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

UDP/TCP packets

- Simplest implementation:
- Problems?

R-Multicast

UDP/TCP packets

- Stronger guarantees
- Overhead is quadratic in N

Any Other?

- E.g., tree-based multicast
- e.g., IP multicast, SRM, RMTP, TRAM, TMTP
- Tree setup and maintenance
- Problems?

UDP/TCP packets
Another Approach

Multicast sender

CSE 486/586

Another Approach

Periodically, transmit to \( b \) random targets

Gossip messages (UDP)

CSE 486/586

Another Approach

Other nodes do same after receiving multicast

Gossip messages (UDP)

CSE 486/586

Another Approach

“Gossip” (or “Epidemic”) Multicast

Protocol \( r \) rounds (local clock)

\( b \) random targets per round

Gossip Message (UDP)

Infected

Uninfected

CSE 486/586

CSE 486/586 Administrivia

- PA2-B is due in ~2 weeks.
  - Please start now!
  - This is when some people seriously consider code-copying.
- PA1 grades are posted.
- PA2-A grading is in progress.
- Undergrads: we will have recitations this week.
Properties

- Lightweight
- Quick spread
- Highly fault-tolerant
- Analysis from old mathematical branch of Epidemiology [Bailey 75]

Parameters c,b:
- c for determining rounds: (c*\log(n)), b: \# of nodes to contact
- Can be small numbers independent of n, e.g., c=2; b=2:
- Within c*\log(n) rounds, [low latency]
  - all but \frac{1}{N} \text{ of nodes receive the multicast [reliability]}
  - each node has transmitted no more than c*b*\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

- 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

Using Gossip for Failure Detection: Gossip-style Heartbeating
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).