Reducing Noise in Gossip-Based Reliable Broadcast*

P. Kouznetsov[1] R. Guerraoui[1] S. B. Handurukande[1] A.-M. Kermarrec[2]
[1] Distributed Programming Laboratory
Swiss Federal Institute of Technology, Lausanne, CH 1015, Switzerland
[2] Microsoft Research Ltd., Cambridge, UK

Abstract

We present in this paper a general garbage collection scheme that reduces the "noise" in gossip-based broadcast algorithms. In short, our garbage collection scheme uses a simple heuristic to trade "useless" messages with "useful" ones. Used with a given gossip-based broadcast algorithm, a given size of buffers, and a given number of disseminated messages (e.g., per gossip round), our garbage collection scheme provides higher overall reliability than more conventional schemes. We illustrate our approach through two algorithms: Bimodal Multicast (pbcast) [1] and Lightweight Probabilistic Broadcast (lpbcast) [5].

Our scheme is based on the intuitive idea of discarding messages according to their "age". The "age" of a message represents the number of times the message has been retransmitted. Roughly speaking, if you have to choose among a set of jokes to memorize, you might probably not choose the one you have heard the most often: it is very likely that there will be someone in your audience to already know this joke.

1 Introduction

Context. Traditional reliable broadcast algorithms (in the sense of [6]) might be considered efficient in the context of local area networks but they do not scale to large settings. Network-level algorithms, such as IP Multicast [3], provide a limited form of reliability, that degrades drastically with the scale of the system.

Gossip-based broadcast algorithms represent an adequate solution to the general problem of broadcasting information among a large group of participants. Their scalability relies on a decentralized peer to peer interaction model where each participant gossips received messages periodically to a set of other participants. These algorithms differ according to the period of gossip, the number of times a message is gossiped and the choice of participants to gossip to. The redundancy, inherent to gossip-based algorithms, helps to achieve a high degree of reliability without any centralized mechanism.

Motivation. The buffering and periodic exchange of messages tend, however, to hamper the global scalability from the network perspective. Gossip messages might include out-of-date items, or "noisy" messages, that unnecessarily load the network. An obvious challenge in gossip-based algorithms is to reduce this "noise" without decreasing the overall reliability. Messages need to be discarded anyway, and the difficulty is to find the most appropriate messages to discard, i.e., messages that have been already delivered by "enough" processes, instead of messages that have not. This is a non-trivial issue given that no global information is available. An improper purging mechanism, which would choose useful messages to remove instead of noisy ones, can compromise reliability guarantees. At first place, a reasonable approach consist in purging "old" messages and keep recently published ones.

In a large scale and dynamic setting, the notion of time is hard to measure and might even be meaningless. A message m_1 might have been broadcast at time t but, because of network partitions and/or randomization, m_1 might have only been forwarded a couple of times and might have consequently been delivered only by very few processes. Another more recent message m_2, broadcast at time $t + \Delta$t, might have been retransmitted many times, and might have consequently reached a larger number of processes.

Contributions. We propose in this paper a general garbage collection scheme for gossip-based reliable broadcast algorithms, based on the "age" of messages. Our notion of message "age" captures a logical notion of time. More precisely, the age of a message is the number of times the message has been already gossiped. The age information is carried by messages themselves and computed on each process. Each process is able to take decisions about the garbage collection independently, which is crucial in decentralized systems. Our age-based garbage collection is then

*This work is supported by the Swiss National Foundation and Microsoft Research.

inherently scalable.

2 Garbage collection in gossip-based broadcast algorithms

2.1 Reminder: gossip-based broadcast algorithms

We briefly present in this section two variants of gossip-based broadcast algorithms through which we shall illustrate our garbage collection scheme.

Bimodal Multicast (*pbcast*)[1] is composed of two sub-protocols structured roughly as in the Internet MUSE protocol [7]. The first is an unreliable, hierarchical multicast that makes best-effort attempt to efficiently deliver each message to its destination. IP Multicast [2] can be used where available. The second is a two-phase *anti-entropy* [4] algorithm that operates in a series of asynchronous rounds. During each round, the first phase detects message losses. To this end a gossip-based dissemination phase, where each node sends a digest of its message history, is initiated. The second phase corrects such losses and executes only if needed.

In the present work, we are concerned only with the first phase of the anti-entropy protocol, namely the gossip-based dissemination. A process will continue to gossip the message during a fixed number of rounds after its initial reception. The message is then considered out-of-date and garbage-collected.

Lightweight Probabilistic Broadcast (*lpbcast*) [5] is close to *pbcast*, but, in addition to the information dissemination, it also encloses the membership layer. *Lpbcast* is *lightweight* because it consumes little memory resources and requires no dedicated services for membership management. Aside from the membership scheme, the main differences between *lpbcast* and *pbcast* are (1) that the latter algorithm limits the number of hops as well as (2) repetitions for a given message, and (3) that *lpbcast* melts the two phases of *pbcast* (dissemination of messages and exchange of digests) into a single phase. In *lpcast*, messages received within a round are buffered before being gossiped. If the number of messages exceeds the buffer size, message to be removed are randomly chosen.

2.2 Noise and garbage collection

In a gossip-based algorithm, an efficient garbage collection mechanism should maximize the *utility* of gossiped messages and therefore limit the noise in the system. But these algorithms admit the cases where some messages might be unnecessarily gossiped whereas there are messages that "deserve" more dissemination.

In order to limit the size of the buffers, a garbage collection scheme is involved. To be meaningful, a garbage collection should be closely related to a reliable *stability detection* mechanism. Such mechanisms must detect when a message has been delivered by all (or some pre-defined part) of its recipients, or in other words, has become *stable*, at which point it can be discarded. The decentralized approach we profess here does not allow to detect precisely when a message is delivered by every process. Thus, we are aiming at a *best-effort* garbage collection scheme which allows, in a probabilistic sense, to use only the information received locally, when deciding which message to garbage collect.

3 Age-based garbage collection

3.1 Message age

The idea underlying our garbage collection scheme is to associate with every message some integer, corresponding to the number of rounds the message spent in the system by the current moment. This number is called *age* of the message and is updated in every gossip round. Informally, the age represents the *dissemination degree* of a message in the system. Figure 1 presents a pseudo-code which implements our scheme.

3.2 Gossip processing

There are several phases in our garbage collection scheme (Figure 1(a)).

I. **Broadcast message.** Upon a message broadcast, its age value is initialized to 0. The message is then added to the message history events and if its maximal size is exceeded, "oldest" elements are purged. This is done by the auxiliary function REMOVE_OLDEST_ELEMENTS() (Figure 1(b), see more details on the function in Section 3.3).

II. **Gossip transmission.** This phase is executed periodically (every T seconds) and includes choosing randomly the gossip target and sending the gossip. The ages of stored messages are incremented.

III. **Gossip reception.** When a received gossip is processed, the messages which have not been seen before by process p_i are delivered and stored in the buffer. If a received message has been seen before and the copy of it is stored in the buffer, its age is updated: the maximum of the ages of received and stored messages is taken. As before, REMOVE_OLDEST_ELEMENTS() is invoked to purge the "oldest" items.

Process p_i:

```
upon BROADCAST_MESSAGE(e)
...
e.age← 0
REMOVE_OLDEST_ELEMENTS()

every T seconds
for all e ∈ events do
   e.age←e.age+1
...
SEND_GOSSIP()

upon RECEIVE (gossip)
...
{Update the ages}
for all e ∈ gossip.events do
   if events contains e′ such that
   e′.id = e.id and e′.age < e.age then
      e′.age ← e.age
REMOVE_OLDEST_ELEMENTS()
```

(a) Gossip processing

```
REMOVE_OLDEST_ELEMENTS()
{Out-of-date}
while |events| > |events|_m AND events contains e and e′ such that
(e.source = e′.source and (e.id - e′.id) > LONG_AGO) do
   events ← events/ {e′}
{Age}
while |events| > |events|_m do
   if |events| > |events|_m then
      let e′ ∈ events such that
            e′.age = max_{e∈events}(e.age)
      events ← events/ {e′}
```

(b) Auxiliary function

Figure 1. Age-based buffer processing.

3.3 Age-based purging

When choosing an element to remove from the buffer, two criteria are applied (see Figure 1(b)). A message is purged if:

I. (out-of-date) the message is received a long time ago, with respect to more recent messages from the same broadcast source. This period of time is measured in gossip rounds and compared with LONG_AGO parameter.

II. (oldest) the message has the largest age parameter in the buffer.

4 Practical evaluation

We describe here the results of applying our garbage collection scheme to *pbcast* and *lpbcast*. In short, we show below that our age-buffering scheme allows gains of up to 10% of message stability and to increase the throughput in more than two times.

4.1 Testing environment

The measurements we present here are obtained with 60 processes. The message history buffer size at every process is limited to 30. The fanout, i.e. the number of other processes each process gossips to per round, is fixed to 4. For modelling failures we use process crash ratio equal to 5% and message loss ratio equal to 10%. Where not explicitly mentioned, the broadcast rate is 30, that is, 30 new messages are introduced in the system per gossip round.

4.2 Results

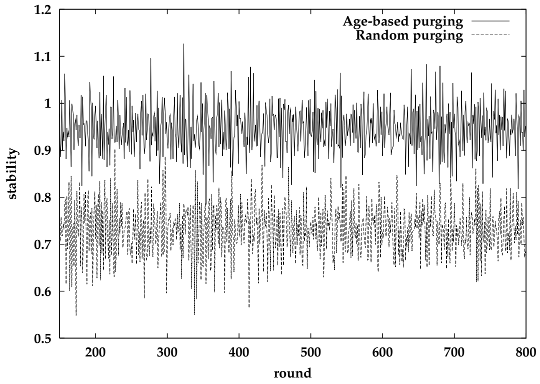
We can summarize the results as follows:

- Figure 2 depicts the message stability levels provided by the broadcast algorithms implementing age-based and randomized buffering schemes. The stability level for age-based buffering is considerably higher for both *pbcast* and *lpbcast*.

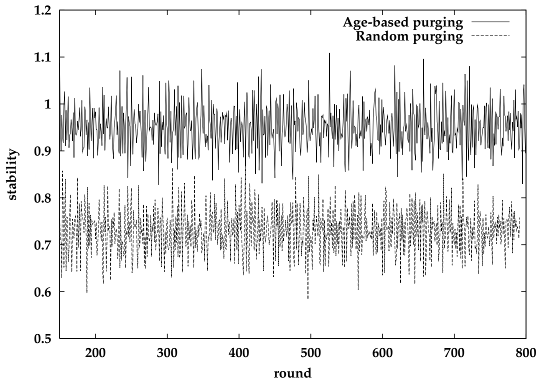
- The throughput estimation presented in Figure 3 implies that age-based buffering enables a broadcast algorithm to stand twice higher broadcast rate providing the same level of message stability.

Despite process crashes and message losses, reliability is not sacrificed by our age-based garbage collection: the average stability level is almost the same for both age-based and randomized buffering schemes. The simulation results clearly show this though it is not presented in the paper, due to space constraint. In fact, our age-based garbage collection scheme does not decrease the *useful* redundancy level of an algorithm. When age-based garbage collection is implemented, it is less probable to gossip a noisy message. This is the source of considerable performance gains shown in Figures 2 and 3.

We should mention that for the practical evaluations, we study here the circumstances that are somewhat beneficial for the age-based buffering scheme: we consider only the gossip-based *anti-entropy* phase [4] (there is no "unreliable" phase as in [1]) and we model the high and regular broadcast rate. We have run a number of experiments in less stressful conditions, in particular, when broadcast rate is small with respect to the buffer sizes. The results are not so impressive, although the advantages of our age-based garbage collection scheme over a random one still hold.

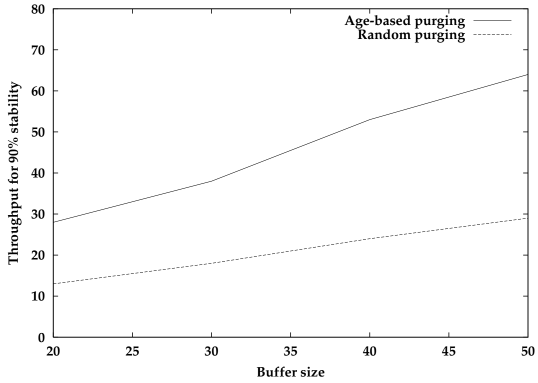


(a) *Pbcast*

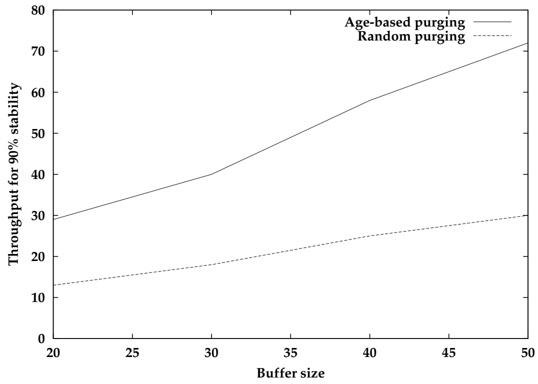


(b) *Lpbcast*

Figure 2. Measurements: stability of non-optimized and optimized (age-based purging) versions of *pbcast* and *lpbcast*.



(a) *Pbcast*



(b) *Lpbcast*

Figure 3. Measurements: throughput of non-optimized and optimized (age-based purging) versions of *pbcast* and *lpbcast* (message stability level 90%).

References

[1] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. In *ACM Transactions on Computer Systems*, volume 17(2), pages 41–88, 1999.

[2] S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, 1991.

[3] S. Deering. Internet multicasting. In *ARPA HPCC 94 Symposium*. Advanced Research Projects Agency Computing Systems Technology Office, Mar. 1994.

[4] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87)*, pages 1–12, Aug. 1987.

[5] P. T. Eugster, R. Guerraoui, S. B. Handurukande, A.-M. Kermarrec, and P. Kouznetsov. Lightweight probabilistic broadcast. In *IEEE International Conference on Dependable Systems and Networks (DSN2001)*, July 2001.

[6] V. Hadzilacos and S. Toueg. *Distributed Systems*, chapter 5: Fault-Tolerant Broadcasts and Related Problems, pages 97–145. Addison-Wesley, 2nd edition, 1993.

[7] K. Lidl, J. Osborne, and J. Malcolm. Drinking from the firehose: Multicast USENET news. In USENIX Association, editor, *Proceedings of the Winter 1994 USENIX Conference*, pages 33–45, jan 1994.