hops, thereby reducing the contribution to final end-to-end latency; *(ii)* using hops to nodes with large outgoing bandwidth, able to simultaneously communicate with a larger set of nodes, thus reducing the total required number of hops to delivery. Therefore, we need to dynamically assess the capacity of each target to reduce latency according to both criteria.

The latency of the immediate hop can be estimated by the round-trip time. Notice that the average round trip time can be obtained with no overhead whatsoever at the transport layer, as it is computed implicitly by TCP/IP and can be retrieved with the `getsockopt()` system call. This value is exported as *rtt[i]* by the transport layer.

The outgoing bandwidth of the target node cannot be directly estimated. Instead, a target node i estimates its own outgoing bandwidth for each connection *bw[j]* at the transport layer, thus for each node in its membership. The total outgoing bandwidth of node i is the sum of the bandwidth of all outgoing connections at i and is stored in variable *ofb* (line 37). This value is propagated back to all nodes with incoming connections, i.e. all nodes which which contain i in its local membership. Such nodes store the value in variable *ifb* at the transport layer.

During startup, when there is no available feedback from the transport layer, a very small bandwidth value is assumed (e.g. 1 byte/s). Also, if a node observes message losses in any of its transport connections, it immediately reduces its advertised bandwidth to the same value. This prevents a node from advertising more capacity than it is actually able to sustain. To prevent the control traffic from consuming excessive system resources, an update of the *ofb* value is not sent unless it has changed by more than some predefined threshold (in this paper, we have used 10% as the threshold value). Also, a node that is still observing message losses at the buffer level does not advertise any measured increment in bandwidth.

Using the feedback data discussed above, a preference value, simply named $weight[j]$, can be locally assigned to each target node. This weight is a function of the advertised bandwidth of j but also of the estimated round-trip time $rtt[j]$ between i and j, more precisely, every interval T_2, the weight is computed (line 32) as $\mathrm{weight}[j] = \frac{\mathrm{ifb}[j]^p}{\mathrm{rtt}[j]^q}$, where p and q are configuration parameters. The preference weight captures the fact that a "very close" peer (in terms of round-trip-time) may be a preferable target for gossip than a very "distant" peer, even if its capacity is not as high as the capacity of the distant peer. The intuition for this criteria is that very short hops introduce little or no negative effect in the overall end-to-end latency.

Larger values for both p and q make weights be farther apart, thus reinforcing bias. One may also select different values for p and q to balance between short hops to slow nodes and long hops to fast nodes. Since an excessive bias may compromise the natural resilience properties of epidemic broadcast, one typically should target to maximize performance with minimum bias, thus using small values for p and q. In Section 3.5 we will show, supported by experimental data, how p and q can be configured. In the rest of the paper we use $p = 3$ and $q = 3$.

2.3 Weighted Membership

As noted before, LoLa pbcast embodies the idea of maintaining only a partial membership of the system at each node. In [7, 14], the membership of each node is updated as follows. When a process gossips, it piggybacks in the gossip message information about a known node, uniformly selected among the nodes from its own membership and itself (line 21). When a process receives a gossip message, it merges these nodes into its membership list and then applies a function to reduce the membership size, in order to keep membership under a manageable size.

LoLa pbcast augments this procedure by introducing a bias in the membership management by selecting nodes to remove from the membership, as follows. When deciding which nodes to remove from the local membership, the node first marks a small sub-set of nodes as not eligible for removal (line 43). This sub-set of marked nodes is also selected using a weighed probability based on the preference weights. Nodes to be removed are then selected uniformly from the non-marked nodes. This means that preferred nodes are less likely to be removed.

Finally, LoLa requires that nodes with enough resources adjust its maximum membership size to a value larger than that strictly required for epidemic reliability. The limit can be set arbitrarily high, as flooding will adjust to a safe value. The choice is therefore mostly a policy decision of how much the node wants to contribute to the overall performance of the group, i.e., nodes with higher capacity should maintain a larger number of peers in their membership list than nodes with small capacity. There is one additional constraint. Nodes with very limited bandwidth should set the value low, to avoid the implicit buffering in connections and thus the resulting increased queuing latency. In our experiments the maximum membership size varies from a lower value of 15 to a maximum value of 40.

In summary, LoLa's weighted membership management can be tuned using:

- The maximum number of connected peers $|peers|_{max}$. In the current prototype, this value is set manually according to the policy described above. This procedure can in fact be replaced by a dynamic membership management protocol [5].
- The minimum number of preferred peers $|peers|_{min}$. The configuration of this parameter is discussed in Section 3.5 and a value of 5 used throughput this paper.
- The membership adjustment period T_2 should be large enough such that nodes can be evaluated. A value of 120s is used throughout the paper.

2.4 Weighted Early Gossip

In order to decrease the latency of the gossip procedure, LoLa pbcast bias the first k rounds of the protocol towards preferred nodes. In other words, instead of selecting the gossip target uniformly across the membership, LoLa weights the probabilistic selection of targets using the weight[] values. By limiting the bias to the early rounds of the gossip procedure we effectively prevent this mechanism from affecting the overall reliability of the gossip procedure. In fact, the bias is only applied when there are still few processes infected and, therefore, most targets (preferred or not) are valid targets. By eliminating the bias in subsequent rounds one ensures that all nodes have a uniform probability of being infected in later rounds. Additionally, the same strategy also prevents preferred nodes from receiving many duplicates of the same message. The reader should also notice that, when the network is homogeneous, this mechanism has no impact on the gossip procedure.

The choice of a low value for parameter k reduces bias and thus reduces the latency improvement. On the other hand, a large k might result in an uneven spread of messages and thus impact bimodal delivery guarantees. We discuss the configuration of k in Section 3.5 and use $k = 3$ throughout the paper.

2.5 Adaptive Fanout

An unique feature of LoLa is that the fanout used by each node during the first $k+1$ rounds of a gossip is not static but, instead, adjusted dynamically accordingly to the capacity of that node. The rationale for this is that nodes that have received a message up to round k have been perceived by other nodes as preferred and thus should make an effort to spread the message as much as possible.

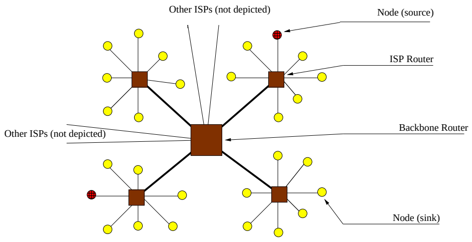
Let f_{min} be the minimum fanout to ensure the reliability properties of the epidemic broadcast (f_{min} is a logarithmic function of the total system size N; for a discussion on how to compute f_{min} see, for instance [1, 9]). Let also $|peers|$ be the size of the membership list at process i (we recall that, in LoLa, each node is configured with a different size of $|peers|_{max}$, according to its intrinsic characteristics). The fanout f used by a node during the first $k+1$ rounds of the gossip is a dynamically computed value within the interval $f_{min} \leq f \leq |peers|$.

The value of f is periodically adjusted as follows: *i)* initially, f is set to f_{min}; *ii)* if no message loss is detected during the last period, f is incremented; *iii)* if there is a small loss (below 2% of message loss) f is unchanged; *iv)* if there is a moderate loss during the last period (above 2% of message loss but below 5%), f is decremented; *v)* if there is a significant loss (above 5% of message loss) during the last period, f is reset to f_{min}.

This mechanism ensures that high capacity nodes effectively contribute more to the gossip procedure than low capacity nodes (by using a higher fanout). Notice that increasing the fanout value is useful only during the very first rounds, when only a few nodes have been infected and thus the increased number of messages results in useful deliveries. Using the same increased fanout in later rounds would only contribute to network congestion with little advantage, as most of the nodes are now infected.

In summary, the adaptive fanout mechanism can be tuned using the following parameters:

- The low watermark threshold $loss_{low}$ should be set very low in order avoid almost all losses. In this paper, a value of 2% is used.
- The high watermark threshold $loss_{high}$ should be large enough such that complete loss in a single connection, as resulting from a failed peer, does not preclude a large fanout. In this paper a value of 5% was used.
- The period T_3 should be large enough to avoid reacting on small fluc-



Link type	Uplink	Downlink	Latency
Backbone	10Mbps	10Mbps	10ms
ISP LAN	10Mbps	10Mbps	1ms
ADSL	512Kbps	128Kbps	80ms
V.90 Modem	56Kbps	33Kpbs	80ms

Figure 3: Experimental network topology.

tuations. In this paper, a value of 120s was used.

2.6 Summary of Interactions

We now briefly summarize the interactions among the LoLa mechanisms previously described. Due to the adaptive fanout mechanisms, high capacity nodes contribute with a higher fanout to the gossip procedure. However, this contribution is adjusted to their effective capacity (which is evaluated dynamically). By making the best use of their available bandwidth, high capacity nodes automatically increase their advertised bandwidth value *ofb*. This, in turn, will make high capacity nodes preferred and more likely to be used during the first rounds of the gossip.

By using TCP/IP to support the communication between peers, we ensure that the network capacity is not exceeded and the message loss in the network is eliminated. This allows nodes to accurately detect message loss in their gossip buffers and to automatically adjust both their fanout and their advertised bandwidth. Therefore, LoLa not only is able to adapt the gossip to an heterogeneous environment but also to dynamic changes in the network conditions.

It is worth noting that there is a substantial difference between the LoLa approach and a naive approach based on blindingly increasing the fanout at all nodes without any sort of feedback. In fact, it is obvious that the

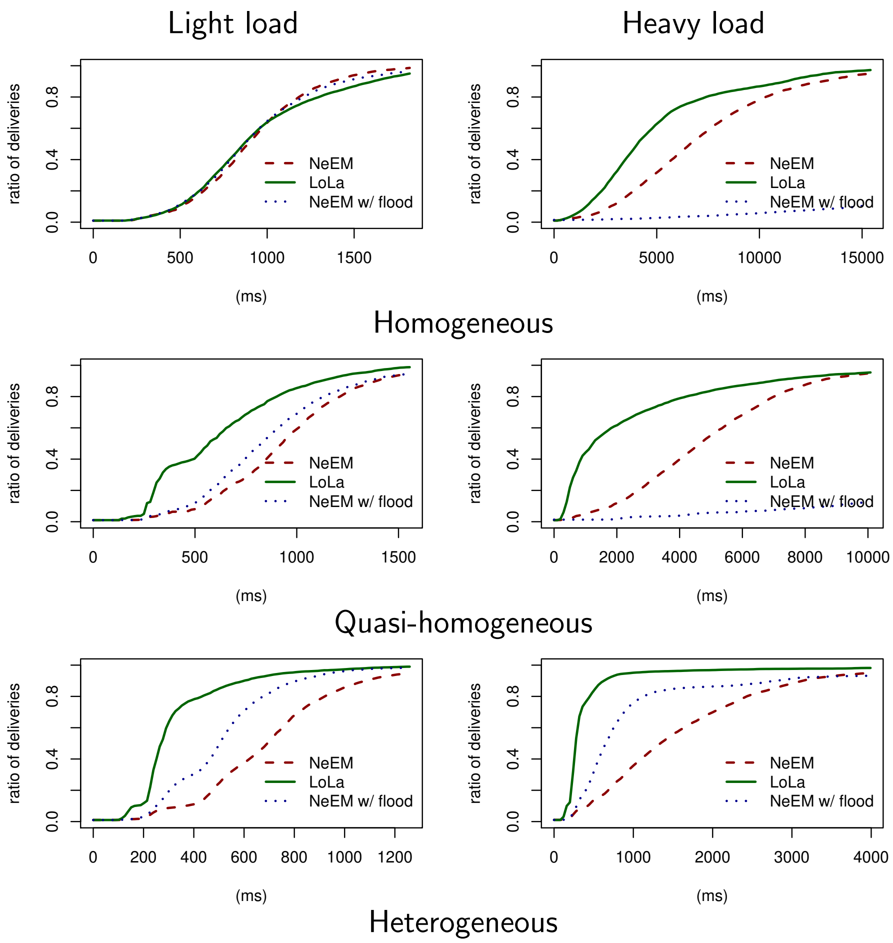


Figure 4: Distribution of latency.

simple choice of augmenting the fanout may reduce the end-to-end latency of the broadcast. However, such approach is only effective if the network does not suffer congestion. In a real setting, with heterogeneous nodes, if no adaptive mechanisms are embodied in the protocol, this may result in an over-utilization of system resources and may result in high message losses experienced by low capacity nodes. With our work we show that a careful combination of complementary adaptation mechanisms effectively promotes the best use of the heterogeneity that characterizes large-scale system. In the next section, we evaluate LoLa under different network scenarios.

3 Evaluation

We have evaluated the LoLa protocol through simulations, using the SSFNet[2] network simulation infrastructure. We first show the advantages of our protocol in heterogeneous scenarios by giving latency results for several network configurations. Later we offer a detailed analysis of relevant scenarios that illustrate specific features of our protocol. Finally, we also measure the impact of LoLa mechanisms in the atomicity of message delivery.

3.1 Network Topology

We run both homogeneous and heterogeneous scenarios using the same underlying network topology. The topology, illustrated in Figure 3 consists of a star-shaped network. At the core of the network there is a *central* router. This is connected to a number of ISP routers supporting both local area networks and wide area residential connections where gossip peers are hosted in the edges of the network.

Note that this topology captures and abstracts the hierarchical nature of today's networks. This underlying topology is not explicitly visible to the participants in the epidemic broadcast: When establishing a peer relationship, an outer node may peer with any other outer node. Any structure in the resulting peer-relationships will only emerge indirectly, due to the LoLa feedback mechanisms.

We assume that the links between central and ISP routers have a bandwidth of 10Mb/s. Different types of heterogeneity are obtained by varying the available bandwidth in the links connecting ISP routers to nodes. We have considered three types of relevant scenarios, all with 10 networks of 10 nodes, namely:

- *Heterogeneous scenario.* In this scenario, 10% of the outer nodes are connected trough a LAN within the ISP Ethernet links, 20% through ADSL links and 70% through standard V.90 Modem links.
- *Homogeneous scenario.* In this scenario, all nodes are connected trough standard V.90 Modem links.
- *Quasi-homogeneous scenario.* In this scenario, 5% of the outer nodes are connected through Ethernet links, and the remaining 95% are connected through standard V.90 Modem links.

The most realistic scenario is the first, in which link capacities are highly heterogeneous. This scenario provides several combinations of nodes with large bandwidth and varying delay (e.g. local and remote nodes collocated at ISPs), as well as, nodes with the same base delay and varying bandwidth (e.g. nodes attached by ADSL and V.90 Modems). The quasi-homogeneous is used to confirm that a small number of strategically placed fast nodes can be used to improve the performance of the broadcast protocol. The homogeneous scenario is used only as a baseline for comparison.

For all scenarios we have also experimented the performance of LoLa under two different load conditions. The load on the system was imposed by a fixed number of senders (5 nodes, in all scenarios) sending 100 byte messages at two different rates:

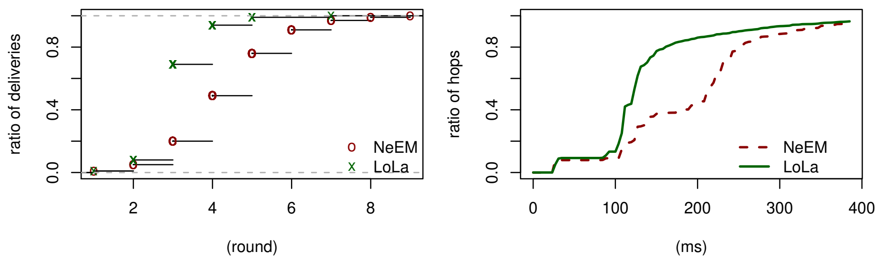
- *Light load scenario:* Messages sent every ten seconds by each sender;
- *Heavy load scenario:* Messages sent every second by each sender.

Note that the load caused by the heavy load scenario is enough to congest all V.90 Modem links although still providing reliable delivery (this is illustrated by Figure 7 which is presented later in the text).

When a comparative measure is useful, we use results obtained by running the NeEM protocol [14] in the same conditions as LoLa. We recall that NeEM also avoids network congestion, by a correct use of TCP but, in opposition to LoLa, makes no attempt to benefit from the network heterogeneity.

3.2 Achieving Low Latency

To illustrate the effectiveness of the protocol, we depict latency results for different system configurations in Figure 4. The x axis indicates end-to-end delivery latency (in ms) and the y axis the empirical cumulative distribution function (ECDF) of latency. Latency is measured by an external observer as the interval between application level broadcast and delivery.



(a) Rounds to delivery (b) Hop latency

Figure 5: Impact on hop count and latency.

For each configuration, we compare the latency of LoLa with that of NeEM. We also show the results achieved with a version of NeEM where the fanout is increased blindingly to a fixed high value. In result, this version of NeEM floods the network and is therefore denoted “NeEM w/ flood”: it illustrates why a naive solution, without feedback mechanisms, does not provides satisfactory results.

Figure 4 clearly shows the advantages of LoLa. For instance, it can be observed that in the heterogeneous scenario with heavy load, LoLa delivers approximately 90% of messages in less that 500ms while with NeEM, this value drops to less than 20%. It takes up to 3s to deliver the same share of messages. Notice also that the impact of “NeEM w/ flood” in congested networks is catastrophic, resulting in huge queuing delays. Even when there are plenty of resources, the reduction of average latency is less that with LoLa.

3.3 Detailed Analysis

We now provide a detailed analysis of the LoLa behavior in the most favorable of the considered scenarios, the heterogeneous network in which 30% of the nodes have enough resources. With this exercise we intend to provide the reader with a deeper insight of how the LoLa mechanisms contribute to the observed latency gains.

We start by showing, in Figure 5(a), the number of hops required to deliver the messages with LoLa and NeEM. It confirms that, with LoLa, messages require less hops to reach the destinations, namely, almost all messages have been delivered after 4 hops, down from 6 hops with the original protocol. This is due to high capacity nodes using their available bandwidth

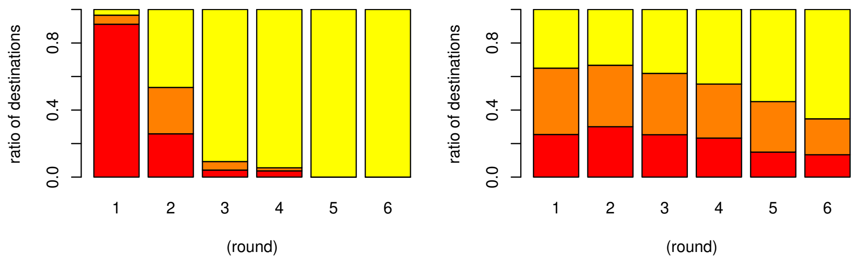


Figure 6: Impact of bias in the heterogeneous scenario: Dark to LAN; gray to ADSL; and light to modem.

to increase the fanout, as shown by the large increments in number of deliveries in the third hop. In addition, Figure 5(b) confirms that hops are also shorter, by selecting nodes that are closer and less congested or by originating from nodes with larger bandwidth and thus less queuing delays.

More interesting is the analysis of which type of links are used in each round of the gossip procedure. This is computed by recording all routes that lead to deliveries and then counting in each round the number of hops to each kind of node. This distribution is depicted in Figure 6 for the first six rounds (the results for later rounds is of no statistic relevance due to the small number of messages exchanged). The percentage of messages exchanged via high speed links is depicted in black, the messages exchanged via ADSL links in gray, and the messages exchanged via the modem connections in light gray. As it can be seen, with LoLa, high bandwidth links are mostly used in the first round, contributing to fast dissemination of the message. Notice that the bias for fast nodes in early hops still exists in the NeEM protocol, although it is much lower magnitude. This happens despite this protocol choosing destinations uniformly and is explained as follows: Messages that are relayed by faster nodes are more likely to hit nodes that have not been infected previously while messages that are relayed by slow nodes are still in transit. This means that in the end, as we consider only routes that lead to deliveries, such fast nodes are more likely to show up.

Finally, to highlight the importance of the adaptive mechanisms embodied in LoLa we compare the performance of the following three protocols: the NeEM protocol configured with an adequate fanout, the NeEM protocol configured with a fixed high fanout, and the LoLa protocol (with its adaptive

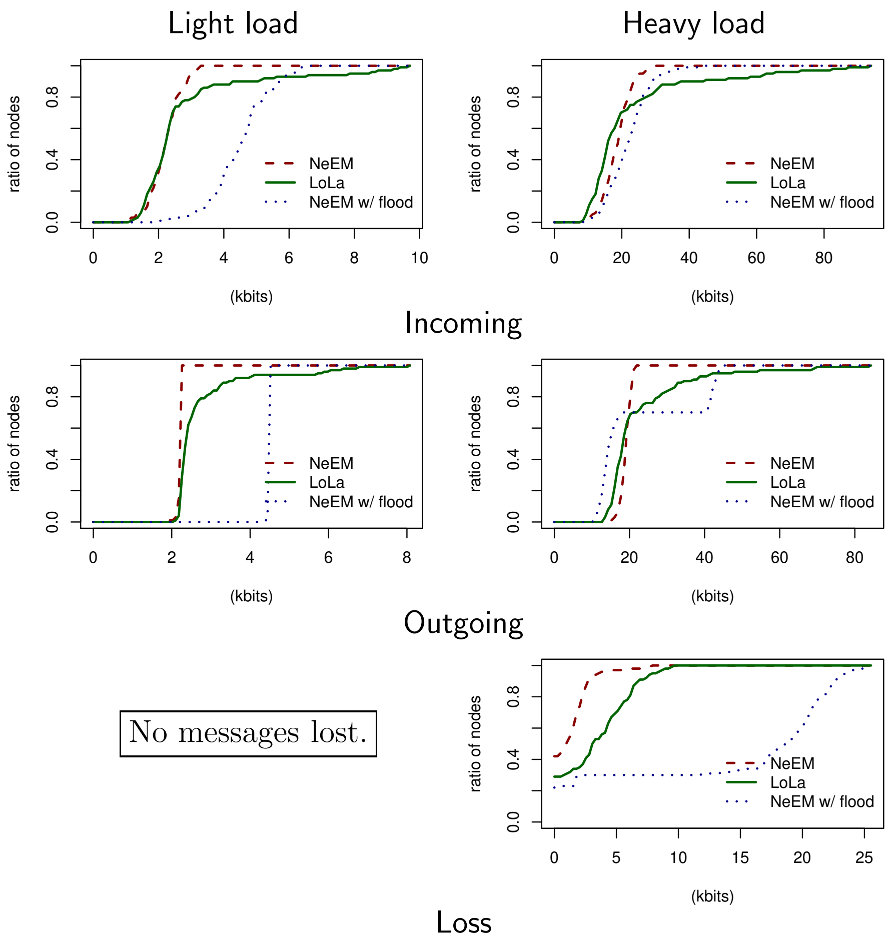


Figure 7: Bandwidth usage and losses.

	NeEM		LoLa	
	Heavy	Light	Heavy	Light
Before failures	97.17	98.25	98.38	99.45
After failures	94.39	97.34	97.66	98.82

Table 1: Impact on atomicity.

fanout). We analyze three relevant performance parameters: effective incoming and outgoing network bandwidth usage and measured message loss. Note that messages are discarded only within the buffering layer and not in the network.

The results are depicted in Figure 7. As expected, with the standard gossiping protocol in NeEM, the incoming and outgoing used bandwidth is approximately equal for all nodes, despite the large difference in available resources. Respectively, 2Kbits and 20kbits of payload with light and heavy loads. This happens because gossiping protocols are strictly symmetric. Notice also that considering the overhead of TCP/IP and gossiping headers of approximately 40%, modem uplinks with a raw 33kbits are saturated with the heavy load. In contrast, with LoLa up to 30% of nodes exhibit a much larger bandwidth usage, up to 10kbits with light load and 100kbits with the heavy load. This happens due to weighted gossip, as other nodes select fast nodes during the first rounds, and due to adaptive fanout at fast nodes.

If the fanout is statically increased, there is a much higher usage when available, *i.e.* with a light load or in fast nodes even with the heavy load. The drawback is however a large increase in dropped messages in the buffering layer, which compromises reliability. The large number of messages being dropped indicate also full buffers that result in additional queuing delays and increased latency.

3.4 Atomicity and Reliability

This section illustrates that the bias introduced by the adaptive mechanisms of LoLa does not decrease the reliability of the epidemic broadcast. Our quality metric in this respect is atomicity, defined as the average number of nodes that receive each message (in percentage). When comparing the atomicity of LoLa against NeEM, we can see that LoLa slightly outperforms the NeEM behavior. This is mainly due to the fact that, on average, LoLa uses a higher fanout than NeEM. The results are depicted in Table 1 using

	$q = 0$	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
$p = 0$	408	374	316	315	303	293
$p = 1$	301	317	294	286	283	287
$p = 2$	287	284	288	288	290	284
$p = 3$	314	288	285	280	290	286
$p = 4$	290	338	293	363	352	293
$p = 5$	368	374	370	363	359	359

(a) Heavy load

	$q = 0$	$q = 1$	$q = 2$	$q = 3$	$q = 4$	$q = 5$
$p = 0$	383	319	322	293	291	286
$p = 1$	346	308	287	296	279	281
$p = 2$	354	316	292	285	277	274
$p = 3$	306	288	285	277	278	280
$p = 4$	301	296	280	291	281	279
$p = 5$	288	288	287	290	276	284

(b) Light load

Figure 8: Latency with varying p and q (ms).

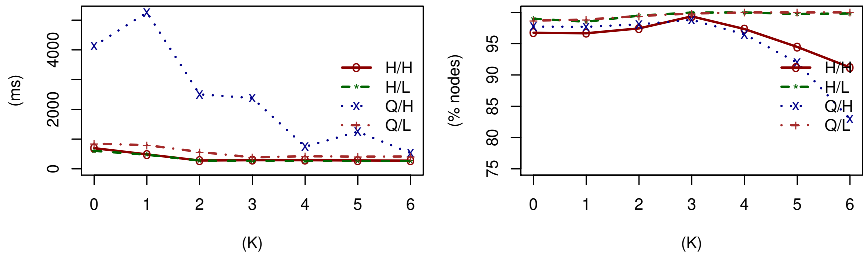
the quasi-homogeneous model.

We have also measured the comparative behavior of both protocols in face of node crashes. Specifically, 2 out of the 5 fast nodes and 2 of the regular nodes are simultaneously crashed in the simulation run. Both protocols use the status of the underlying TCP connections to monitor the activity of the peers. If a node crashes, it is excluded from the local view and replaced by other (correct) node as a result of the membership management embodied in the normal gossip procedure. Therefore, both protocols preserve the well known resilience of epidemic broadcast [1], as can be seen by the results depicted in Table 1.

3.5 Configuration Parameters

Finally, we discuss the rationale for setting the values of the most important LoLa configuration parameters. As it will be seen, an interesting feature of LoLa is that it is possible to select a set of values that provides good results for a wide range of operational conditions.

We start by discussing the importance of the bias parameters p and q in the computation of the preference weight. We recall that smaller values will result in less bias. In fact, setting both p and q to zero disables biasing altogether. It is therefore interesting to consider the minimum values that



(a) Latency (b) Atomicity

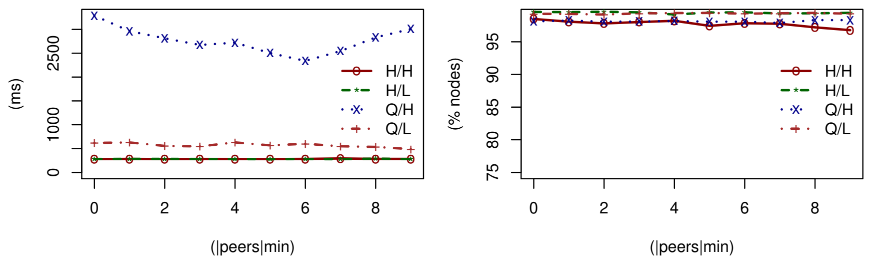
Figure 9: Configuration of parameter k.

result in good performance. Notice however that even without biasing, the adaptive fanout alone is still useful. The average latency using the NeEM protocol in the same network configuration is 1358ms and 697ms with heavy and light loads, respectively.

It can be observed that with a heavy load, the configuration with $p = 3$ and $q = 3$ results in good performance. It is also noteworthy that both criteria are useful. The same configuration is also adequate with a light load, although good results can be obtained with each criterion alone.

It is also interesting to consider the optimum value for parameter k, the number of biased hops, which determines also the number of hops with adaptive fanout. Figure 9 presents atomicity and latency results with heterogeneous (H/*) and quasi-homogeneous (Q/*) models, with both heavy (*/H) and light (*/L) loads. When using a light load, an arbitrarily large value for k is acceptable, although $k = 3$ is enough for low latency. With heavy loads, a value of $k > 3$ severely impacts atomicity as the fanout is reduced due to congestion and thus biasing makes it likely that slower nodes do not receive all messages. Note that the reduction of average latency observed with $k > 3$ with a heavy load in the quasi-homogeneous model (Q/H) is obtained at the cost of missing deliveries to slow nodes.

Finally, we have also experimented with different values for parameter $|peers|_{min}$ in the various network and load scenarios. Figure 10 presents atomicity and latency results. As expected, when there is a number of fast nodes such that there is a high probability of each local membership containing at least one of them, the value of $|peers|_{min}$ is irrelevant. This is the case of the heterogeneous scenario (H/*). In fact, a large value of the parameter has a slight impact in atomicity with a heavy load. When there is a very small



(a) Latency (b) Atomicity

Figure 10: Configuration of $|\text{peers}|_{min}$.

number of fast nodes, as happens in the quasi-homogeneous scenario, and especially when there is a heavy load on the network, the $|peers|_{min}$ parameter becomes relevant and a value of 5 is enough to ensure good performance.

4 Related Work

There have been several different proposals to take advantage to network topology to improve the performance of gossip based protocols. HiScamp [4] is a decentralized and scalable membership management protocol which provides hierarchical gossiping. This approach has however a severe impact in protocol latency in order to reduce the bandwidth used in backbone links. In addition, although several possible criteria are proposed for structuring the group, the evaluation is done with a static configuration.

The Directional Gossip [12] is also targeted at minimizing the network traffic in network links. It depends however on explicitly discovering network topology and is not targeted at improving latency.

In [6], a set of solutions to avoid the lack of adaptivity and the high network overhead of gossip are discussed. These solutions are based on a hierarchy defined on the process group called Leaf Box Hierarchy. Another hierarchical approach is described in [8], where nodes are grouped into clusters according to geographical or network proximity. [13] presents an algorithm to refine the overlay network to reflect geographic locality. These protocols imposes lower network overhead than flat gossiping but suffer from a small decrease in reliability and also small increase in latency.

The same direction has been taken in order to improve certain valuable services by using gossip. The goal of [16] is to develop a failure detector based on gossiping in order to improve scalability. The protocol also shows how members can avoid the amount of redundant information in routers and bridges by automatically detecting the bounds of Internet domains and subnets and reducing gossips that cross these bounds. The protocol also allows for accelerated detection times within subnets and is more resilient against network partitions. On the down side, it has a negative impact in the total latency of the entire system.

5 Conclusions

The LoLa protocol can be seen as an adaptive self-organizing hierarchical epidemic broadcast protocol, in which high capacity nodes are the backbone of the network. When a message is first sent, the protocol tries to route it first to faster nodes, which will then use all their available bandwidth to quickly and reliably disseminate the message to the remaining nodes. With LoLa's adaptive mechanisms, a structure emerges probabilistically from an unstructured network. The usage of TCP/IP to indirectly assess network conditions and the resulting feedback mechanisms ensure that, in the worst case scenario, the protocol defaults to a conventional flat epidemic protocol, thus preserving reliability. We have evaluated our protocol under several network conditions. Experimental results have shown that our approach effectively reduces the latency of the gossip protocol, not only in heterogeneous environments, but also when the system is subject to a high load. This is achieved without negatively affecting the gossiping performance in the scenarios where the adaptive mechanisms are not required (namely, in homogeneous and lightly loaded networks). As future work we plan to devise strategies to automate the adjustment of the configuration parameters used by LoLa.

References

[1] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Trans. Computer Systems*, 17(2), May 1999.

[2] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards realistic million-node internet simulation. In *Proc. Intl. Conf. Parallel and Distributed Processing Techniques and Applications (PDPTA)*, 1999.

[3] P. Eugster, R. Guerraoui, S. Handrukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *Proc. IEEE Intl. Conf. Dependable Systems and Networks (DSN)*, 2001.

[4] A. Ganesh, A. Kermarrec, and L. Massoulie. Hiscamp: self-organising hierarchical membership protocol. In *Proc. European ACM SIGOPS Workshop*, 2002.

[5] A. Ganesh, A.-M. Kermarrec, and L. Massoulie. Peer-to-peer membership management for gossip-based protocols. *IEEE Trans. Computers*, Feb. 2003.

[6] I. Gupta, A.-M. Kermarrec, and A. Ganesh. Efficient epidemic-style protocols for reliable and scalable multicast. In *Proc. IEEE Intl. Symp. Reliable Distributed Systems (SRDS)*, 2002.

[7] M. Hayden and K. Birman. Probabilistic broadcast. Technical Report TR96-1606, Cornell University, Computer Science, 1996.

[8] A. Kermarrec, L. Massoulie, and A. Ganesh. Probabilistic reliable dissemination in large-scale systems. *IEEE Trans. Parallel and Distributed Systems*, 14(3), 2003.

[9] A.-M. Kermarrec, L. Masssoulié, and A. Ganesh. Reliable probabilistic communication in large-scale information dissemination systems. Technical Report 2000-105, Microsoft Research, 2000.

[10] B. Koldehofe. Buffer management in probabilistic peer-to-peer communication protocols. In *Proc. IEEE Symp. Reliable Distributed Systems (SRDS)*, 2003.

[11] P. Kouznetsov, R. Guerraoui, S. Handurukande, and A.-M. Kermarrec. Reducing noise in gossip-based reliable broadcast. In *Proc. IEEE Symp. Reliable Distributed Systems (SRDS)*, 2001.

[12] M.-J. Lin and K. Marzullo. Directional gossip: Gossip in a wide area network. In *Proc. European Dependable Computing Conference (EDCC)*, 1999.

[13] L. Massoulie, A. Kermarrec, and A. Ganesh. Network awareness and failure resilience in self-organising overlay networks. In *Proc. IEEE Intl. Symp. Reliable Distributed Systems (SRDS)*, 2003.

[14] J. Pereira, L. Rodrigues, M. J. Monteiro, R. Oliveira, and A.-M. Kermarrec. Neem: Network-friendly epidemic multicast. In *Proc. IEEE Symp. Reliable Distributed Systems (SRDS)*, 2003.

[15] R. van Renesse, K. Birman, and W. Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. *ACM Trans. Computer Systems*, 21(2):164–206, May 2003.

[16] R. van Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. Technical Report TR98-1687, 28, 1998.