CLOUD COMPUTING CONCEPTS

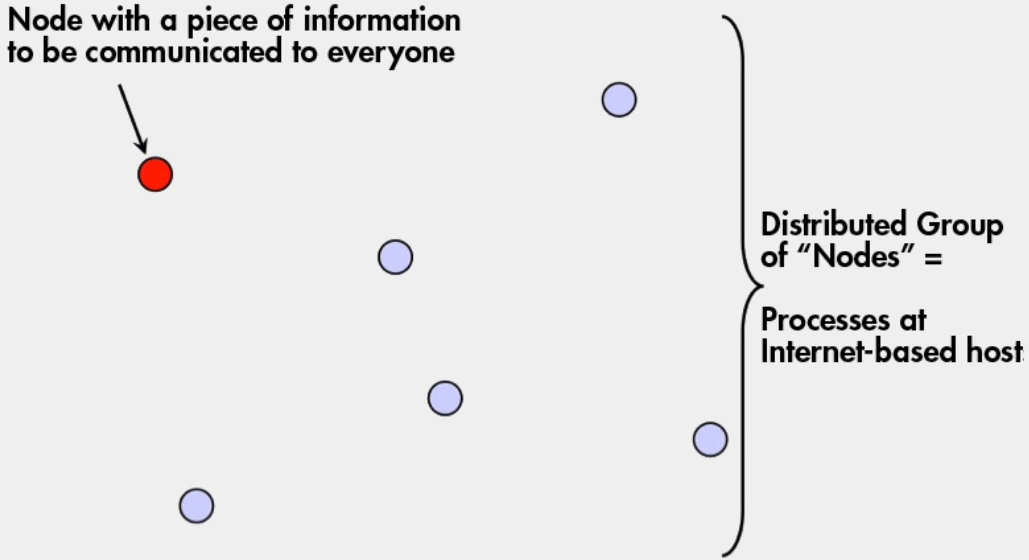
with Indranil Gupta (Indy)

GOSSIP

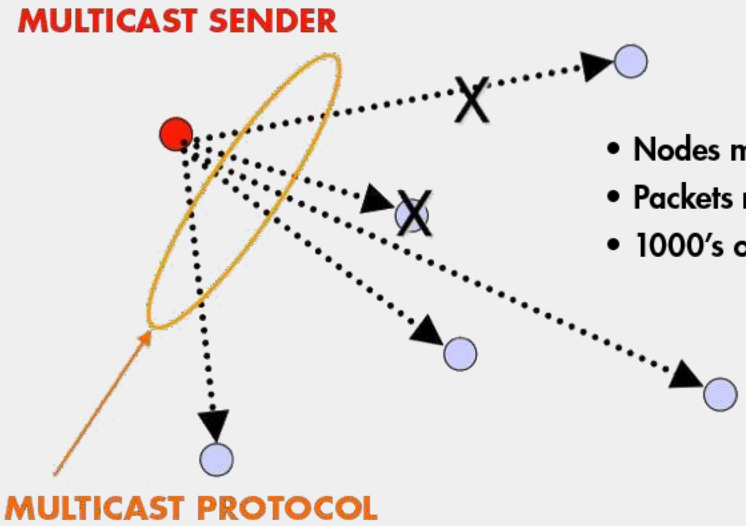
Lecture A

MULTICAST PROBLEM

Multicast

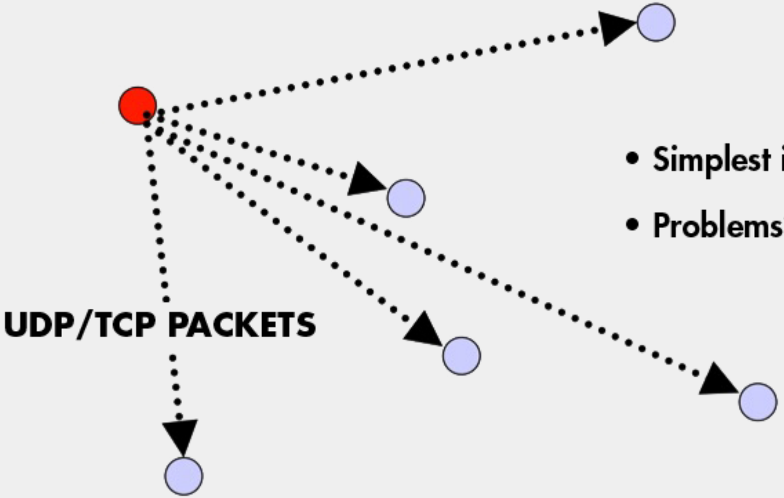

Node with a piece of information to be communicated to everyone
Distributed Group of "Nodes" =
Processes at Internet-based host

Fault-tolerance and Scalability



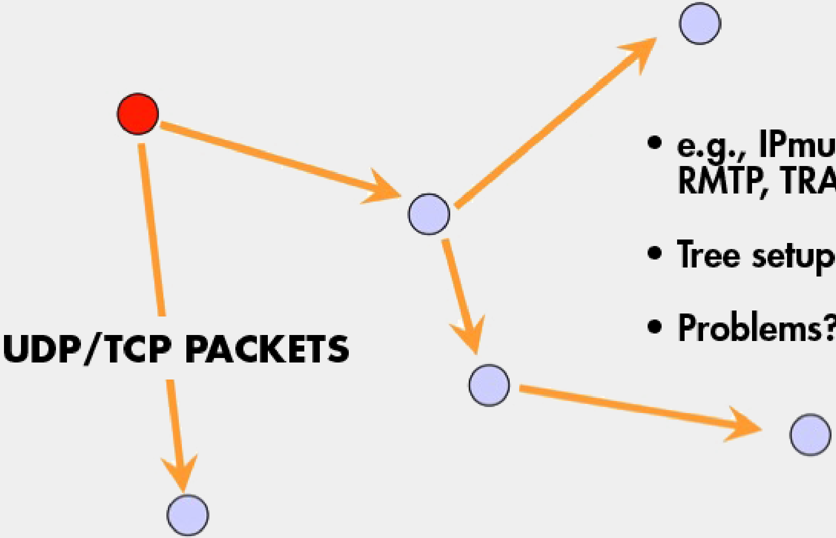
- Nodes may crash
- Packets may be dropped
- 1000's of nodes

Centralized



- Simplest implementation
- Problems?

Tree-Based



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

Tree-based Multicast Protocols

- Build a spanning tree among the processes of the multicast group
- Use spanning tree to disseminate multicasts
- Use either acknowledgments (ACKs) or negative acknowledgements (NAKs) to repair multicasts not received
- SRM (Scalable Reliable Multicast)
  - Uses NAKs
  - But adds random delays, and uses exponential backoff to avoid NAK storms
- RMTP (Reliable Multicast Transport Protocol)
  - Uses ACKs
  - But ACKs only sent to designated receivers, which then re-transmit missing multicasts
- These protocols still cause an O(N) ACK/NAK overhead

CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

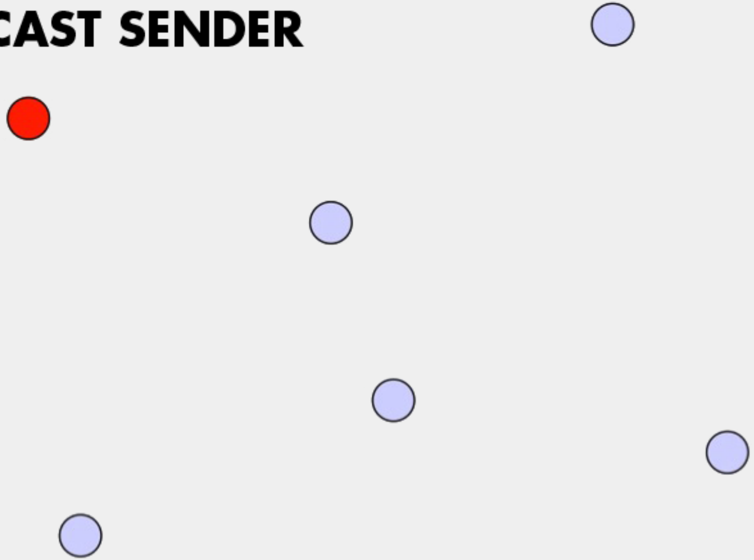
GOSSIP

Lecture B

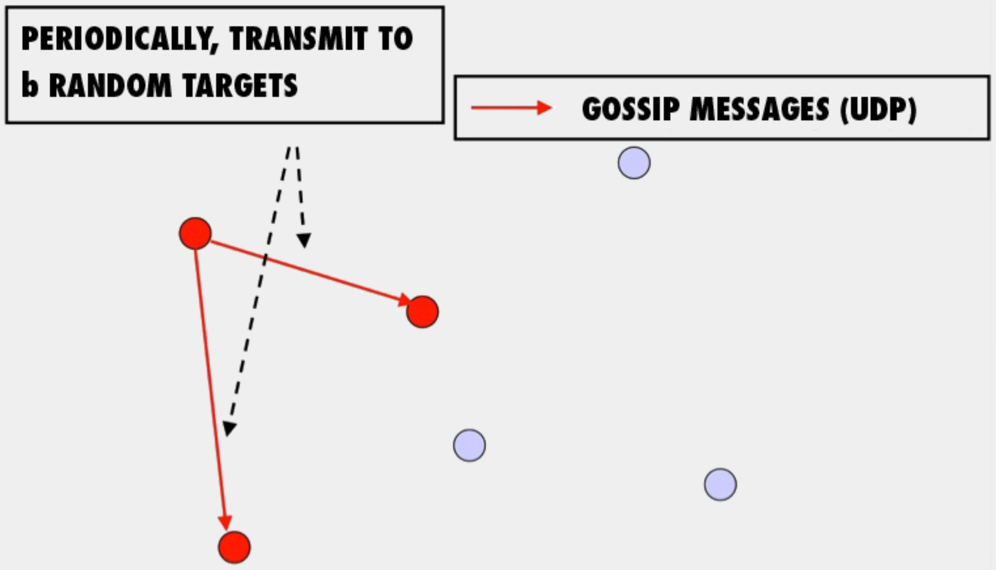
THE GOSSIP PROTOCOL

A Third Approach

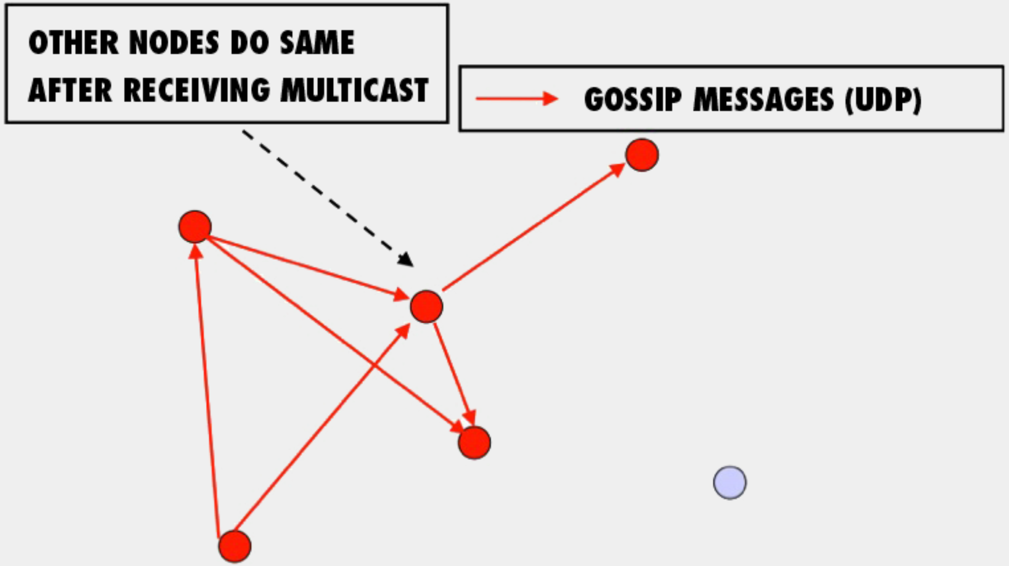
MULTICAST SENDER



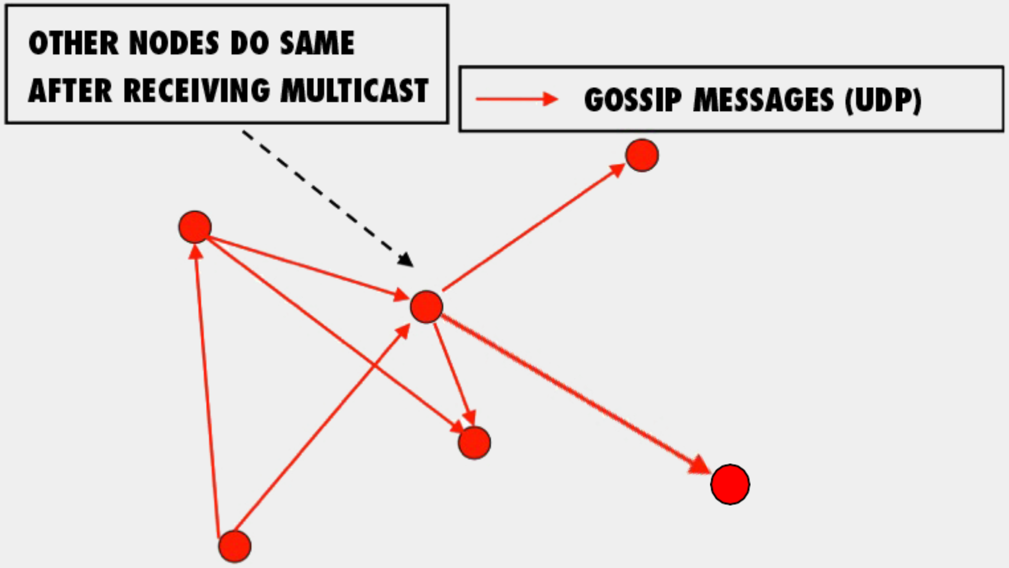
A Third Approach



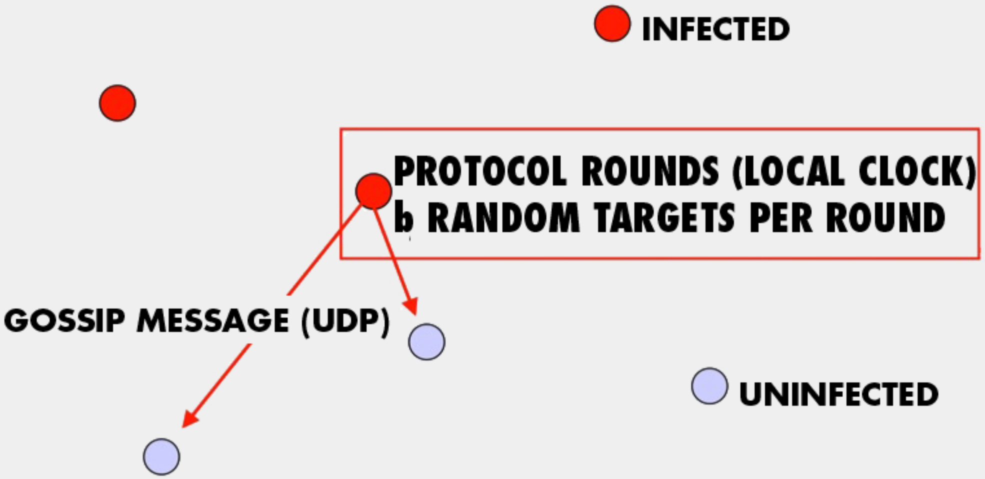
A Third Approach



A Third Approach



"Epidemic" Multicast (or "Gossip")



Push vs. Pull

- So that was "Push" gossip
  - Once you have a multicast message, you start gossiping about it
  - Multiple messages? Gossip a random subset of them, or recently-received ones, or higher priority ones
- There's also "Pull" gossip
  - Periodically poll a few randomly selected processes for new multicast messages that you haven't received
  - Get those messages
- Hybrid variant: Push-Pull
  - As the name suggests

CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

GOSSIP

Lecture C

GOSSIP ANALYSIS

Properties

Claim that the simple Push protocol

- Is lightweight in large groups
- Spreads a multicast quickly
- Is highly fault-tolerant

Analysis

From old mathematical branch of *Epidemiology* [Bailey 75]

- Population of ($n+1$) individuals mixing homogeneously
- Contact rate between any individual pair is β

0<B<1 (Probability of Contact)

- At any time, each individual is either uninfected (numbering x) or infected (numbering y)
- Then, $x_0 = n$, $y_0 = 1$

  and at all times $x + y = n + 1$
- Infected–uninfected contact turns latter infected, and it stays infected

Analysis (contd.)

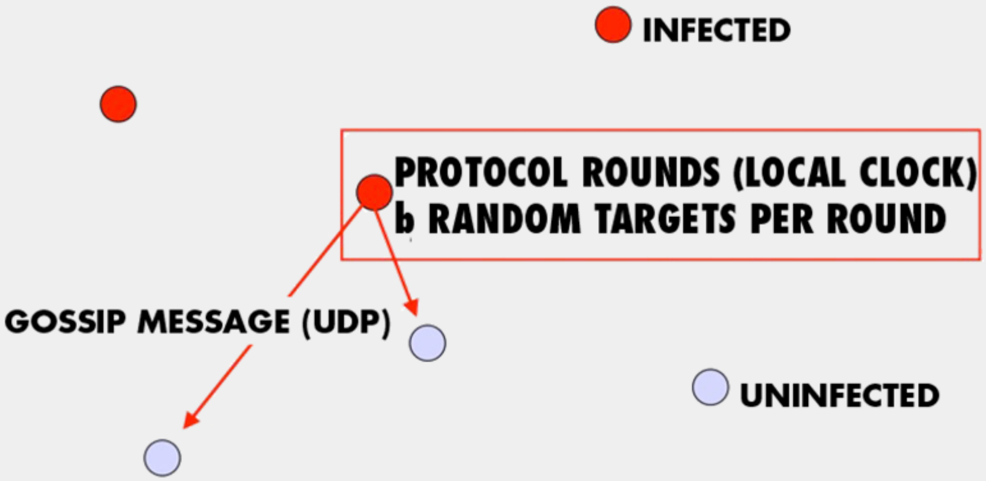
- Continuous time process
- Then

$$\frac{dx}{dt} = -\beta xy \qquad \text{(why?)}$$

with solution:

$$x = \frac{n(n+1)}{n + e^{\beta (n+1)t}}, \; y = \frac{(n+1)}{1 + ne^{-\beta (n+1)t}}$$

Epidemic Multicast



Epidemic Multicast Analysis

$$\beta = \frac{b}{n}$$

(why?)

b of n possible targets gets contacted. Total n+1.

Substituting, at time *t=clog(n)*, the number of infected is

O(logn) rounds have happened

$$y \approx (n+1) - \frac{1}{n^{cb-2}}$$

- In all forms of gossip, it takes O(log(N)) rounds before about N/2 gets the gossip
  - Why? Because that’s the fastest you can spread a message – a spanning tree with fanout (degree) of constant degree has O(log(N)) total nodes

Answer – Push Analysis (contd.)

Using: $\beta = \dfrac{b}{n}$

$$y = \frac{(n+1)}{1+ne^{-\beta\,(n+1)t}}$$

Substituting, at time $t = clog(n)$

$$y = \frac{n+1}{1+ne^{-\frac{b}{n}(n+1)c\log(n)}} \approx \frac{n+1}{1+\dfrac{1}{n^{cb-1}}}$$

$$\approx (n+1)\left(1-\frac{1}{n^{cb-1}}\right)$$

$$\approx (n+1)-\frac{1}{n^{cb-2}}$$

Analysis (contd.)

$$y \approx (n+1) - \frac{1}{n^{cb-2}}$$

- Set *c*,*b* to be small numbers independent of *n*
- Within *clog*(*n*) rounds, [**low latency**]
  - all but $\frac{1}{n^{cb-2}}$ number of nodes receive the multicast [**reliability**]
  - each node has transmitted no more than *cblog*(*n*)gossip messages [**lightweight**]

y is nearly equal to (n+1) even after 2 rounds with b=2 and c=2

x+y=(n+1)

y=((n+1)-x)

Why is log(N) low?

- Log(N) is not constant in theory
- But pragmatically, it is a very slowly growing number
- Base 2
  - Log(1000) ~ 10
  - Log(1M) ~ 20
  - Log(1B) ~ 30
  - Log(all IPv4 address) = 32

Fault-tolerance

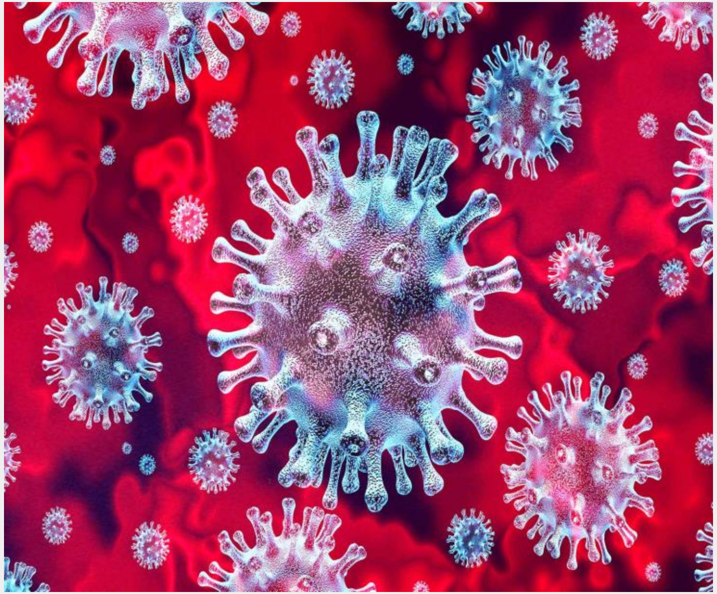
- Packet loss
  - 50% packet loss: analyze with *b* replaced with *b/2*
  - To achieve same reliability as 0% packet loss, takes twice as many rounds
- Node failure
  - 50% of nodes fail: analyze with *n* replaced with *n/2* and *b* replaced with *b/2*
  - Same as above

$$y \approx (n+1) - \frac{1}{n^{cb-2}}$$

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]
- Think: Why do rumors spread so fast? Why do infectious diseases cascade quickly into epidemics? Why does a virus or worm spread rapidly?



Pull Gossip: Analysis

- In all forms of gossip, it takes O(log(N)) rounds before about N/2 gets the gossip
  - Why? Because that's the fastest you can spread a message – a spanning tree with fanout (degree) of constant degree has O(log(N)) total nodes
- Pull gossip is faster than push gossip
- After the *i*th, round let p_i be the fraction of non-infected processes. Then (k=number of gossip pulls per round per process)

$$p_{i+1} = (p_i)^{k+1}$$

k= number of Gossip targets equal to b
Being fractions multiplication leads to even smaller value

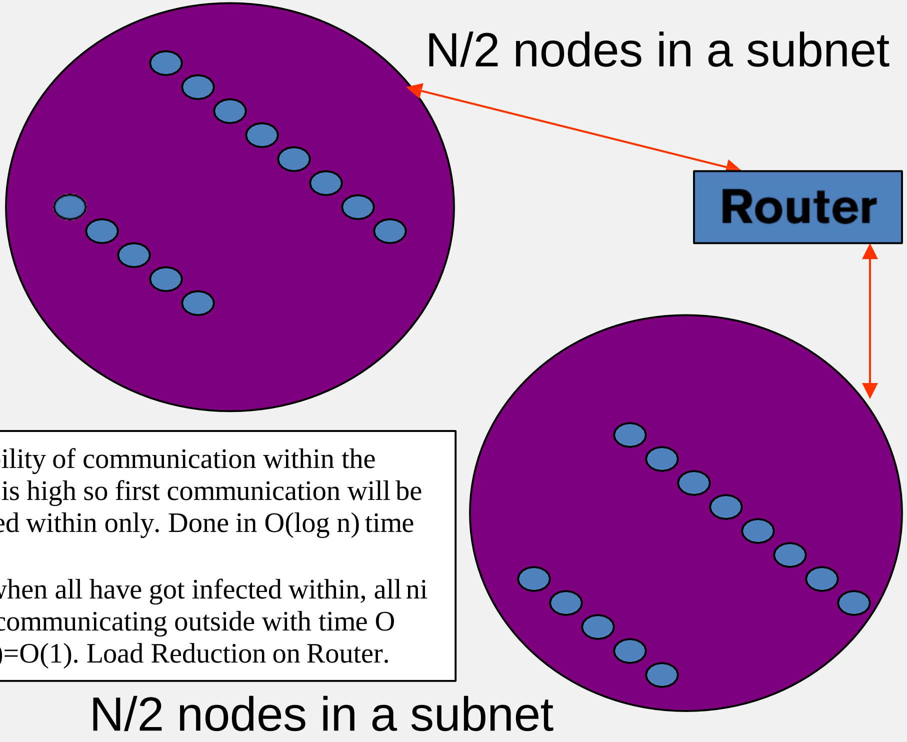
- This is super-exponential
- Second half of pull gossip finishes in time O(log(log(N))

Topology-Aware Gossip

- Network topology is hierarchical
- Random gossip target selection => core routers face O(N) load (Why?)
- Fix: In subnet *i*, which contains n_i nodes, pick gossip target in your subnet with probability $1-1/n_i$ and

outside with 1/ni

- Router load=O(1)



Probability of communication within the subnet is high so first communication will be confined within only. Done in O(log n) time

Later when all have got infected within, all ni nodes communicating outside with time O (n*1/n)=O(1). Load Reduction on Router.

CLOUD COMPUTING CONCEPTS

with Indranil Gupta (Indy)

GOSSIP

Lecture D

GOSSIP IMPLEMENTATIONS

SO, ...

- Is this all theory and a bunch of equations?
- Or are there implementations yet?

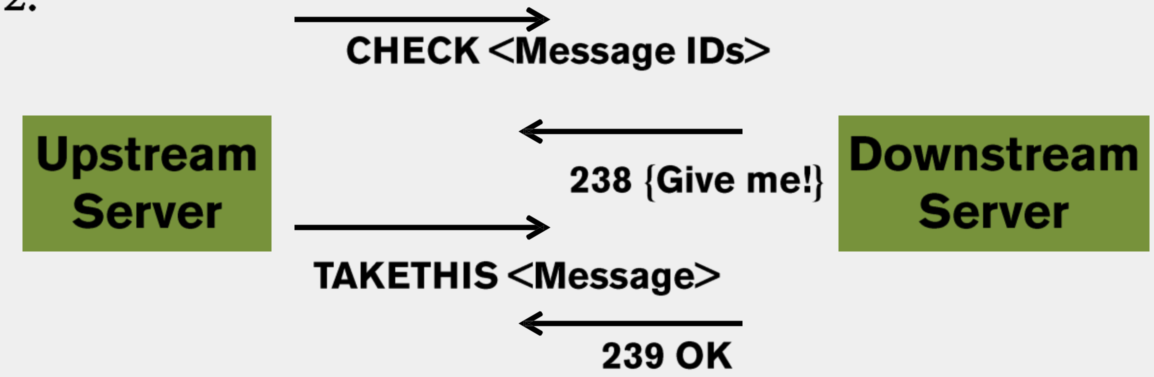
Some implementations

- Clearinghouse and Bayou projects: email and database transactions [PODC ' 87]
- refDBMS system [Usenix ' 94]
- Bimodal Multicast [ACM TOCS ' 99]
- Sensor networks [Li Li et al, Infocom ' 02, and PBBF, ICDCS ' 05]
- AWS EC2 and S3 Cloud (rumored). [' 00s]
- Cassandra key-value store (and others) use gossip for maintaining membership lists
- Usenet NNTP (Network News Transport Protocol) [' 79]

NNTP Inter-server Protocol

1. Each client uploads and downloads news posts from a news server

2.



Server retains news posts for a while,
transmits them lazily, deletes them after a while.

Summary

- Multicast is an important problem
- Tree-based multicast protocols
- When concerned about scale and fault-tolerance, gossip is an attractive solution
- Also known as epidemics
- Fast, reliable, fault-tolerant, scalable, topology-aware

Prof. Indranil Gupta
and
All of you