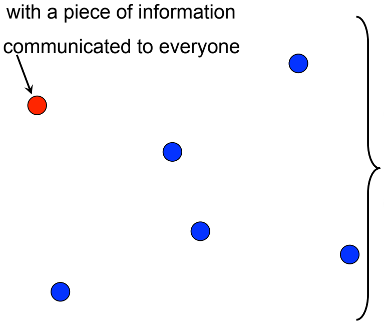
CSE 486/586 Distributed Systems
Gossiping

Steve Ko
Computer Sciences and Engineering
University at Buffalo

Revisiting Multicast

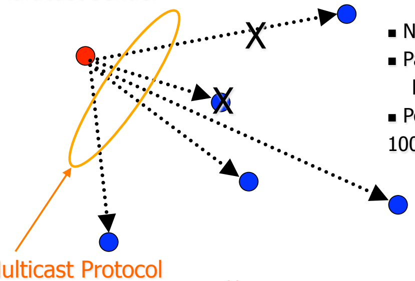
Node with a piece of information
to be communicated to everyone



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

Fault-Tolerance and Scalability

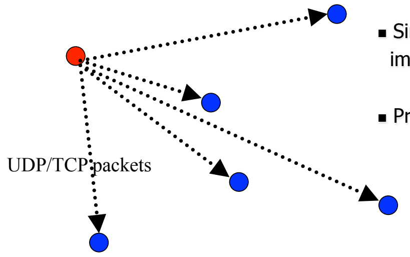
Multicast sender



Multicast Protocol

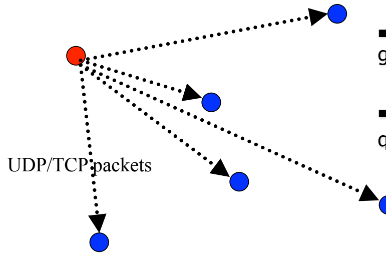
- Nodes may crash
- Packets may be dropped
- Possibly 1000's of nodes

B-Multicast



- Simplest implementation
- Problems?

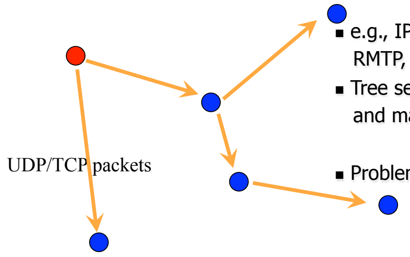
R-Multicast



- Stronger guarantees
- Overhead is quadratic in N

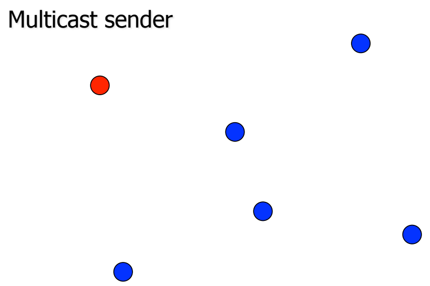
Any Other?

- E.g., tree-based multicast



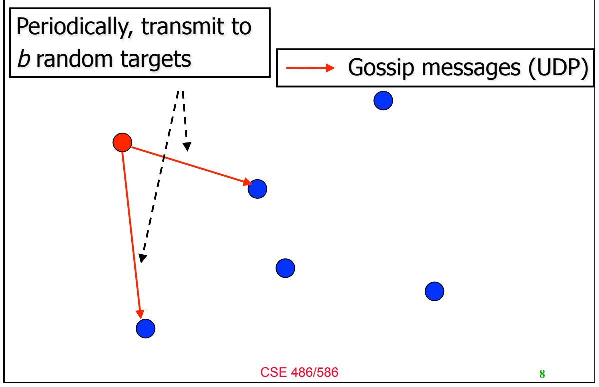
- e.g., IPmulticast, SRM RMTP, TRAM,TMTP
- Tree setup and maintenance
- Problems?

Another Approach



Multicast sender

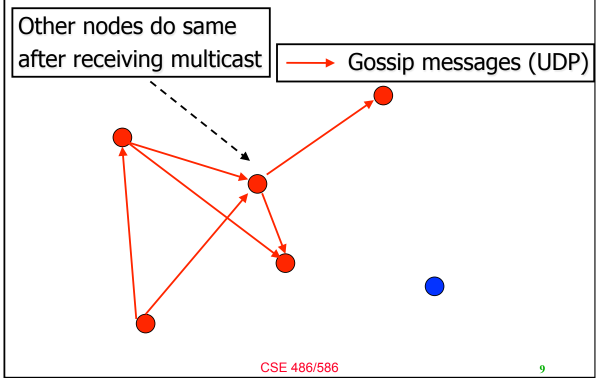
Another Approach



Periodically, transmit to *b* random targets

Gossip messages (UDP)

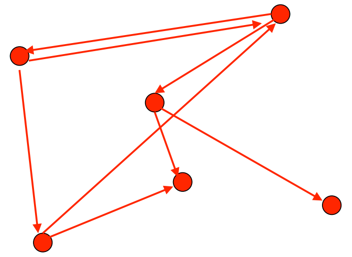
Another Approach



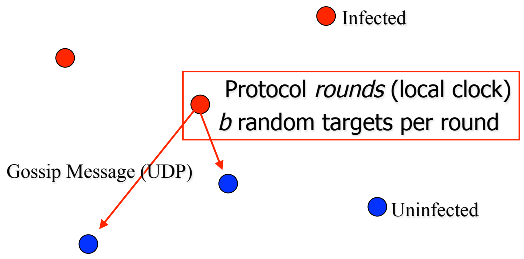
Other nodes do same after receiving multicast

Gossip messages (UDP)

Another Approach



"Gossip" (or "Epidemic") Multicast



Infected

Protocol *rounds* (local clock)
b random targets per round

Gossip Message (UDP)

Uninfected

CSE 486/586 Administrivia

- PA2-B is due in ~2 weeks.
- PA1 grades are posted.
- PA2-A grading is in progress.

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 $\frac{1}{n^{cb-2}}$ of nodes receive the multicast [reliability]
  - each node has transmitted no more than *c*b*log(n)* gossip messages [lightweight]

Fault-Tolerance

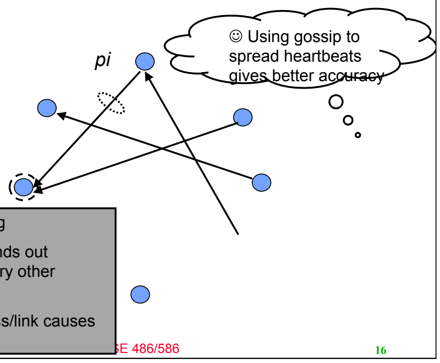
- Packet loss
  - 50% packet loss: analyze with *b* replaced with *b/2*
  - To achieve same reliability as 0% packet loss, takes twice as many rounds
- Node failure
  - 50% of nodes fail: analyze with *n* replaced with *n/2* and *b* replaced with *b/2*
  - Same as above

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)

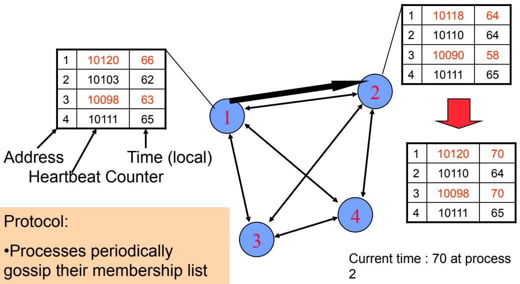
Using Gossip for Failure Detection: Gossip-style Heartbeating



All-to-all heartbeating

- Each process sends out heartbeats to every other process
- Con: Slow process/link causes false positives

Gossip-Style Failure Detection



1	10120	66
2	10103	62
3	10098	63
4	10111	65

1	10118	64
2	10110	64
3	10090	58
4	10111	65

1	10120	70
2	10110	64
3	10098	70
4	10111	65

Protocol:

•Processes periodically gossip their membership list

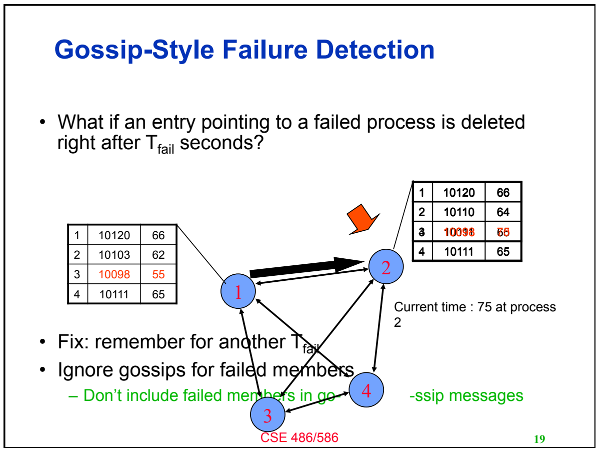
•On receipt, the local membership list is updated

Current time : 70 at process 2

(asynchronous clocks)

Gossip-Style Failure Detection

- If the heartbeat has not increased for more than T_{fail} seconds (according to local time), the member is considered failed
- But don't delete it right away
- Wait another $T_{cleanup}$ seconds, then delete the member from the list



Gossip-Style Failure Detection

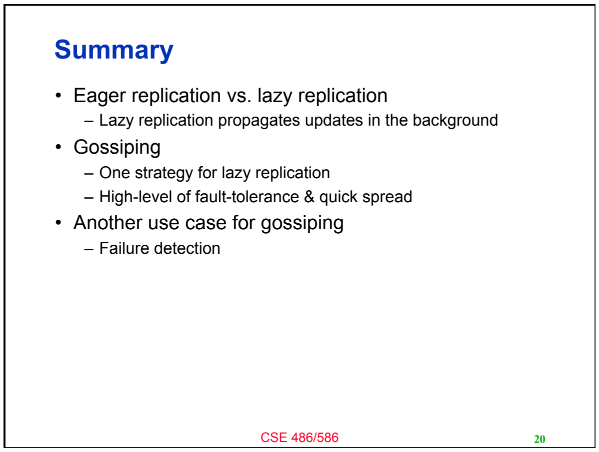
- What if an entry pointing to a failed process is deleted right after T_{fail} seconds?

1	10120	66
2	10103	62
3	10098	55
4	10111	65

1	10120	66
2	10110	64
3	10098	75
4	10111	65

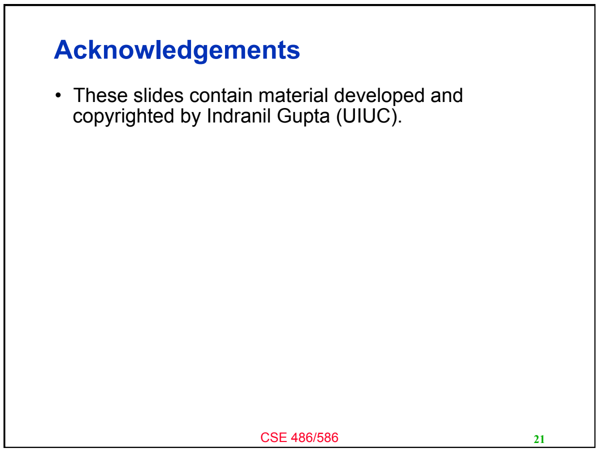
Current time : 75 at process 2

- Fix: remember for another T_{fail}
- Ignore gossips for failed members
  - Don't include failed members in go- -ssip messages



Summary

- Eager replication vs. lazy replication
  - Lazy replication propagates updates in the background
- 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).