
**Institute of Parallel and Distributed Systems
Distributed Systems Department**



Universität Stuttgart



Reliable Gossip Based Multicast

Seminar Report

**Advanced Topics in Distributed Systems:
Modern Group Communication Mechanisms**

Author: Andrew Mofid Botros Bektor
Supervisor: Dr. Boris Koldehofe
Submission Date: 14th of January, 2009

Abstract

The meaning of Reliable Multicast has changed a lot recently. From “Strong” reliable End-to-End protocols to probabilistic reliable Gossip protocols. This paper discusses some problems about reliable End-to-End multicast protocols. Then talks about Gossip Multicast protocols as an emerging alternative for the traditional reliable protocols. A basic gossip approach is discussed, explaining how it could propose possible solutions to older protocols problems. Also a Group Membership Management protocol that can be used in cooperation with gossip protocols was discussed. Some statistics about how gossip protocols can provide good probabilistic constraints on reliability are presented. The result of the evaluations re-enforce the position of Gossip protocols between other multicast protocols.

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Report Overview	2
2	Gossip Based Multicast and Group Membership Management	2
2.1	Gossip Based Multicast	2
2.2	Group Membership Management	3
2.2.1	Group Membership Management Protocol	4
2.2.2	Uniform Distribution	4
3	Conclusion	5
	Bibliography	6

1 Introduction

1.1 Motivation

General reliable End-to-End multicast protocols, talking about message passing, normally would rely on ACK or NACK approaches. ACK approaches usually experience what is called ACK explosions: ACK explosions happen after a message is sent, all nodes that received the message successfully would reply to the sender with an ACK, with the presence of a large number of members in the multicast group, this amount of ACKs sent to the same destination will block the network, with the problem increasing as we move closer to the destination (the multicast message original sender). NACK approaches were originally developed to avoid ACK explosions, but they are found to only impose another problem which is NACK implosions: NACK approaches, give sequence numbers to each message, when a node fails to receive a message, it will detect a missing sequence number upon the reception of the next message, this node will then send a NACK to the original message sender, which in turn will resend the message. The problem arises when the message fails to leave the source at the first place, then, all members of the multicast group will have a missing sequence number, upon the reception of the next message, again, all the members will send a NACK to the message source, and a problem that is very similar to ACK explosions will arise.

Another problem with reliable End-to-End multicast protocols, is that they require each node to have a full view of the system, which is not realistic in large systems due to memory limitations, and due to synchronization issues.[2] To increase scalability, one would think about some distributed approach to relief the load from the bottlenecks that appear in the End-to-End approaches, with each node having a partial view of the network, and each node participating with only a small part of the message replication required by the multicast. One of these approaches is the *Gossip Based Multicast* (also known as *Probabilistic Multicast*).

Gossip Multicast spreads the information the same way an epidemic spreads (thus also called *Epidemic Multicast/Broadcast*[6]): upon the reception of a new message, each node forwards the new message to a randomly selected set of its neighbors. This approach provides only probabilistic guarantees about the reception of the message at each member of the multicast group. Although the guarantee provided is not a “strong” guarantee, this approach is a really strong competent for large, disperse multicast systems.

The reliability of the *Gossip Based Multicast* is a result of the way it passes around information. In case the first attempt to pass a new message to member X failed, eventually, another member will pass the same information successfully. To avoid message storms, most gossip algorithms allow each members to repeat a specific message only a given number of times before it stops.

1.2 Report Overview

This report is composed of three sections, *Section 1* introduces the concept of *Gossip Based Multicast* as well as the motivation for it explaining some problems with the traditional End-to-End multicast. *Section 2* gives a more detailed overview about gossiping and describes some message passing techniques. It also introduces some Group Membership concepts and a description of an algorithm that implements them. It gives also a little evaluation done on the described Group Membership protocol. *Section 3* is the conclusion, it gives a fast resumé for the material presented in this report.

2 Gossip Based Multicast and Group Membership Management

2.1 Gossip Based Multicast

The word “Reliable Multicast” has become ambiguous. In terms of reliability, approaches providing Reliable Multicast are divided into two groups. One group providing “Strong” reliability, which means, it guarantees message delivery to all active group members. Such approaches might experience some misbehavior under high message traffic due to the heavy load of control messages that have to be passed around between the members to achieve this kind of reliability. The other group only provides best effort reliability, or in other words, probabilistic reliability. *Gossip Based Multicast* falls under the second group. [1]

A basic Gossip Based Multicast relies on interaction between nodes to provide the message replication needed. Assume there is a membership management protocol that provides each node with l random selected uniformly distributed other nodes. The source of the message will request the l random nodes from the membership management system, and will forward his message only to these members. Then, each member, upon the reception of a new message, will again, request l random members from the membership management system and forward the message it just received to them. The number l represents the amount of redundancy in the system, for $l = 1$ there is no redundancy at all. Gossip protocols can use some slightly different approaches to achieve the message forwarding, each to adapt to the needs imposed by the network structure. [5]

A *Gossip Based Multicast* can use one of the following approaches to forward messages:

- **Eager Push:**

Each member will forward any new messages to other members.

- **Pull:**

A member would send a periodic query message to some other members, asking if any new messages were received, a member will update another upon the reception of a query.

- **Lazy Push:**

Each member will forward only the message header to other members, which in turn will request the message if they don't already have it.

The interesting thing about Probabilistic Multicast, is that the probabilistic guarantees they put on message delivery, though they are not deterministic, are really tight guarantees. If there are n nodes in the system, and each node forwards the message to $\log(n) + k$ nodes, then the probability that all nodes in the system get the message is $e^{-e^{-k}}$. This relation is represented in *Figure 1*. [3]

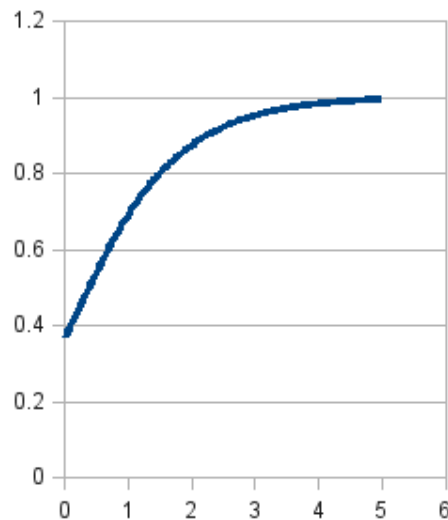


Figure 1: Probability of infection in relation to k

Besides the technique a Gossip approach uses to forward messages, it has also to use some methods to keep information about the group members, and a way for them to subscribe, unsubscribe, and to detect isolation and recover from it. The methods used for group membership has also to reflect the changes as soon as possible, by means of spreading the new information to the other members of the group. [6]

2.2 Group Membership Management

In spite of the very good scalability properties of the Gossip Based message passing protocols, they rely on some other poorly scalable protocols to manage group membership: they require each node, or the group membership management system, to know about each other node in the system, which doesn't scale really well. To improve on these membership protocols, we need to think again about finding some distributed

approach to accomplish the same functionality, managing the group membership. [3]
[4]

2.2.1 Group Membership Management Protocol

Subscription

- **Subscription.** A new node can join the system by sending a subscription request to any member of the multicast group.
- **New Subscription.** Upon the reception of a new subscription, a node informs all other nodes in its partial view about this new subscription. It also makes a number of additional copies of the subscription and sends them to a randomly selected set of nodes from its partial view.
- **Forwarded Subscription.** When a node receives a subscription notification, it either stores it in its partial view, or it forwards it to a random node from its partial view. This ensures that the forwarded subscriptions will not get lost, they keep being forwarded till a node stores them.
- **Keeping a Subscription.** Each node keeps two lists, a *PartialView*, containing the nodes it sends messages to, and an *InView* containing the nodes that send messages to it. If a node i decides to store another node's j subscription, it stores it to its *PartialView* and it sends a message to j telling it to add i to its *InView* list.

[3]

Unsubscription To unsubscribe, node X will send a message to the first $n - c$ (where n is the length of the list) nodes in its *InView list*, asking them to replace its own id in their *PartialView* list with the first $n - c$ ids in node X's *PartialView* list, thus, passing a big part of its *PartialView* to its message sources, and discarding the rest of the list. This will ensure the average list sizes in all nodes in the system will decrease with a node leaving the system. In case of duplicates, or in case of a node storing its own id, these items are deleted immediately from the *PartialView* list.[3]

Recovery from Isolation Isolation happens to a node, when all nodes having its id in their *PartialView* list fail. To detect that, a heartbeat approach was proposed. Each node sends a periodical, alive message to all nodes in its *PartialView*. If a node doesn't receive any of these alive messages in a long while, it knows it became isolated. The isolated node can then re-subscribe with a random node from its *PartialView* list.[3]

2.2.2 Uniform Distribution

The real challenge facing a group membership management protocol is that the random nodes yielded should be randomly distributed over the set of all nodes in the system. Although this is the basic requirement for a group membership management system, most of the evaluations done on group membership management protocols use the

impact of node failure on message delivery to evaluate the efficiency of the system. This is done by imposing a percentage of node failure for the system, and measuring the portion of nodes infected by a message. In [3] the membership protocol described above, with some modifications, yields around 93% with 70% node failure rate.

Another criteria to evaluate the reliability of the system, is the effect of unsubscriptions on the portion of nodes infected. With half the nodes unsubscribing, in a 50,000 nodes group, the above system yields around 96% infected nodes. [3]

3 Conclusion

This paper discusses some of the problems with traditional End-to-End reliable multicast protocols, like ACK/NACK explosions/implosions and membership management. Gossip based multicast protocols arise as a very competent alternative to the traditional End-to-End multicast, especially in vast networks and applications that don't require "Strong" reliability constraints. Gossip based approaches rely on spreading the messages the way an epidemic spreads, which was found to yield a very high probability of infection between the system members. With a such completely decentralized approach, ACK/NACK explosion/implosion problems are avoided.

As a solution to the membership management problem, a basic distributed membership management protocol was described. The presented protocol provides methods to handle the impact of unsubscriptions on the system as well as methods to detect and recover from node isolation.

References

- [1] Kenneth P. Birman, Mark Hayden, Ozgur Ozkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. Bimodal multicast. *ACM Trans. Comput. Syst.*, 17(2):41–88, 1999.
- [2] P. Th. Eugster, R. Guerraoui, S. B. Handurukande, and P. Kouznetsov. Lightweight probabilistic broadcast. In *ACM Trans. Comput. Syst.*, pages 443–452, 2001.
- [3] A. Ganesh, A. Kermarrec, and L. Massoulié. Peer-to-peer membership management for gossip-based protocols, 2003.
- [4] Meng jang Lin and Keith Marzullo. Directional gossip: Gossip in a wide area network. In *In European Dependable Computing Conference*, pages 364–379, 1999.
- [5] A. Kermarrec, L. Massoulié, and A. Ganesh. Probabilistic Reliable Dissemination in Large-Scale Systems. *IEEE Trans. Parallel Distrib. Syst.*, 14(3):248–258, 2003.
- [6] Joao Leitaó, Jose Pereira, and Luis Rodrigues. Epidemic broadcast trees. In *SRDS '07: Proceedings of the 26th IEEE International Symposium on Reliable Distributed Systems*, pages 301–310, Washington, DC, USA, 2007. IEEE Computer Society.