Gossip and Epidemic Approaches

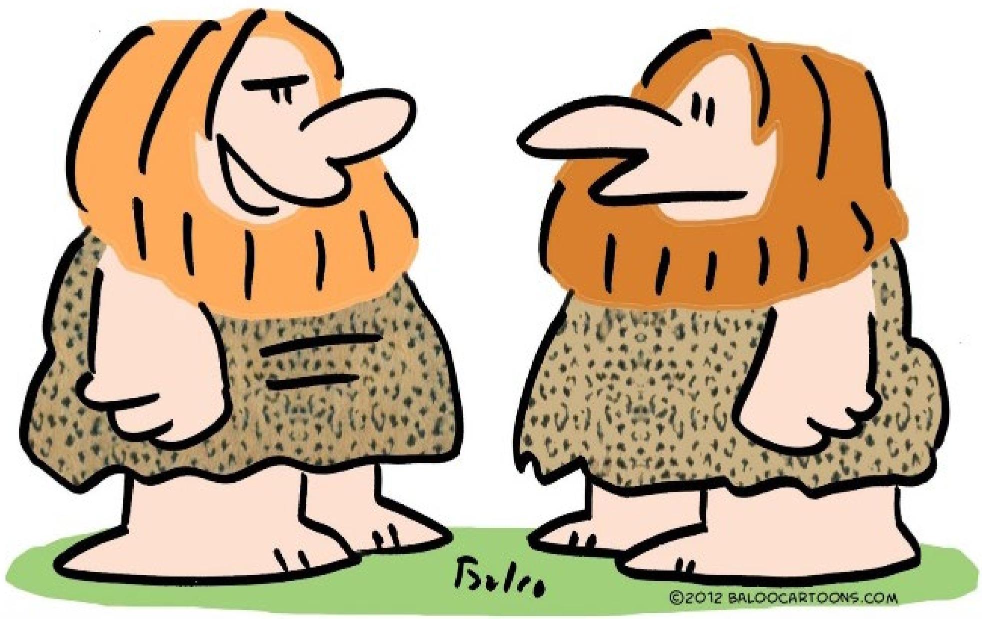
CS 6410: Advanced Systems

Fall 2024

Hakim Weatherspoon

Cornell Bowers CIS
Computer Science

[Slides borrowed liberally from Ki Suh Lee and Eugene Bagdasaryan]



"Now that we've learned to talk,
let's start some *rumors!*"

What is the big idea?

What are these ideas aimed for?

What is the difference with other approaches?

What is the big idea?

What are these ideas aimed for?

- Data consistency, fault-tolerance

What is the difference with other approaches?

- “Eventual” consistency, scalability, fault-tolerance

CAP theorem

CAP = Consistency, Availability, Partition tolerance

- Other approaches focus on Consistency and Partition Tolerance
  E.g. Paxos sometimes is unavailable for writes, but would remain consistent

- This paper wants to provide Availability, Partition Tolerance, and “relaxed” form of consistency; i.e. eventual consistency
  i.e. all replicas have all updates *eventually*

EPIDEMIC ALGORITHM FOR REPLICATED DATABASE MAINTENANCE

Xerox Palo Alto Research Center 1987

Authors

Alan Demers
Cornell Univ.

Dan Greene
Palo Alto
Research Center

Carl Hauser
Washington State
Univ.

Wesley Irish
Coyote Hill
Consulting LLC

Scott Shenker
EECS Berkley

Doug Terry
Samsung Research
America

John Larson
Howard Sturgis
Dan Swinehart

Timeline

1978	**Lamport**	Time, Clocks, and the Ordering of Events in a Distributed System
1982	**Lamport**	The Byzantine Generals Problem
1985	**FLP**	Impossibility of Distributed Consensus with One Faulty Process
1987	**Demers**	Epidemic algorithms for replicated database maintenance
1990	**Schneider**	Implementing fault-tolerant services using the state machine approach: A tutorial
1995	**Terry**	Managing update conflicts in Bayou, a weakly connected replicated storage system
1998	**Lamport**	The part-time parliament

Real applications

- Amazon Web Services (AWS), Microsoft Azure blobstore, Google
- Uber
- Apache Cassandra
- Docker’s multi-host networking
- Cloud providers multi node networking (Heroku)

Context

- Xerox wanted to replicated a database on to hundreds to thousand sites
- Each update is injected at a single site and must be propagated to all other sites
- Xerox Corporate Internet (CIN): A packet from a machine in Japan to one in Europe may traverse as many as 14 gateways and *7 phone lines*
- CIN predates the Internet

Problem

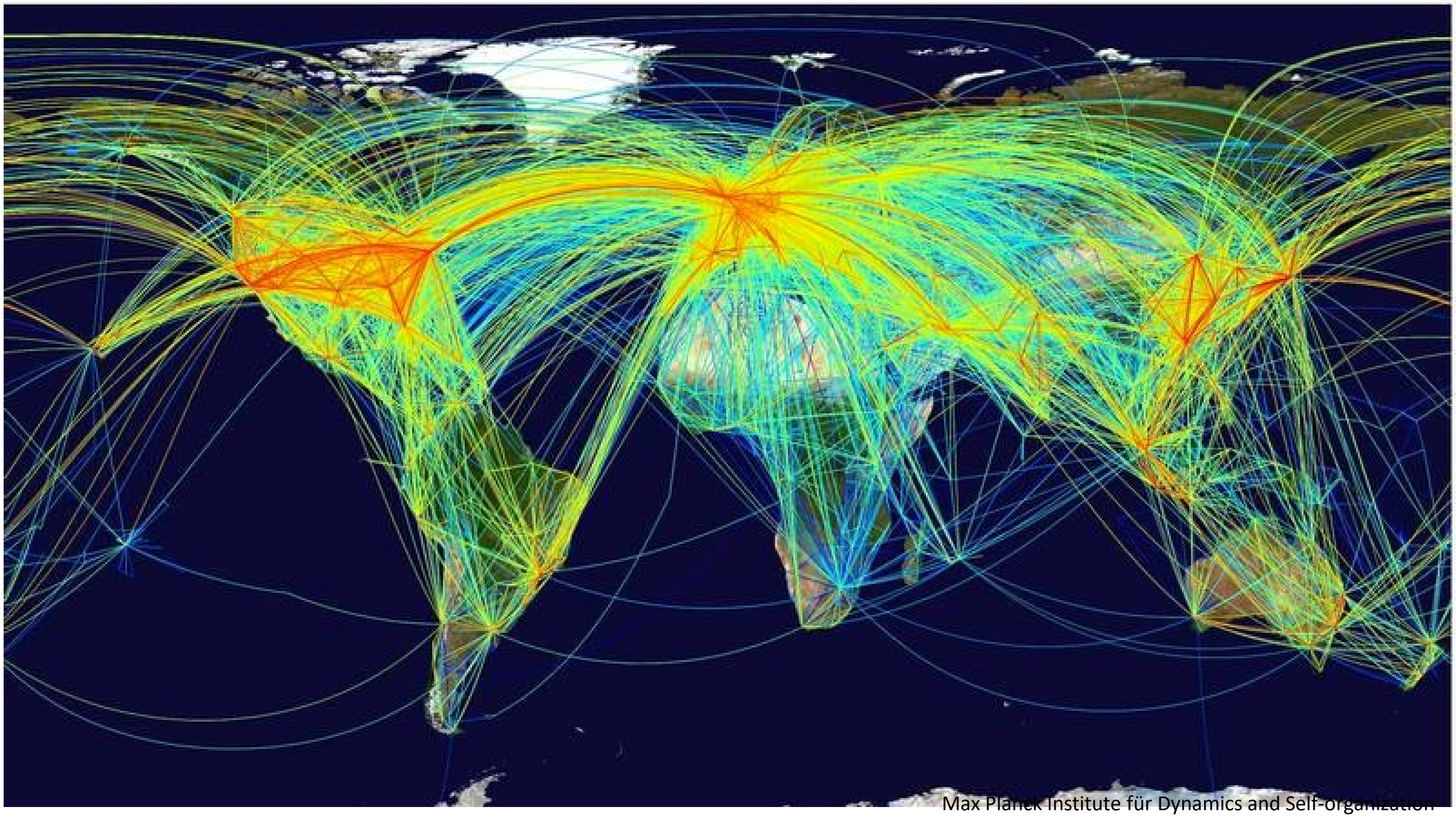
- High network traffic to send update over the large set of nodes
- Time to propagate update to all nodes is significant

Problem

- High network traffic to send update over the large set of nodes
- Time to propagate update to all nodes is significant

” For a domain stored at 300 sites, 90,000 mail messages might he introduced each night”.

Basic idea



Max Planck Institute für Dynamics and Self-organization

Objective

- Design algorithms that scale gracefully
- Every replica receives every update ***eventually***

Objective

- Design algorithms that scale gracefully
- Every replica receives every update ***eventually***

“Replace complex deterministic algorithms for replicated database consistency with simple randomized algorithms that require few guarantees from the underlying communication system.“

Why epidemic? Why gossip?

- Highly available
- Fault-tolerant
- Overhead is tunable
- Fast
- Scalable
- Epidemic spreads eventually to everyone

Types of nodes

- **infective** – node that holds an update it is willing to share
- **susceptible** – node that has not yet received an update
- **removed** – node that has received an update but is no longer willing to share

$$s + i + r = 1$$

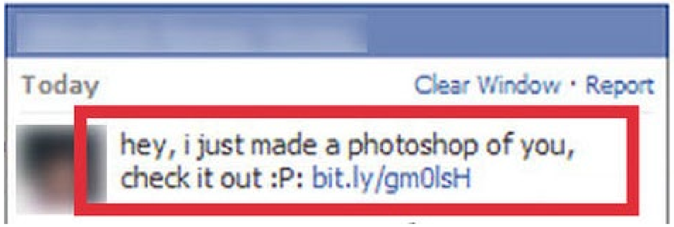
Types of communication

- Direct mail
- Anti-entropy
- Rumor mongering

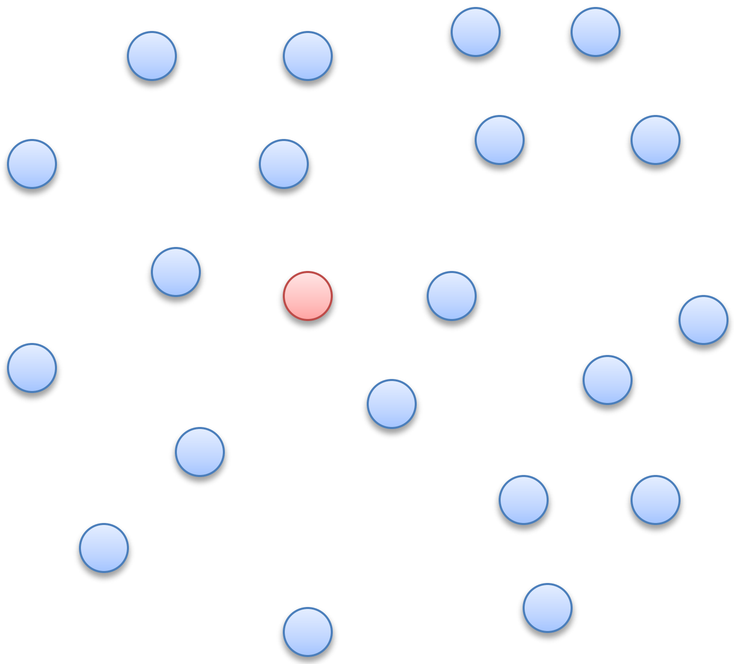
DIRECT MAIL

- attempts to notify all other sites of an update soon after it occurs.

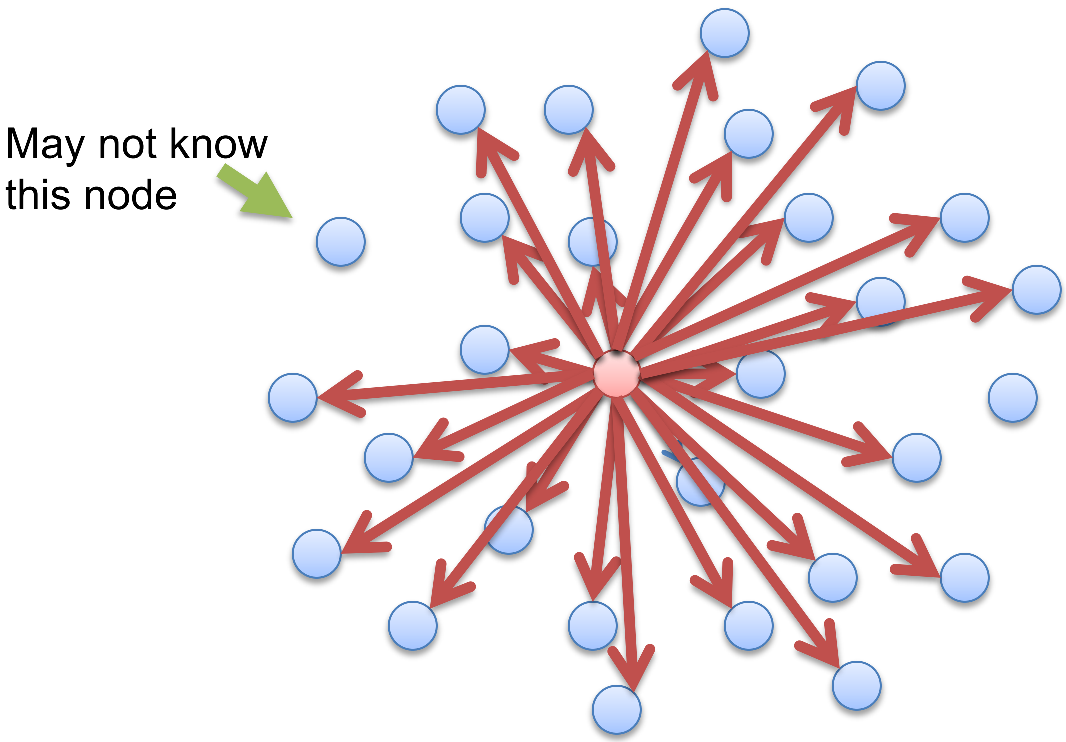
- **Social network case** – infected accounts sends private message to his whole contact list with malicious link



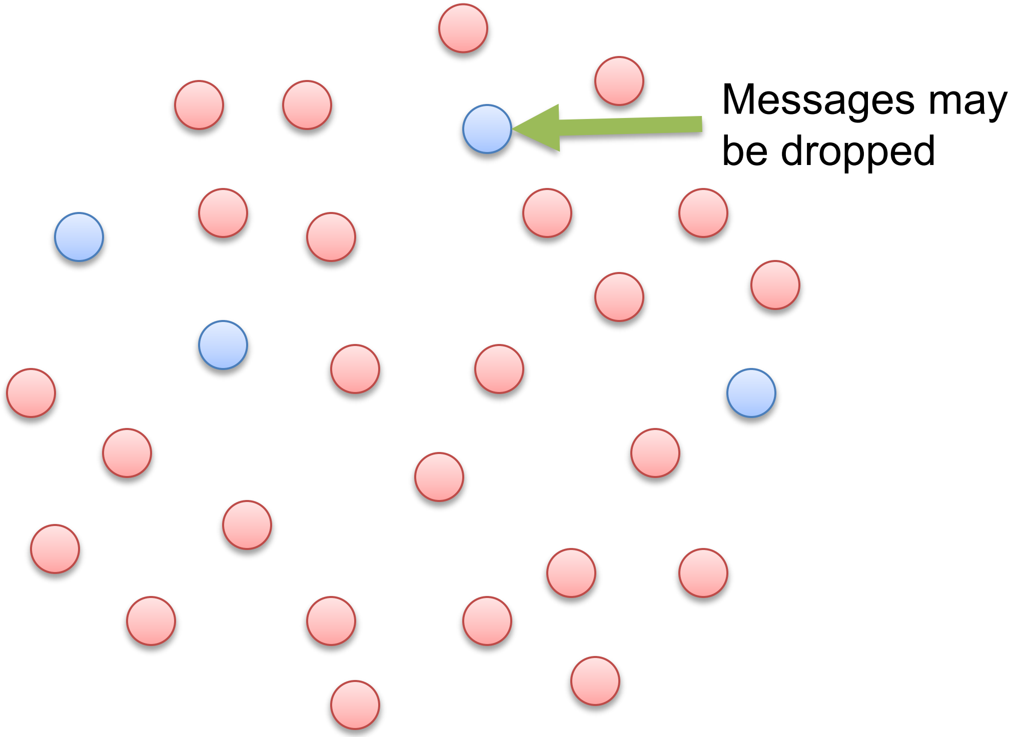
DIRECT MAIL



DIRECT MAIL



DIRECT MAIL



DIRECT MAIL

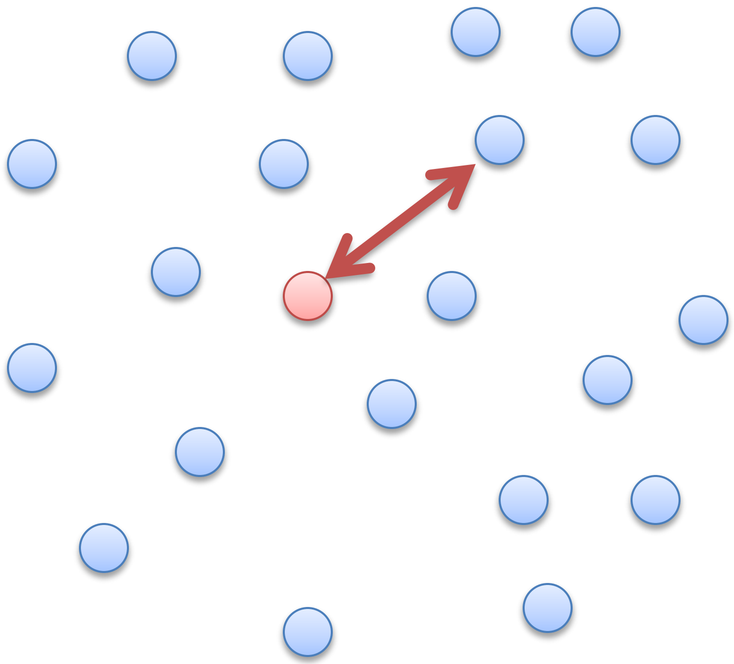
- Pros:
  - Fast
- Cons:
  - not reliable
  - heavy load on network

ANTI-ENTROPY

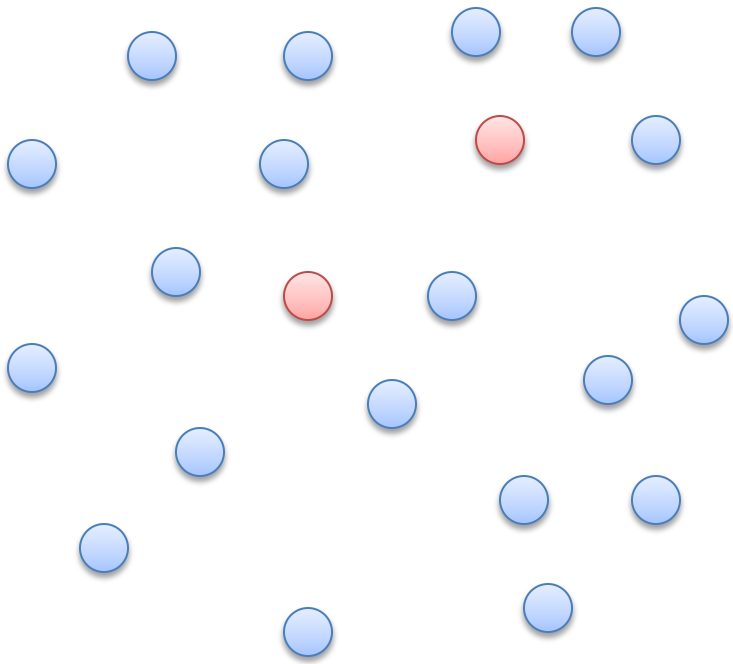
- Every site regularly chooses another site at random and by exchanging database contents with it resolves any differences between the two

- **Real life case** – meet sometimes with old friends and tell all the fun stories about you and your friends.

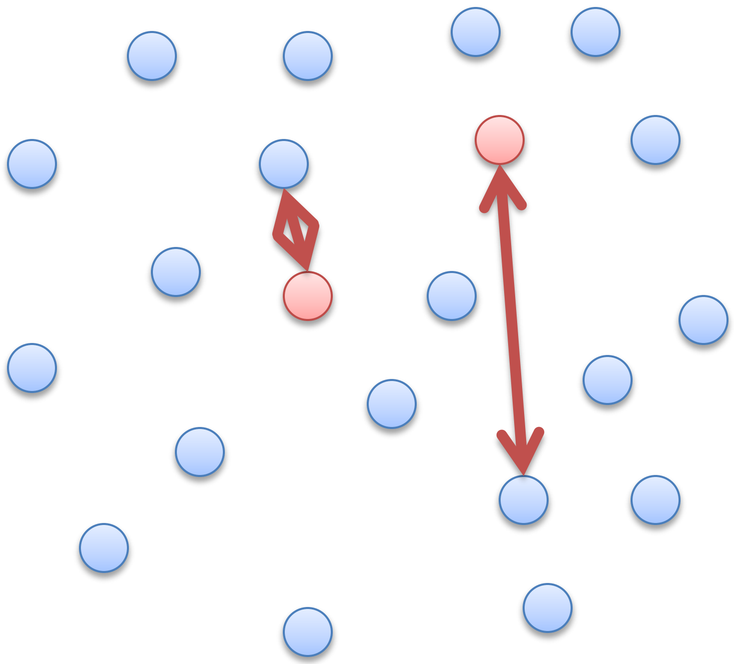
ANTI-ENTROPY



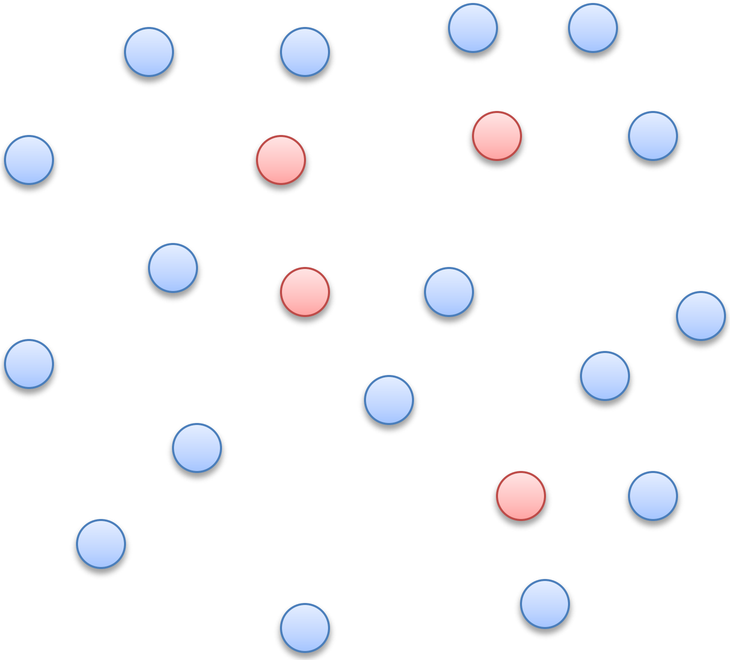
ANTI-ENTROPY



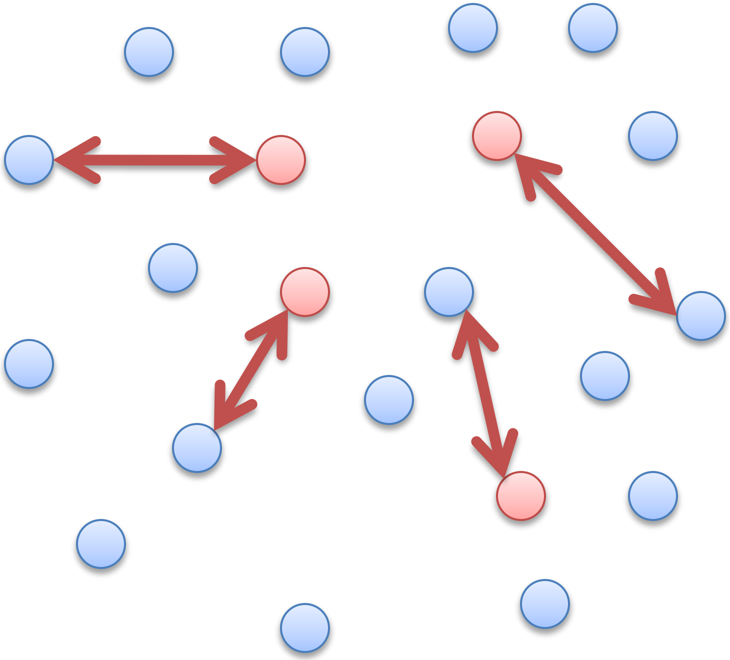
ANTI-ENTROPY



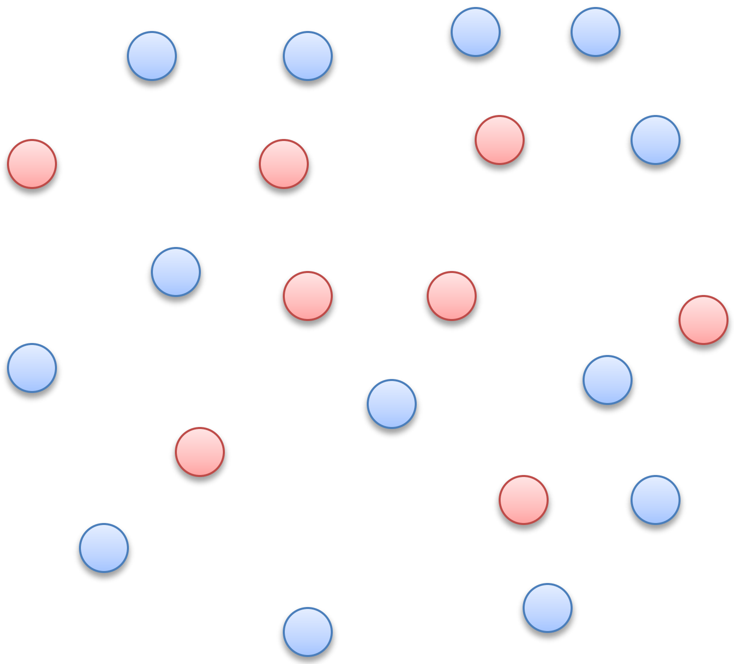
ANTI-ENTROPY



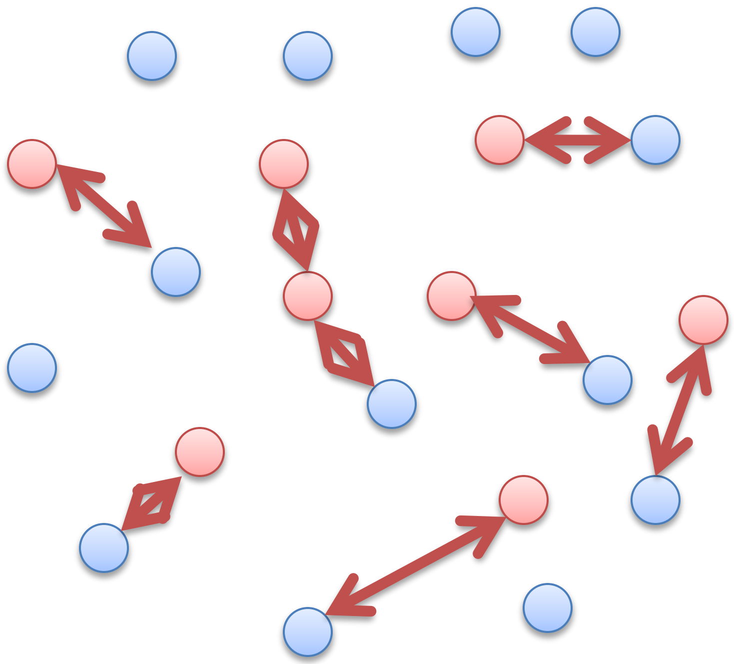
ANTI-ENTROPY



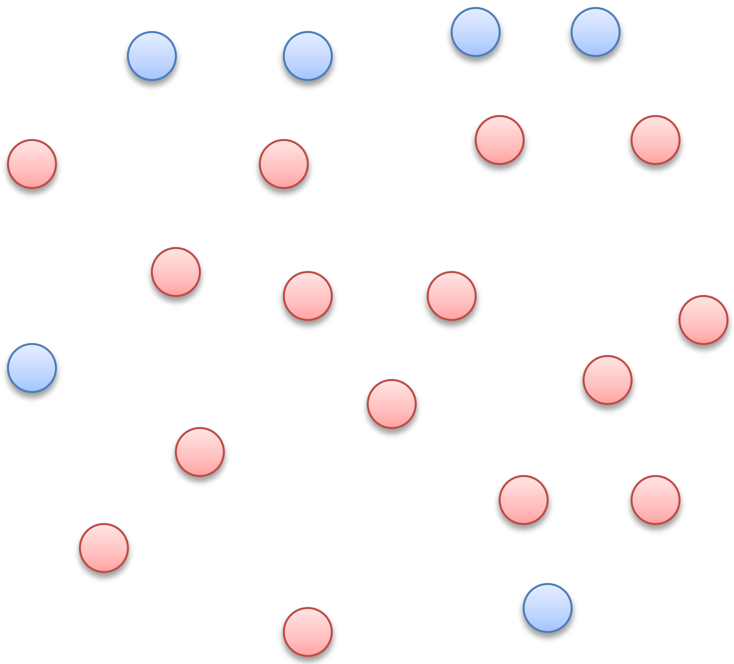
ANTI-ENTROPY



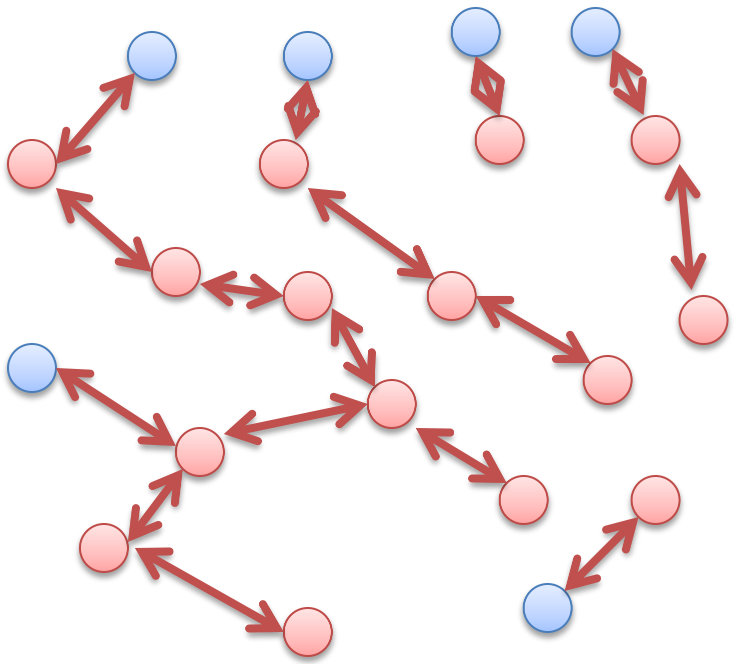
ANTI-ENTROPY



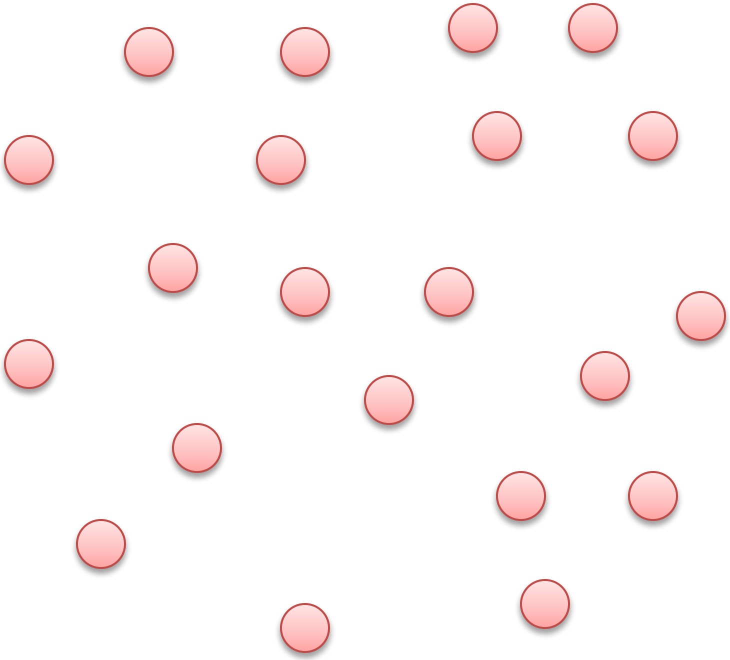
ANTI-ENTROPY



ANTI-ENTROPY



ANTI-ENTROPY



Anti-entropy

Pros

- Complete sync of all info

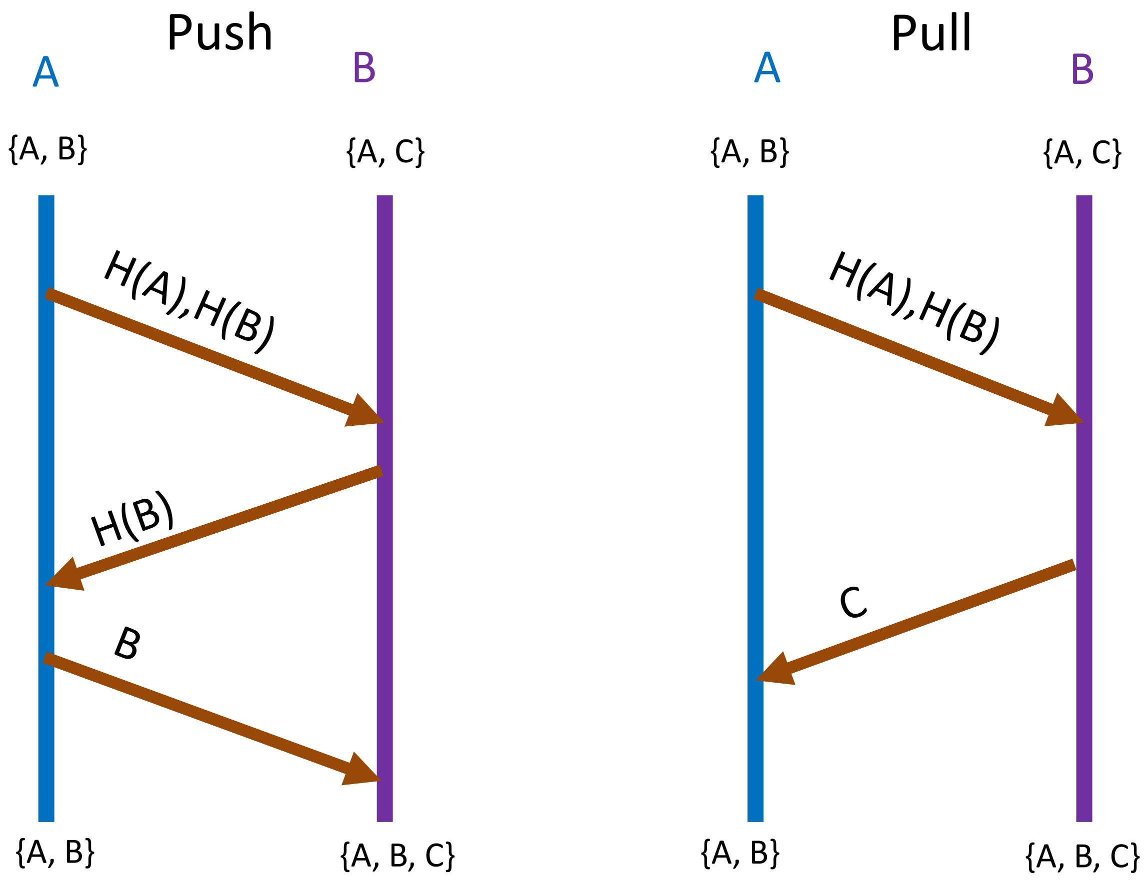
Cons

- Very expensive to run

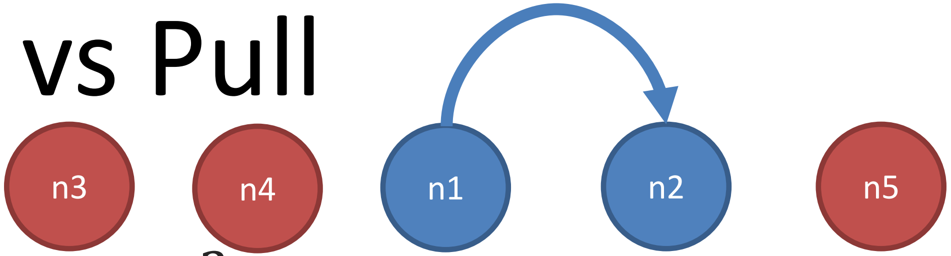
Optimizations:

- Checksums
- Recent Update Lists
- Inverted Index by timestamp

Push vs Pull



Push vs Pull



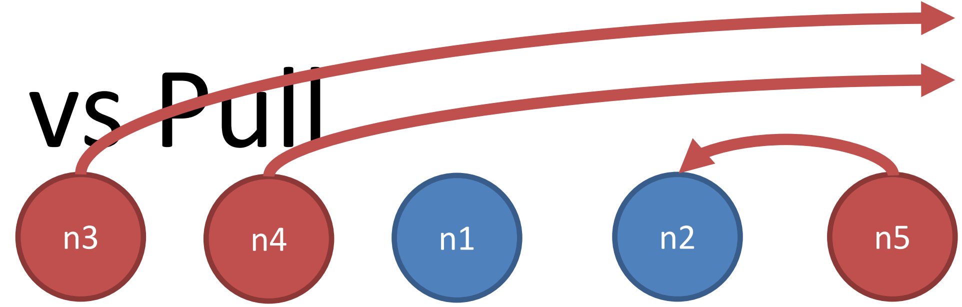
- Pull or Push-pull

$$p_{i+1} = (p_i)^2$$

p_i - probability of a site remaining susceptible after i-th round

To remain susceptible **n1** needs to contact another node **n2** on round **i+1**, which is also susceptible (with probability p_i)

Push vs Pull



- Push

$$p_{i+1} = p_i(1 - \frac{1}{n})^{n(1-p_i)}$$

p_i - probability of a site remaining susceptible after i-th round

$(1 - \frac{1}{n})$ – prob an infected node choose everything except the selected node **n1**

$n(1 - p_i)$ – amount of infected nodes

Push vs Pull

- Pull or Push-pull

$$p_{i+1} = (p_i)^2$$

- Push

$$p_{i+1} = p_i(1 - \frac{1}{n})^{n(1-p_i)} \approx p_i e^{-1}$$

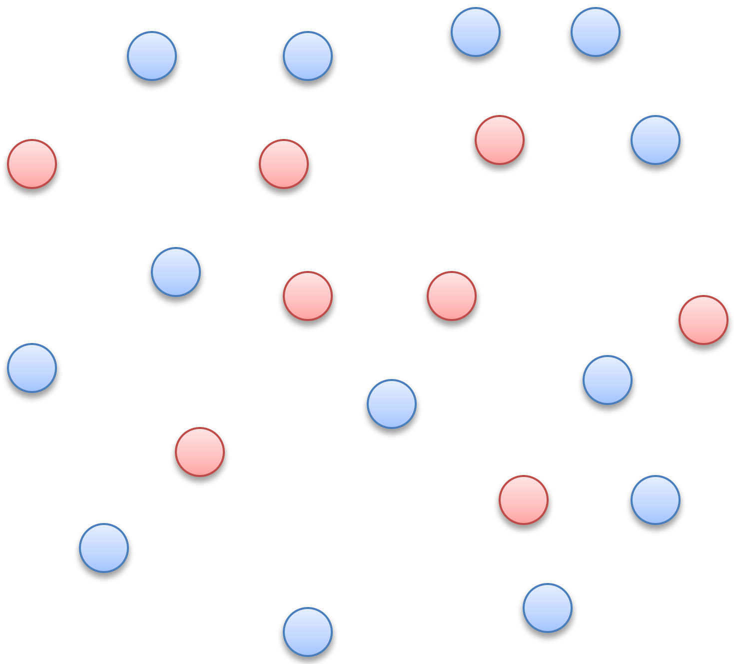
Pull converges to 0 much faster

Rumor mongering

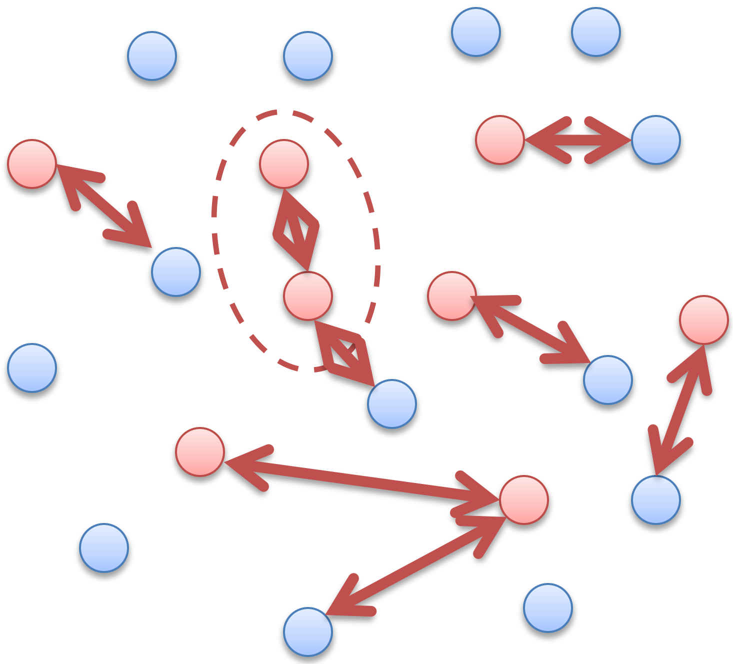
- Share an update, while it is hot. When everyone knows about it stop spreading.

- **News case** – newspapers write more articles on trending topics spreading information.

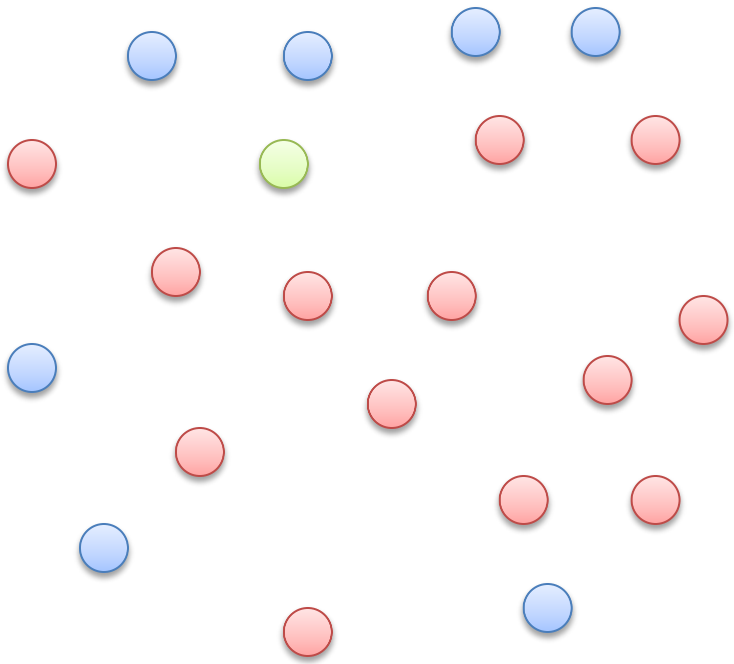
RUMOR MONGERING



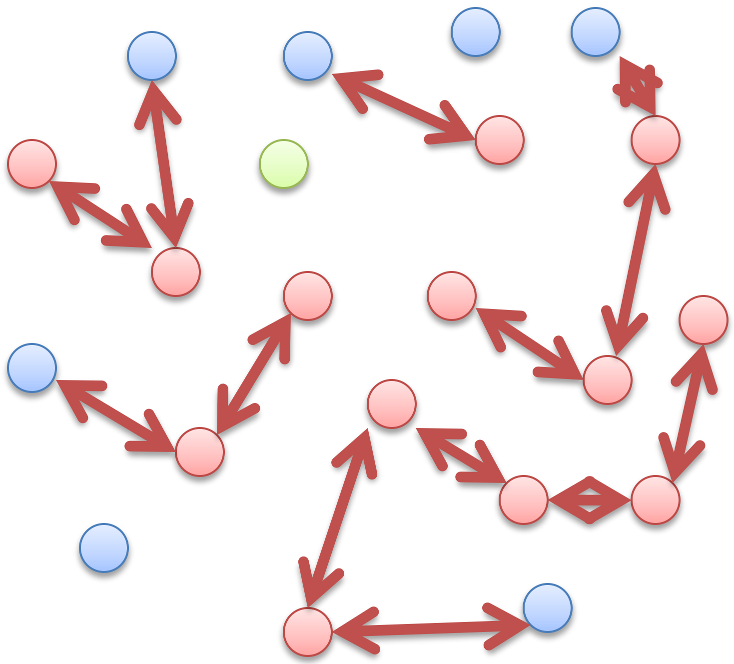
RUMOR MONGERING



RUMOR MONGERING



RUMOR MONGERING



RUMOR MONGERING

Pros

- Less traffic, than Direct mail
- Fast

Cons

- Some sites could miss the information

Can be improved by Complex Epidemics

Complex epidemics

- Hot rumors analogy
- Based on epidemiology literature

  $s + i + r = 1,$ s- susceptible, i - infective, r – removed
- If node contacted already infected node, it loses interest and stops talking with probability $\mathbf{1/k}$
- If k=1, 20% will miss the rumor for k=2 only 6%

$$s = e^{-(k+1)(1-s)}$$

Complex epidemics

Criteria:

- Residue

  Amount of untouched nodes (s) after epidemics ended ($i = 0$) in $s + i + r = 1$

- Traffic

$$m = \frac{Total\ update\ traffic}{Number\ of\ sites}$$

- Delay

  Introduced t_{avg} and t_{last}

Variations

- Blind vs. Feedback
- Counter vs. Coin
- Push vs. Pull
- Minimization
- Connection Limit
- Hunting

Table 1. Push, Feedback & Counters

Counter	Residue	Trafic	Convergence	
k	s	m	t_{avg}	t_{last}
1	0.176	1.74	11.0	16.8
2	0.037	3.30	12.1	16.9
3	0.011	4.53	12.5	17.4
4	0.0036	5.64	12.7	17.5
5	0.0012	6.68	12.8	17.7

Table 2. Push, Blind & Coin

1	0.960	0.04	19	38
2	0.205	1.59	17	33
3	0.060	2.82	15	32
4	0.021	3.91	14.1	32
5	0.008	4.95	13.8	32

Table 3. Pull, Feedback & Counters

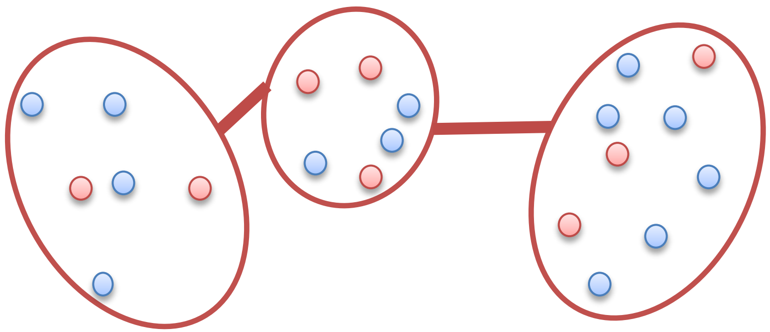
1	0.031	2.7	9.97	17.63
2	0.00058	4.49	10.07	15.39
3	0.000004	6.09	10.08	14.00

Deletion

- Death Certificates
  - Dormant DC
    - Too long to distribute
    - Can be lost
  - Anti-entropy with Dormant DC
    - Activate DC on sync with another node, if this node doesn’t have it
  - Rumor mongering with Dormant DC
    - Parallel to normal data distribution through rumor mongering

Spatial Distributions

- Different weights on connections between nodes
- Can reduce traffic on critical links
- Favor nearby neighbors
- Trade off between convergence time and average traffic per link



Perspective/Questions?

Perspective

- Fast, eventually consistent protocol
- Low traffic in the system

Potential problems:

- Weird topology can decrease performance
- Byzantine Failures

Before Next Time

- Read papers below and write review
  - ***End-to-end arguments in system design,*** J.H. Saltzer, D.P. Reed, D.D. Clark. *ACM Transactions on Computer Systems (TOCS)*, Volume 2, Issue 4 (November 1984), pages 277-288 http://portal.acm.org/citation.cfm?id=357402
  - ***Hints for computer system design***, B. Lampson. *ACM Symposium on Operating Systems Principles (SOSP)*, 1983, pages 33-48 https://dl.acm.org/doi/10.1145/800217.806614
- Check website for updated schedule