CSE 486/586 Distributed Systems
Gossiping

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Revisiting Multicast

Node with a piece of information
to be communicated to everyone

Distributed Group of
"Nodes"=
Processes at Internet-
based hosts

Fault-Tolerance and Scalability

Multicast sender

Multicast Protocol

- Nodes may crash
- Packets may be dropped
- Possibly 1000's of nodes

B-Multicast

- Simplest implementation
- Problems?

UDP/TCP packets

R-Multicast

- Stronger guarantees
- Overhead is quadratic in N

UDP/TCP packets

Any Other?

- E.g., tree-based multicast
- e.g., IPmulticast, SRM, RMT, TRAM, TMTP
- Tree setup and maintenance
- Problems?
Another Approach

Multicast sender

Periodically, transmit to $b$ random targets

Gossip messages (UDP)

Other nodes do same after receiving multicast

Gossip messages (UDP)

"Gossip" (or "Epidemic") Multicast

Protocol rounds (local clock)

$b$ random targets per round

Gossip Message (UDP)

Uninfected

Infected

CSE 486/586 Administrivia

- PA2-B
  - Please start now!
  - This is when some people seriously consider code-copying.
- Re-grading this week
- Midterm on 3/11
  - During class
Properties
• Lightweight
• Quick spread
• Highly fault-tolerant
• Analysis from old mathematical branch of Epidemiology [Bailey 75]
• Parameters c,b:
  – c for determining rounds: \(c \cdot \log(n)\), b: # of nodes to contact
  – Can be small numbers independent of n, e.g., c=2; b=2;
• Within \(c \cdot \log(n)\) rounds, [low latency]
  – all but \(\frac{1}{N^{c/b}}\) of nodes receive the multicast [reliability]
  – each node has transmitted no more than \(c \cdot b \cdot \log(n)\) gossip messages [lightweight]

Fault-Tolerance
• With failures, is it possible that the epidemic might die out quickly?
• Possible, but improbable:
  – Once a few nodes are infected, with high probability, the epidemic will not die out
  – So the analysis we saw in the previous slides is actually behavior with high probability
    [Galey and Dani 98]
• The same applicable to:
  – Rumors
  – Infectious diseases
  – An Internet worm
• Some implementations
  – Amazon Web Services EC2/S3 (rumored)
  – Usenet NNTP (Network News Transport Protocol)

Gossip-Style Failure Detection
• Processes periodically gossip their membership list
• On receipt, the local membership list is updated
  • Current time : 70 at process 2 (asynchronous clocks)

Fault-Tolerance
• Packet loss
  – 50% packet loss: analyze with b replaced with b/2
  – To achieve same reliability as 0% packet loss, takes twice as many rounds
• Node failure
  – 50% of nodes fail: analyze with n replaced with n/2 and b replaced with b/2
  – Same as above

Using Gossip for Failure Detection: Gossip-style Heartbeating
• Each process sends out heartbeats to every other process
• Con: Slow process/link causes false positives

Gossip-Style Failure Detection
• If the heartbeat has not increased for more than \(T_{\text{fail}}\) seconds (according to local time), the member is considered failed
• But don’t delete it right away
• Wait another \(T_{\text{cleanup}}\) seconds, then delete the member from the list
Summary

• Gossiping
  – One strategy for lazy replication
  – High-level of fault-tolerance & quick spread
• Another use case for gossiping
  – Failure detection

Acknowledgements

• These slides contain material developed and copyrighted by Indranil Gupta (UIUC).