

Advanced Topics in Distributed Systems: Reliable Gossip Based Multicast

Andrew M. B. Boker

Institute of Parallel and Distributed Systems

9th of February, 2009

Reliability

Reliable Multicast

- Strong Reliable Multicast

- Gossip Based Multicast

Gossip Based Multicast

- Basic Algorithm

- Message Passing

- Probabilistic Reliability

- Redundancy

Membership Management

- Basic Algorithm

- Join

- Leave

- Recovery from Isolation

- Performance

Conclusion

Reliability

- ▶ **Definition:** (engineering) The probability that a component part, equipment, or system will satisfactorily perform its intended function under given circumstances, such as environmental conditions, limitations as to operating time, and frequency and thoroughness of maintenance for a specified period of time. [from answers.com]
- ▶ **Strong Reliability:** 100% (if the message leaves the source)
- ▶ **Probabilistic Reliability:** very close to 100%

Reliable Multicast

- ▶ **Strong Reliability Category:**
Reliability on a Per-Recipient basis.
- ▶ **Probabilistic Reliability Category:**
Reliability on a Per-Group basis.

Strong Reliable Multicast

Good:

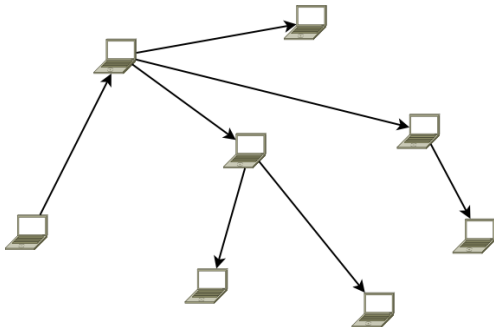
- ▶ Assures delivery to all functional nodes

Bad:

- ▶ Large Overhead
- ▶ Ack Explosions (next slides)
- ▶ NAck Implosions (next slides)

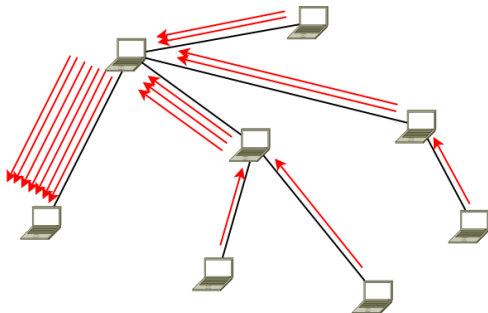
Strong Reliable Multicast: Ack Explosions 1

- ▶ Source sends the message
- ▶ Message propagates along the Multicast tree



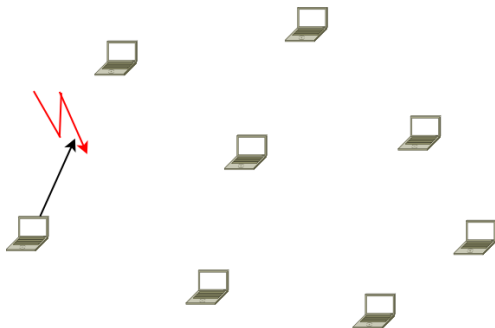
Strong Reliable Multicast: Ack Explosions 2

- ▶ Receivers send Ack to source
- ▶ Acks floods the network



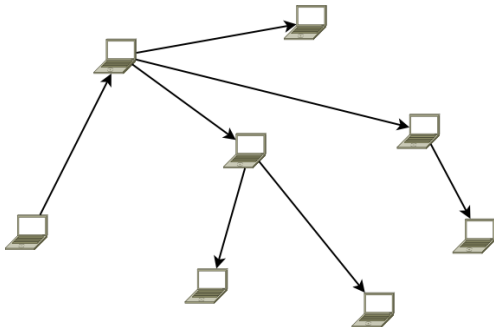
Strong Reliable Multicast: NAck Implosions 1

- ▶ First proposed as solution for Ack Explosions
- ▶ Source sends the message
- ▶ The packet is lost before it reaches another member



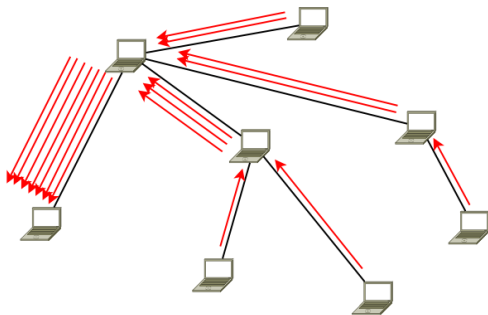
Strong Reliable Multicast: NAck Implosions 2

- ▶ Nobody notices the loss
- ▶ The next message is send successfully



Strong Reliable Multicast: NAck Implosions 3

- ▶ Members notice a missing sequence number
- ▶ They all send NAck to the source
- ▶ NAck floods the network



Epidemic

- ▶ **Epidemic Disease:** any infectious disease that develops and spreads rapidly to many people. [from thefreedictionary.com]
- ▶ Like the FLU. Who doesn't get a flu at least once a year???
- ▶ What about a Multicast like the flu?



Gossip Based Multicast

- ▶ Gossip Based Multicast follows this same scheme
- ▶ Let's see How it works

Gossip Based Multicast

- ▶ Inspired by Epidemics
- ▶ Each node infects some other nodes
- ▶ The message spreads between nodes like an epidemic
- ▶ Eventually, everybody will get infected



Gossip Based Multicast: Basic Algorithm

When you Receive a message:

- ▶ Select “ a ” uniformly distributed members
- ▶ Send the message to them
- ▶ Do that only m times

How to send the message?

What is a good value for “ a ”?

How to select “ a ” uniformly distributed members?

Gossip Based Multicast: Message Passing

- ▶ **Eager Push**

Send any message right away

- ▶ **Pull**

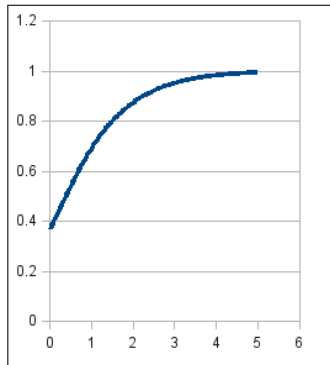
1 asks 2 if there are new messages, 2 sends the new messages to 1 if any

- ▶ **Lazy Push**

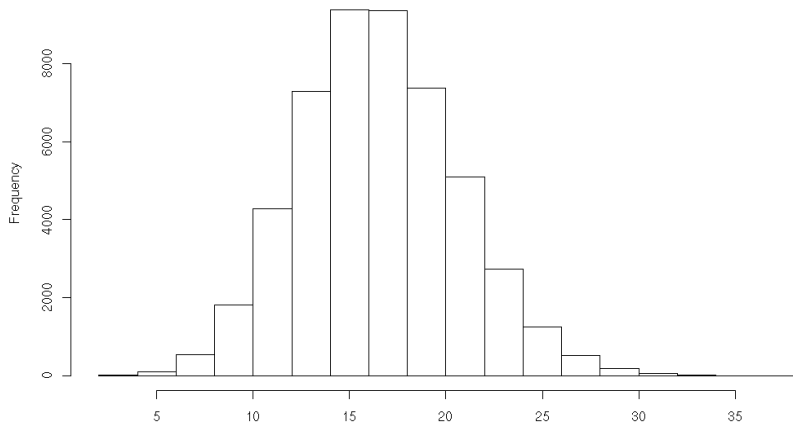
1 sends the message header to 2, 2 requests the message if he needs it

Gossip Based Multicast: Probabilistic Reliability

- ▶ No hard guarantees
- ▶ Only probabilistic guarantees
- ▶ But, really tight ones !!!
- ▶ Derived from epidemic mathematics
- ▶ $Prob = e^{-e^{-k}}$ if each node infects $a = \log(N) + k$ nodes where N is the total number of nodes in the system
- ▶ if $N = 100$ then sending to 15 members will assure almost 100% probability



Gossip Based Multicast: Redundance



Gossip Based Multicast: Managing the Group

Group Membership:

- ▶ Keeps track of the members of the group
- ▶ Provides a way to Join and Leave the group
- ▶ Provides uniformly distributed sub-sets of group members

Membership Management

Old approaches: Centralized

- ▶ Central node has database
- ▶ Join, Leave via central node
- ▶ Central node sends a subset of members upon request

Distributed approach: Decentralized

- ▶ No centralized node
- ▶ Each member keeps track of some other members
- ▶ Join, Leave via any node

Membership Management: Basic Algorithm

SCAMP: SCALable Membership Protocol

Each node has 2 lists:

- ▶ *PartialView*: Nodes it sends messages to
- ▶ *InView*: Nodes whom it receives messages from

Membership Management: Join

The new Node:

- ▶ Sends Join request to any group member

Member nodes:

- ▶ Forward the subscription to all nodes in *PartialView*
- ▶ FW c copies to nodes from *PartialView*
- ▶ When you get a FW subscription, either add it to *PartialView* or FW it to another node
- ▶ If you keep it, then inform the joining node to add you in its *InView*

Membership Management: Leave

Node X is leaving:

- ▶ Send $n - c$ items from the *PartialView* list to $n - c$ nodes from the *InView* list
- ▶ Each node will replace X's ID with the ID received in the *PartialView* list
- ▶ Remove any duplicates or your own ID from the *PartialView*

Membership Management: Recovery from Isolation

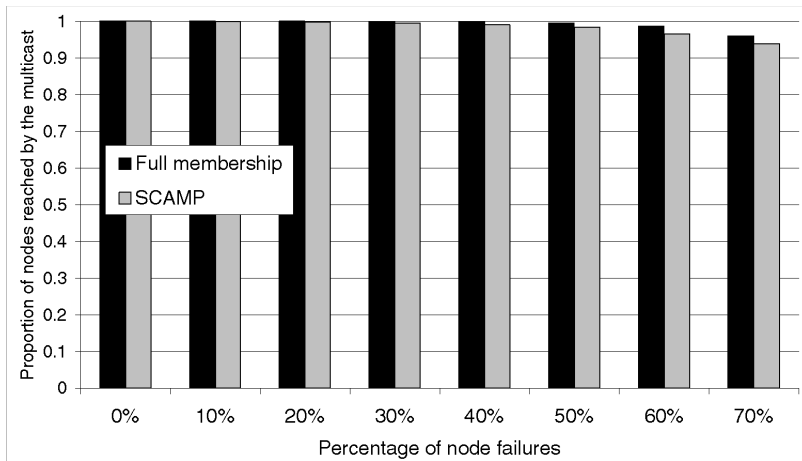
- ▶ Send alive message to nodes in *PartialView* periodically
- ▶ If you don't get any alive messages, you are isolated
- ▶ Re-join the group with any node from the *PartialView*

All Membership information could be piggybacked on normal messages

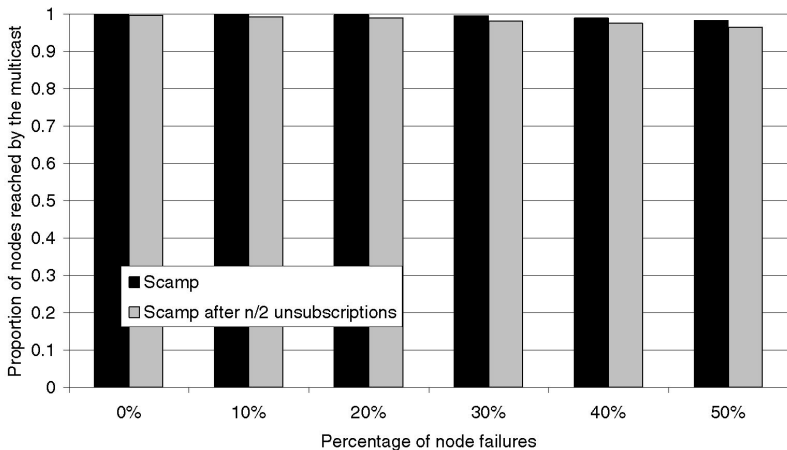
Membership Management: Performance

- ▶ Main Goal: Uniformly Distributed *PartialView* lists
- ▶ *PartialView*'s list size has to be close to $\log(N)$
- ▶ Can measure performance by the impact of failure on reliability
- ▶ Or by the impact of unsubscriptions on reliability

Membership Management: Performance 2



Membership Management: Performance 3



Conclusion

Gossip Based Multicast: Pros:

- ▶ Distributed
- ▶ High Reliability
- ▶ Failure tolerance

Gossip Based Multicast: Cons:

- ▶ Huge Redundancy !!!




How could we achieve that?

- ▶ Epidemic Spreading
- ▶ Distributed Membership Management
- ▶ Piggyback Membership information on messages




Any Questions?



References

-  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.
-  P. Th. Eugster, R. Guerraoui, S. B. Handurukande, and P. Kouznetsov.
Lightweight probabilistic broadcast.
In *ACM Trans. Comput. Syst.*, pages 443–452, 2001.
-  A. Ganesh, A. Kermarrec, and L. Massoulié.
Peer-to-peer membership management for gossip-based protocols, 2003.

References

-  Meng jang Lin and Keith Marzullo.
Directional gossip: Gossip in a wide area network.
In In European Dependable Computing Conference, pages 364–379, 1999.
-  A. Kermarrec, L. Massoulié, and A. Ganesh.
Probabilistic Reliable Dissemination in Large-Scale Systems.
IEEE Trans. Parallel Distrib. Syst., 14(3):248–258, 2003.
-  Joao Leitao, 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.