Gossip Protocols

Márk Jelasity

Hungarian Academy of Sciences and
University of Szeged, Hungary

Introduction

- Gossip-like phenomena are commonplace
  - human gossip
  - epidemics (virus spreading, etc)
  - computer epidemics (malicious agents: worms, viruses, etc)
  - phenomena such as forest fires, branching processes and diffusion are all similar mathematically
- In computer science, epidemics are relevant
  - for security (against worms and viruses)
  - for designing useful protocols (we look at this here)

Outline

- Seminal work by Demers at al, that first coined the term gossip and epidemic protocols
- Other examples of gossip protocols for
  - peer sampling
  - topology maintenance
  - data aggregation

Epidemic Database Updates

- Problem
  - Xerox corporate Internet, replicated databases
  - Each database has a set of keys that have values (along with a time stamp)
  - Goal: all databases are the same, in the face of key updates, removals and additions
  - Updates are made locally and have to be replicated at all sites (300 sites)
- Solution in 1986
  - Anti-entropy and remailing
  - Didn’t work due to huge amount of traffic

Anti-Entropy

- Basic idea: periodic pairwise exchange of new updates
- State: the local database
- In each cycle select a random peer from the members
- resolve differences between the two databases
- some theoretical notes
  - easy to see that eventually all databases get all updates
  - expected time to achieve full coverage is logarithmic (pushpull is fastest)

End-phase convergence of anti-entropy

- Pull
  - p_i is the proportion of not infected nodes in cycle i

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

- Push (slower in the end phase)

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

Rumor spreading

$$\frac{ds}{dt} = -si$$

$$\frac{di}{dt} = si - \frac{1}{k}(1-s)i$$

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

- Rumor spreading
  - Push gossiping, but stop spreading info with probability 1/k if unsuccessful infection attempt (become removed)
  - s: susceptible, i: infective, r: removed
- Eg if k=1, 20% miss the gossip, if k=2, 6% miss it

Some other rumor mongering algorithms

- Some modifications
  - Blind vs feedback: blind is removed with pr. 1/k irrespective of success
  - Counter vs random: counter counts k unsuccessful attempts, random is removed with 1/k probability after each unsuccessful attempt
  - Push vs pull
    - **Push: always $s=e^{-m}$ where s is residue and m is avg number of messages sent by a node (Nm messages are sent altogether, to random targets)**
    - **Pull: better residue, but generates traffic even when there are no updates**

Some empirical results (1000 nodes)

Counter k	Residue s	Traffic m	Convergence t_{ave}	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

Feedback+ Counter+ push

Counter k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
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

Blind+ Random+ push

Counter k	Residue s	Traffic m	Convergence t_{ave}	t_{last}
1	3.1×10^{-2}	2.70	9.97	17.63
2	5.8×10^{-4}	4.49	10.07	15.39
3	4.0×10^{-6}	6.09	10.08	14.00

Feedback+ Counter+ pull

Combining anti-entropy and rumor mongering

- Rumor mongering is used to spread updates
- Anti-entropy is run infrequently to make sure all updates are spread with pr. 1
- When anti-entropy finds an undelivered update: redistribution
  - Redistribution is done via rumor mongering
- [Originally, both primary spreading and redistribution was by email, but costs are prohibitive]

Spacial Gossip

- So far: random contacts
  - This is not good for underlying network traffic
  - Need to take proximity into account
- Spacial gossip: getPeer is biased according to distance of the peer: selecting node i is proportional to d^{-a} where d is the distance of i
- If underlying topology is linear, then expected traffic per link:

$$T(n) = \begin{cases} O(n), & a < 0; \\ O(n/\log n), & a = 1: \\ O(n^{2-a}). & 1 < a < 2; \\ O(\log n). & a = 2; \\ O(1), & a > 2 \end{cases}$$

Spatial Gossip

- a=2 is the best
  - Best tradeoff between speed and traffic
  - Probability is proportional to $1/d^2$
- Generalize to non-linear case
  - Q(d): cumulative number of sites at most at distance d
  - Probability proportional to $1/Q(d)^2$
- Smoothing out pathological topologies
  - Order all sites according to distance
  - Treat it as a linear structure

A Gossip Skeleton

- originally for information dissemination in a very simple but efficient and reliable way
- later the term has been extended to many local probabilistic and periodic protocols
- we will introduce a simple common skeleton and look at
  - information dissemination
  - topology construction
  - aggregation

A Gossip Skeleton

- the push-pull model is sown
- the active thread initiates communication (push) and receives peer state (pull)
- the passive thread mirrors this behavior

```
do once in each T time units at
a random time
    p = selectPeer()
    send state to p
    receive state_p from p
    state = update(state_p)
```

active thread

```
do forever
    receive state_p from p
    send state to p
    state = update(state_p)
```

passive thread

Information dissemination (broadcast)

- state: set of updates
- selectPeer: a random peer from the network
  - very important component, we get back to this soon
- update: add the received updates to the local set of updates
- some notes
  - implementations take care of details to optimize bandwidth usage (check which updates are needed, etc)
  - propagation of one given update can be limited (max k times or with some probability, etc)

Performance of gossip

- various mathematical results are available
  - epidemiological models (virus spreading)
  - percolation theory, complex networks, etc
- underlying network (that is, the implementation of selectPeer) plays a key role
- in a random network
  - push-pull gossip spreads approximately exponentially fast
  - gossip (that is, random networks...) is extremely robust to benign failure (node failure and link failure)

References

- Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87), pages 1–12, Vancouver, British Columbia, Canada, August 1987. ACM Press.
- Eugster, P. T., Guerraoui, R., Kermarrec, A.-M., and Massoulie, L. 2004. Epidemic information dissemination in distributed systems. IEEE Computer 37, 5 (May), 60–67.
- A.-M. Kermarrec and M. van Steen, editors, Special issue of ACM SIGOPS Operating Systems Review on Gossip Protocols, (probably 2007, in press).

Peer Sampling

- A key method is selectPeer in all gossip protocols (determines performance and reliability)
- In earliest works all nodes had a global view to select a random peer from
  - scalability and dynamism problems
- Scalable solutions are available to deal with this
  - random walks on fixed overlay networks
  - dynamic random networks

Random walks on networks

- if we are given any fixed network, we can sample the nodes with any arbitrary distribution with the Metropolis algorithm:

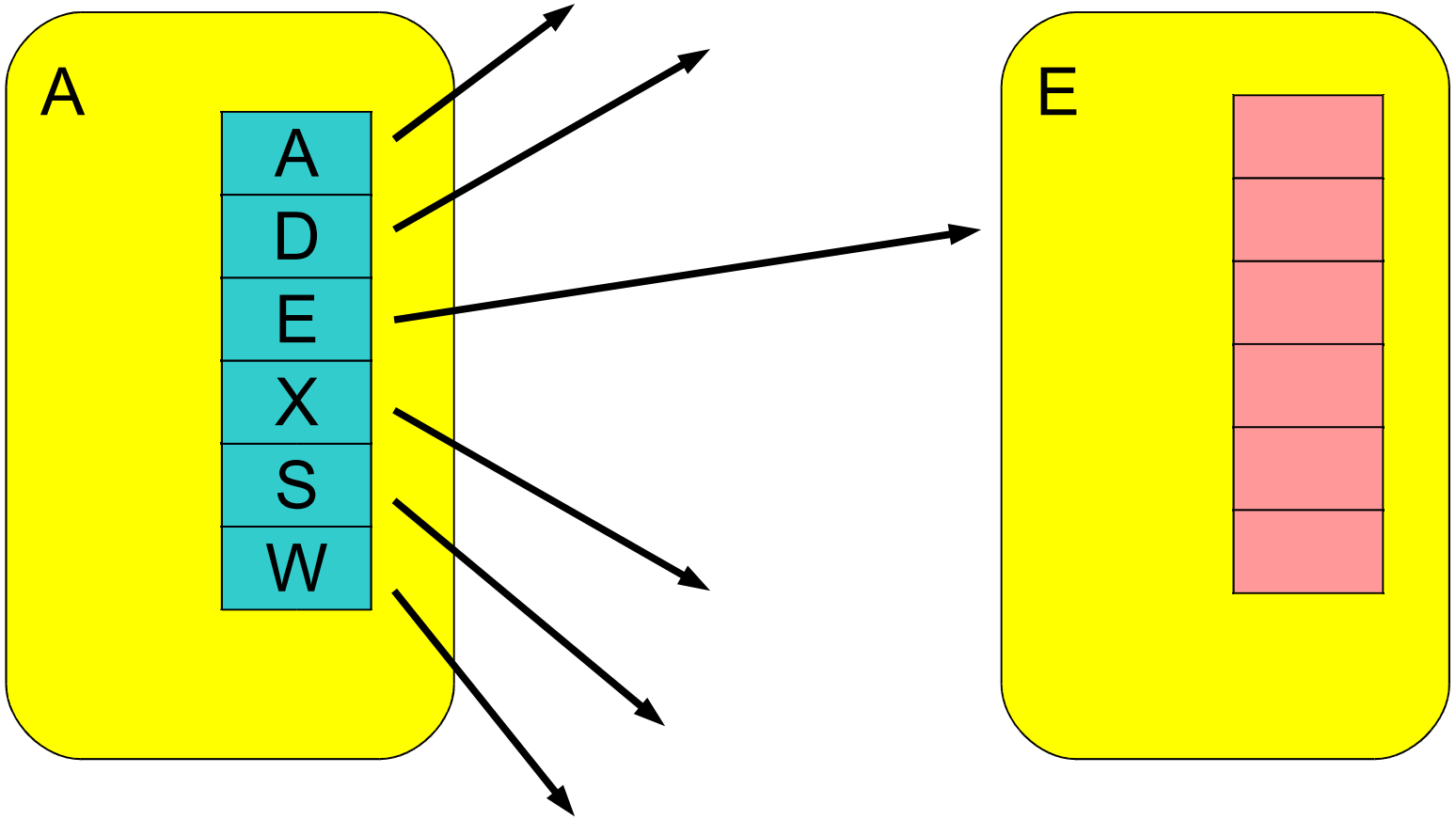
$$P_{i,j} = \begin{cases} \frac{1}{2} \cdot \frac{1}{d_i} & \text{if } \frac{\pi(i)}{d_i} \leq \frac{\pi(j)}{d_j}; \\ \frac{1}{2} \cdot \frac{1}{d_j} \cdot \frac{\pi(j)}{\pi(i)} & \text{if } \frac{\pi(i)}{d_i} > \frac{\pi(j)}{d_j}. \end{cases}$$

- This Markov chain has stationary distribution π where d_i is the degree of node i (undirected graph)

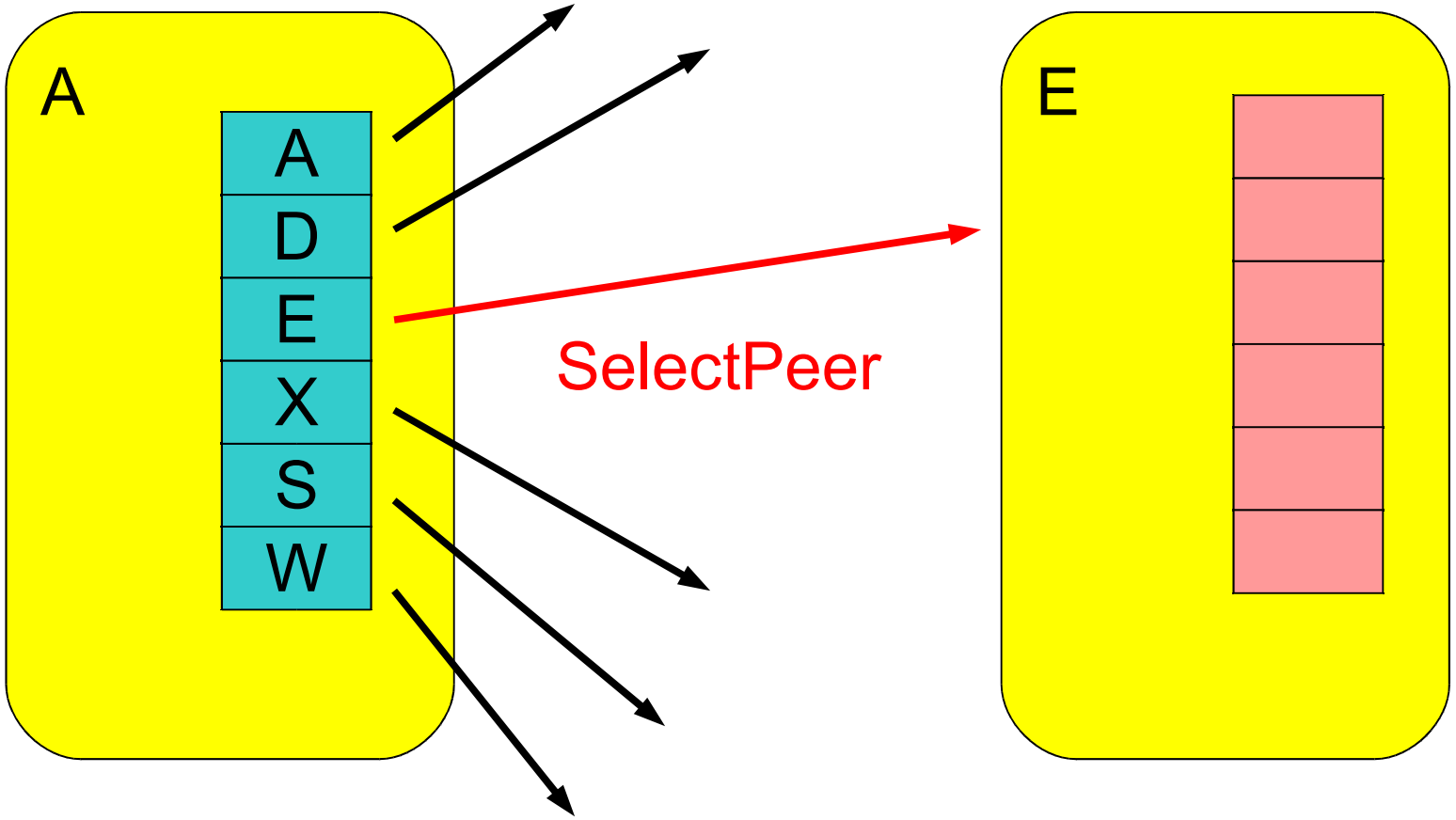
Gossip based peer sampling

- basic idea: random peer samples are provided by a gossip algorithm: the peer sampling service
- The peer sampling service uses itself as peer sampling service (bootstrapping)
  - no need for fixed (external) network
- state: a set of random overlay links to peers
- selectPeer: select a peer from the known set of random peers
- update: for example, keep a random subset of the union of the received and the old link set

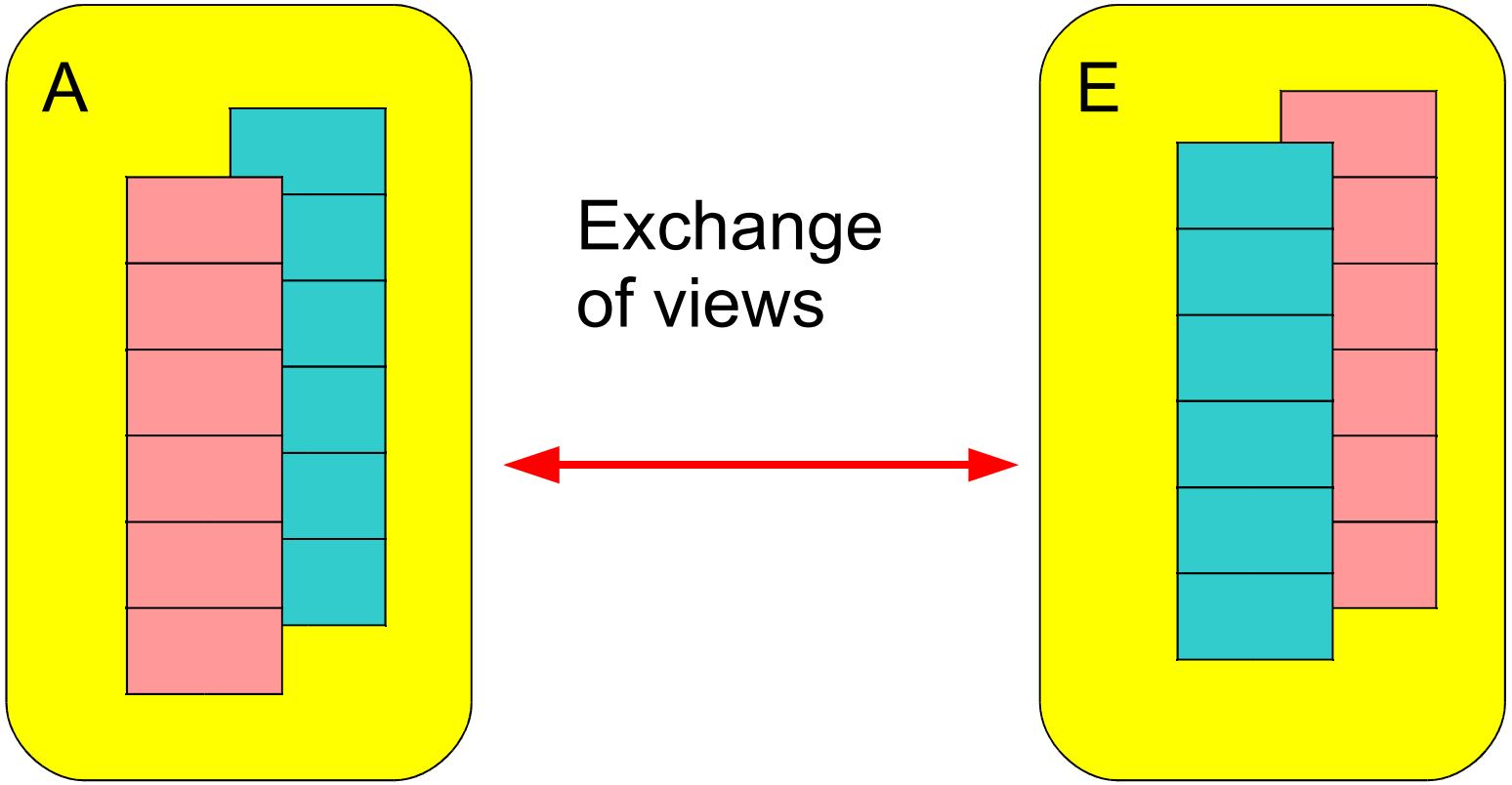
Gossip protocols for topology management



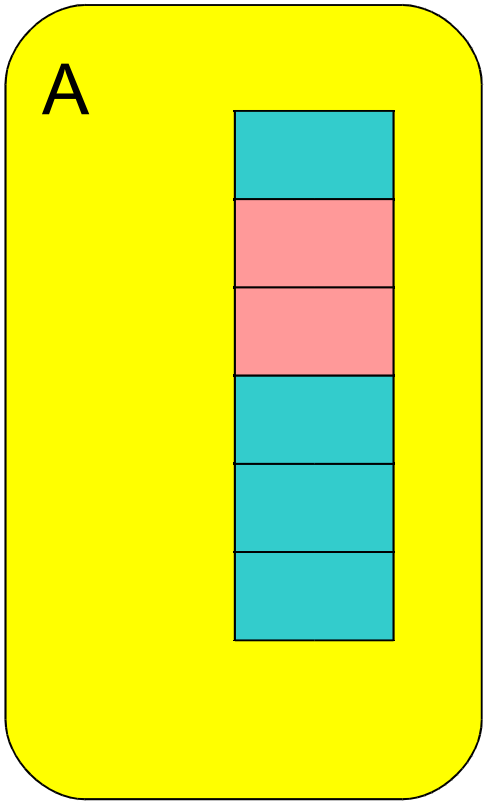
Gossip protocols for topology management



Gossip protocols for topology management

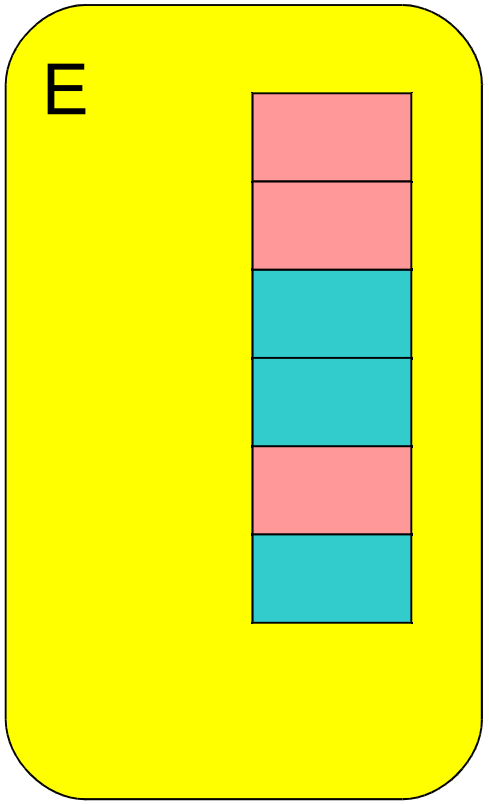


Gossip protocols for topology management



Both sides apply update

thereby redefining topology



Gossip based peer sampling

- in reality a huge number of variations exist
  - timestamps on the overlay links can be taken into account: we can select peers with newer links, or in update we can prefer links that are newer
- these variations represent important differences w.r.t. fault tolerance and the quality of samples
  - the links at all nodes define a random-like overlay that can have different properties (degree distribution, clustering, diameter, etc)
  - turns out actually not really random, but still good for gossip

References

- Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The peer sampling service: Experimental evaluation of unstructured gossip-based implementations. In Hans-Arno Jacobsen, editor, Middleware 2004, volume 3231 of Lecture Notes in Computer Science, pages 79–98. Springer-Verlag, 2004. (journal version: ACM TOCS 2007 aug)
- Zhong, M., Shen, K., and Seiferas, J. 2005. Non-uniform random membership management in peer-to-peer networks. In Proc. of the IEEE INFOCOM. Miami, FL.

Gossip based topology management

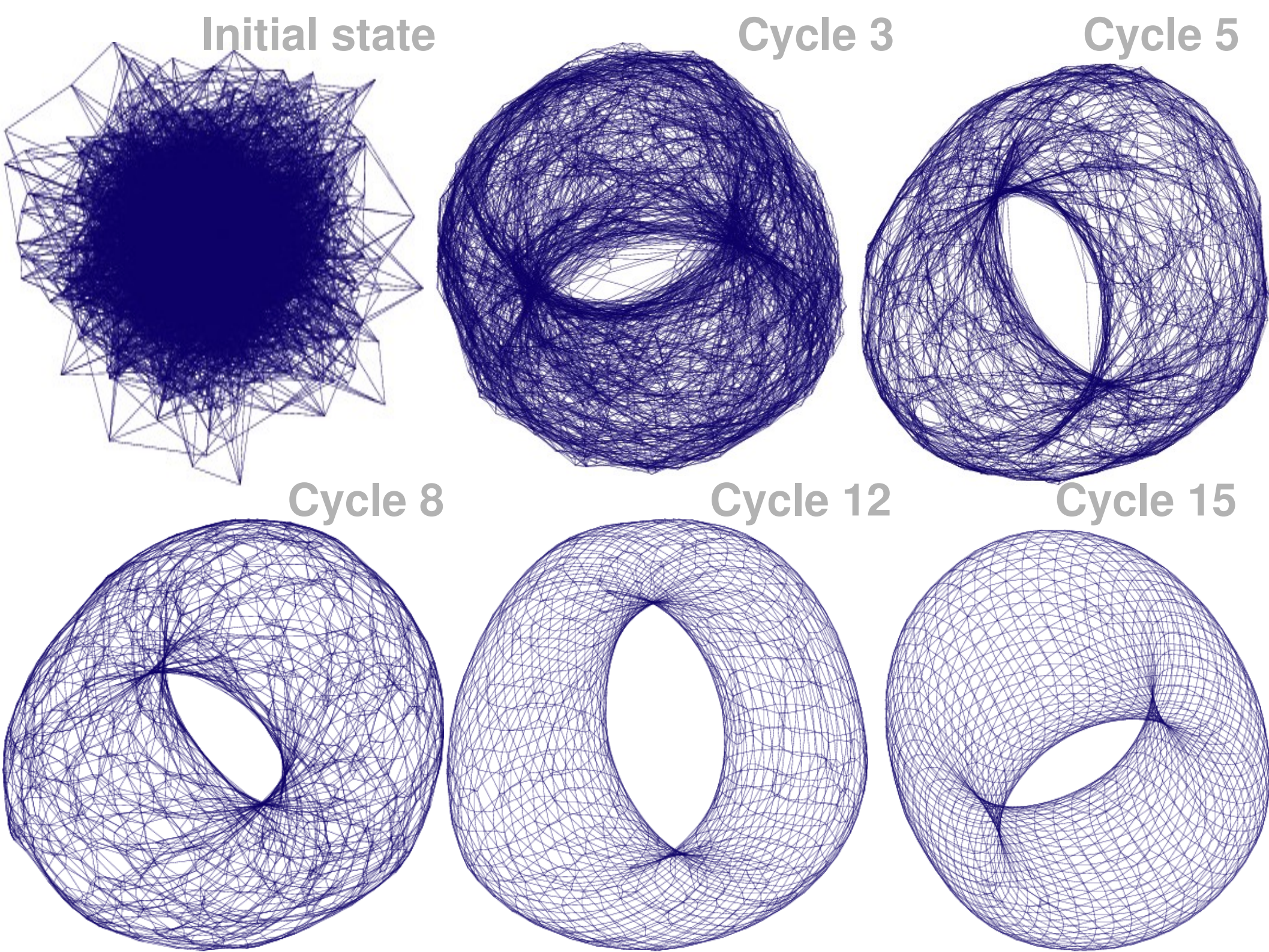
- We saw we can build random networks. Can we build any network with gossip?
- Yes, many examples
  - proximity networks
  - DHT-s (Bamboo DHT: maintains Pastry structure with gossip inspired protocols)
  - semantic proximity networks
  - etc

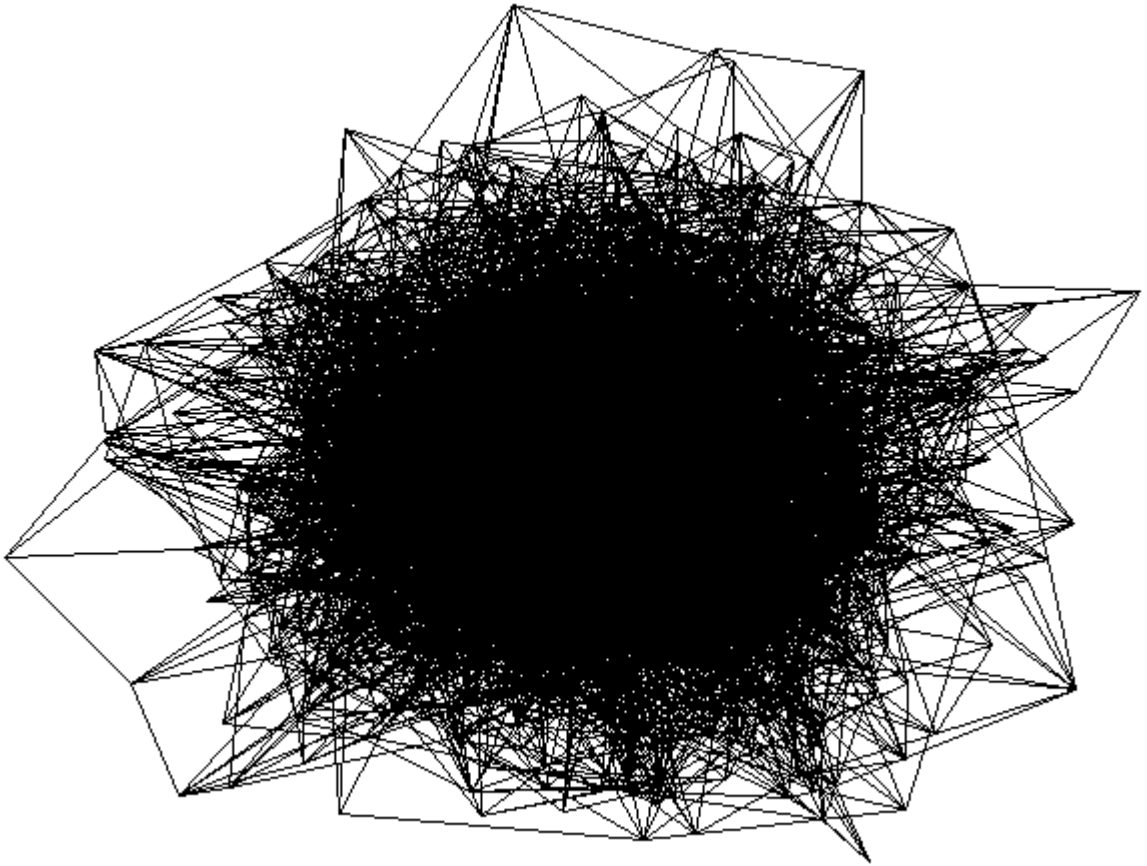
T-Man

- T-MAN is a protocol that captures many of these in a common framework, with the help of the ranking method:
  - ranking is able to order any set of nodes according to their desirability to be a neighbor of some given node
  - for example, based on hop count in a target structure (ring, tree, etc)
  - or based on more complicated criteria not expressible by any distance measure

Gossip based topology management

- basic idea: random peer samples are provided by a gossip algorithm: the peer sampling service
- The peer sampling service uses itself as peer sampling service (bootstrapping)
  - no need for fixed (external) network
- state: a set of overlay links to peers
- selectPeer: select the peer from the known set of peers that ranks highest according to the ranking method
- update: keep those links that point to nodes that rank highest


Initial state
Cycle 3
Cycle 5
Cycle 8
Cycle 12
Cycle 15



References

- Márk Jelasity and Ozalp Babaoglu. T-Man: Gossip-based overlay topology management. In Sven A. Brueckner, Giovanna Di Marzo Serugendo, David Hales, and Franco Zambonelli, editors, Engineering Self-Organising Systems: Third International Workshop (ESOA 2005), Revised Selected Papers, volume 3910 of Lecture Notes in Computer Science, pages 1–15. Springer-Verlag, 2006.
- Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiatowicz. Handling Churn in a DHT. Proceedings of the USENIX Annual Technical Conference, June 2004.
- Laurent Massoulie, Anne-Marie Kermarrec, and Ayalvadi J. Ganesh. Network awareness and failure resilience in self-organising overlay networks. In Proceedings of the 22nd Symposium on Reliable Distributed Systems (SRDS 2003), pages 47–55, Florence, Italy, 2003.
- Spyros Voulgaris and Maarten van Steen. Epidemic-style management of semantic overlays for content-based searching. In Jose C. Cunha and Pedro D.Medeiros, editors, Proceedings of Euro-Par, number 3648 in Lecture Notes in Computer Science, pages 1143–1152. Springer, 2005.

Aggregation

- Calculate a global function over distributed data
  - eg average, but more complex examples include variance, network size, model fitting, etc
- usual structured/unstructured approaches exist
  - structured: create an overlay (eg a tree) and use that to calculate the function hierarchically
  - unstructured: design a stochastic iteration algorithm that converges to what you want (gossip)
- we look at gossip here

Implementation of aggregation

- state: current approximation of the average
  - initially the local value held by the node
- selectPeer: a random peer (based on peer sampling service)
- updateState(s_1,s_2)
  - $(s_1+s_2)/2$: result in averaging
  - $(s_1 s_2)^{1/2}$: results in geometric mean
  - $\max(s_1,s_2)$: results in maximum, etc

Illustration of averaging

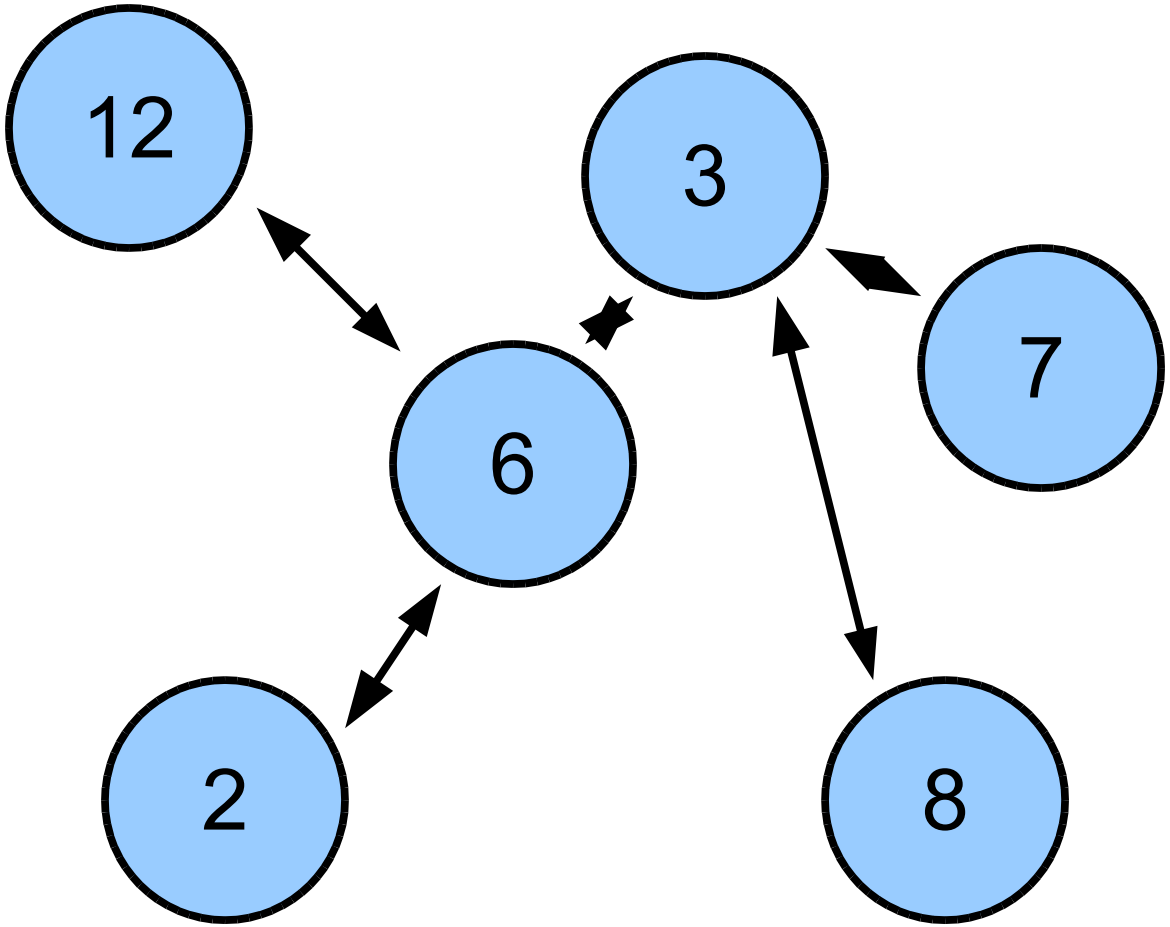


Illustration of averaging

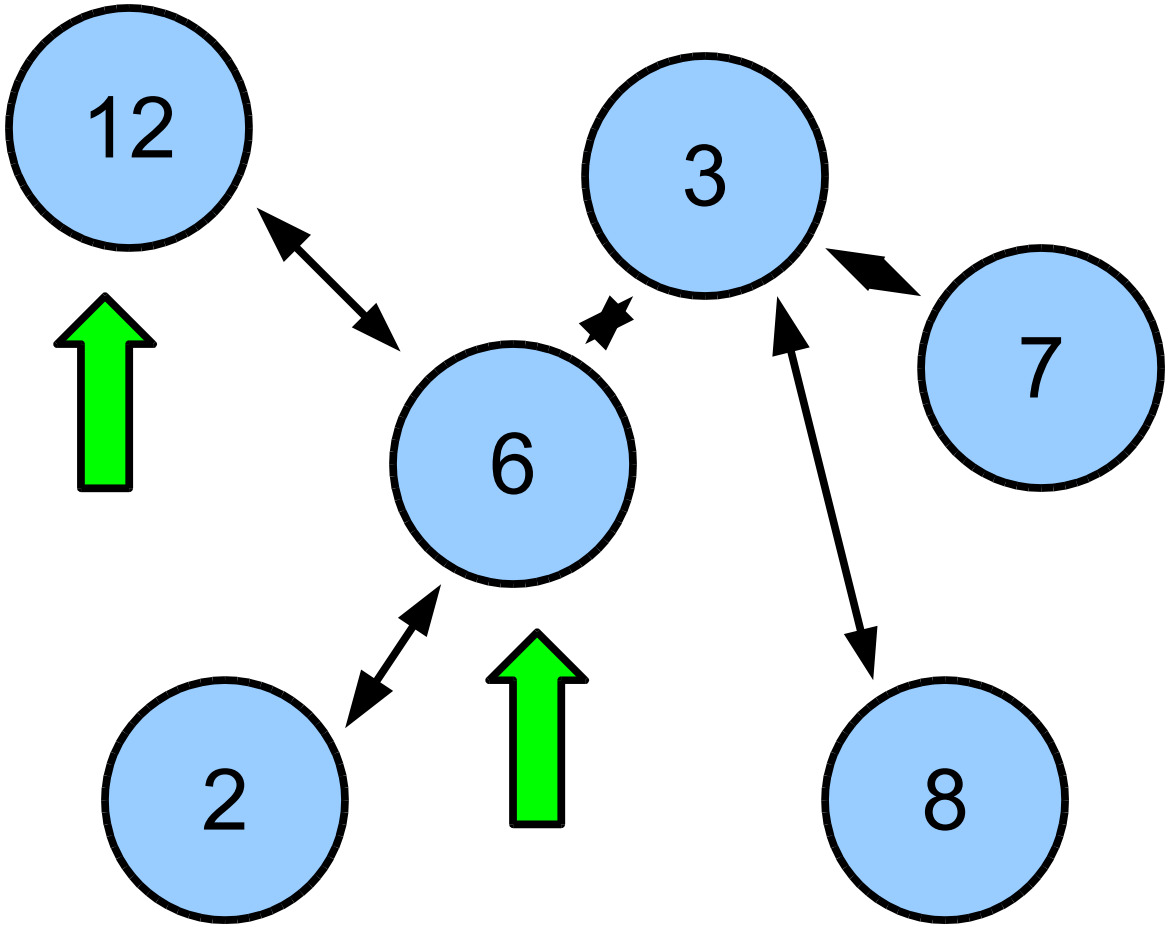
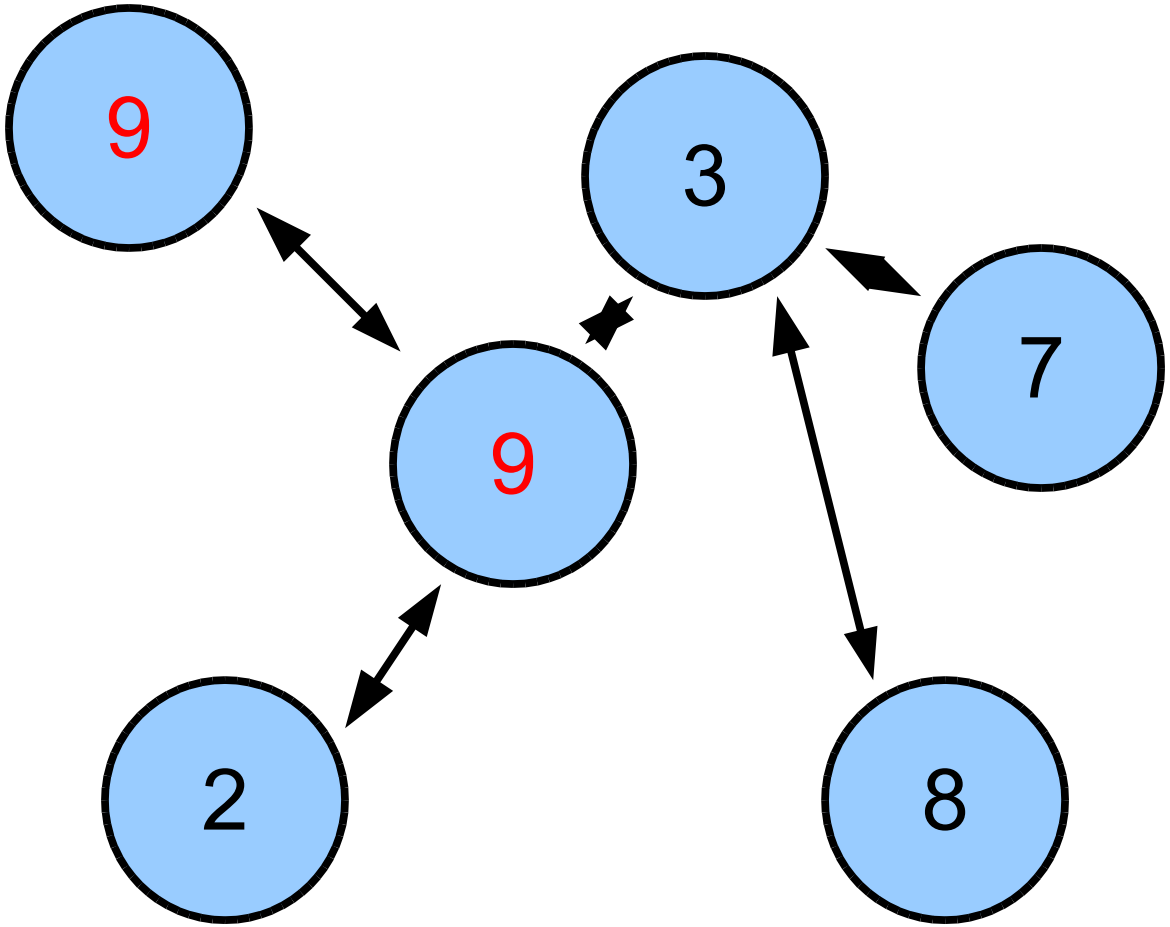


Illustration of averaging

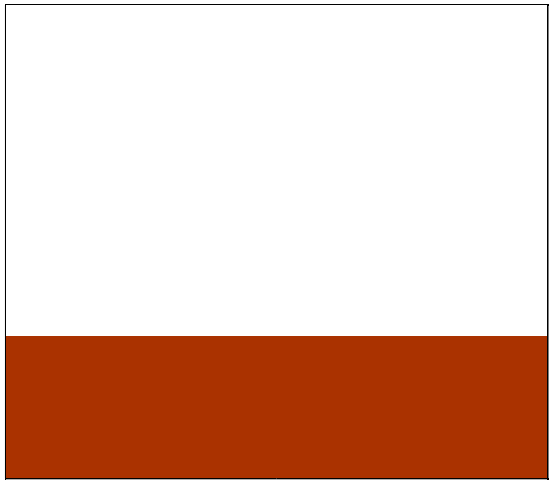


Improvements

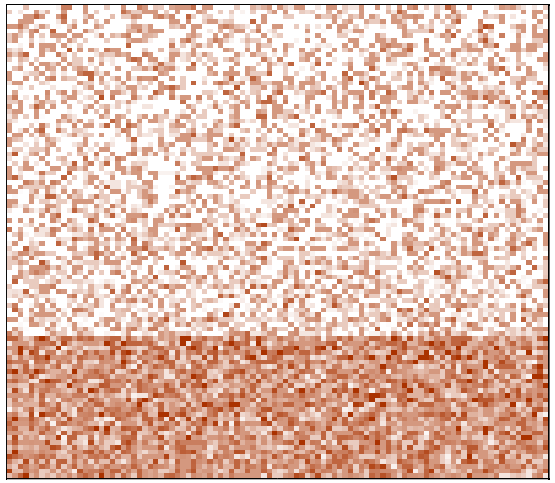
- Tolerates asymmetric message loss (only push or pull) badly
- Tolerates overlaps in pairwise exchanges badly
- [Kempe et al 2003] propose a slightly different version
  - all nodes maintain s (sum estimate) and w (weight)
  - estimate is s/w
  - only push: send (s/2,w/2), and keep s=s/2, w=w/2
- several other variations exist

Illustration of averaging

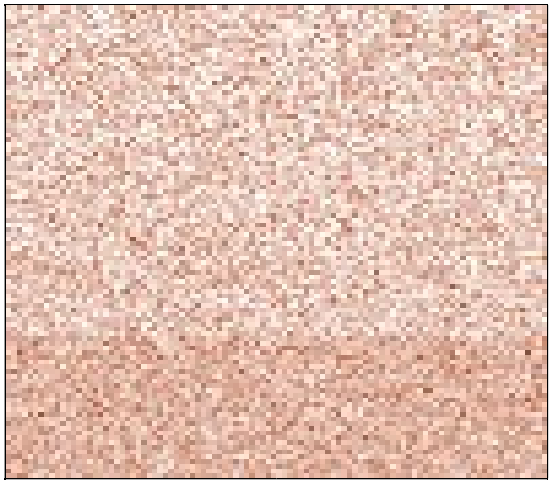
Initial state



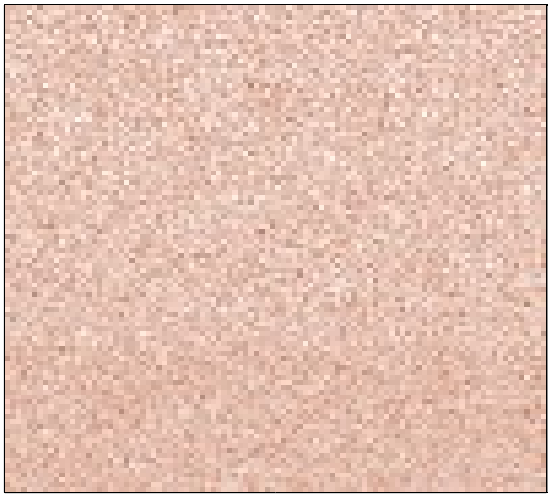
Cycle 1



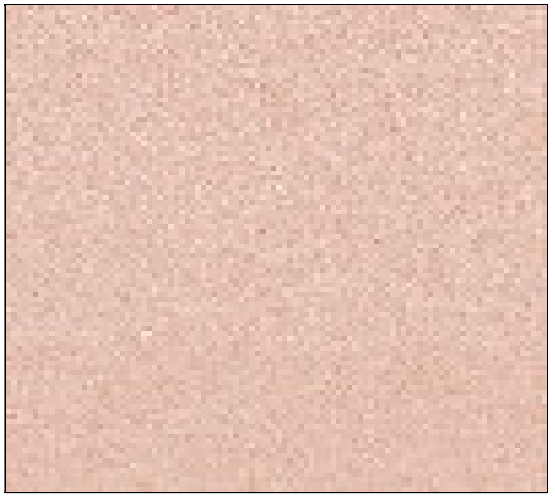
Cycle 2



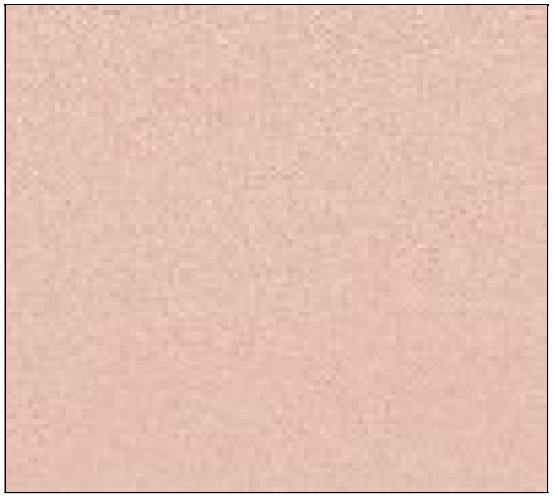
Cycle 3



Cycle 4



Cycle 5



References

- Kempe, D., Dobra, A., and Gehrke, J. 2003. Gossip-based computation of aggregate information. In Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS'03). IEEE Computer Society, 482–491.
- Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-based aggregation in large dynamic networks. ACM Transactions on Computer Systems, 23(3):219–252, August 2005.
- Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. ACM Transactions on Computer Systems, 21(2):164–206, May 2003.

Outlook

- Gossip is similar to many other fields of research that also have some of the following features:
  - periodic, local, probabilistic, symmetric
- examples include
  - swarm systems, cellular automata, parallel asynchronous numeric iterations, self-stabilizing protocols, etc

Astrolabe (middleware)

- Organizes hosts into a domain hierarchy (like DNS)
- Provides online monitoring service based on aggregation; a sort of data mining
- Fully decentralized through gossip
- Allows online configuration of monitoring capabilities (new things to observe, etc)
- Provides an API to applications
- Actually implemented
  - Security, firewalls, etc taken care of

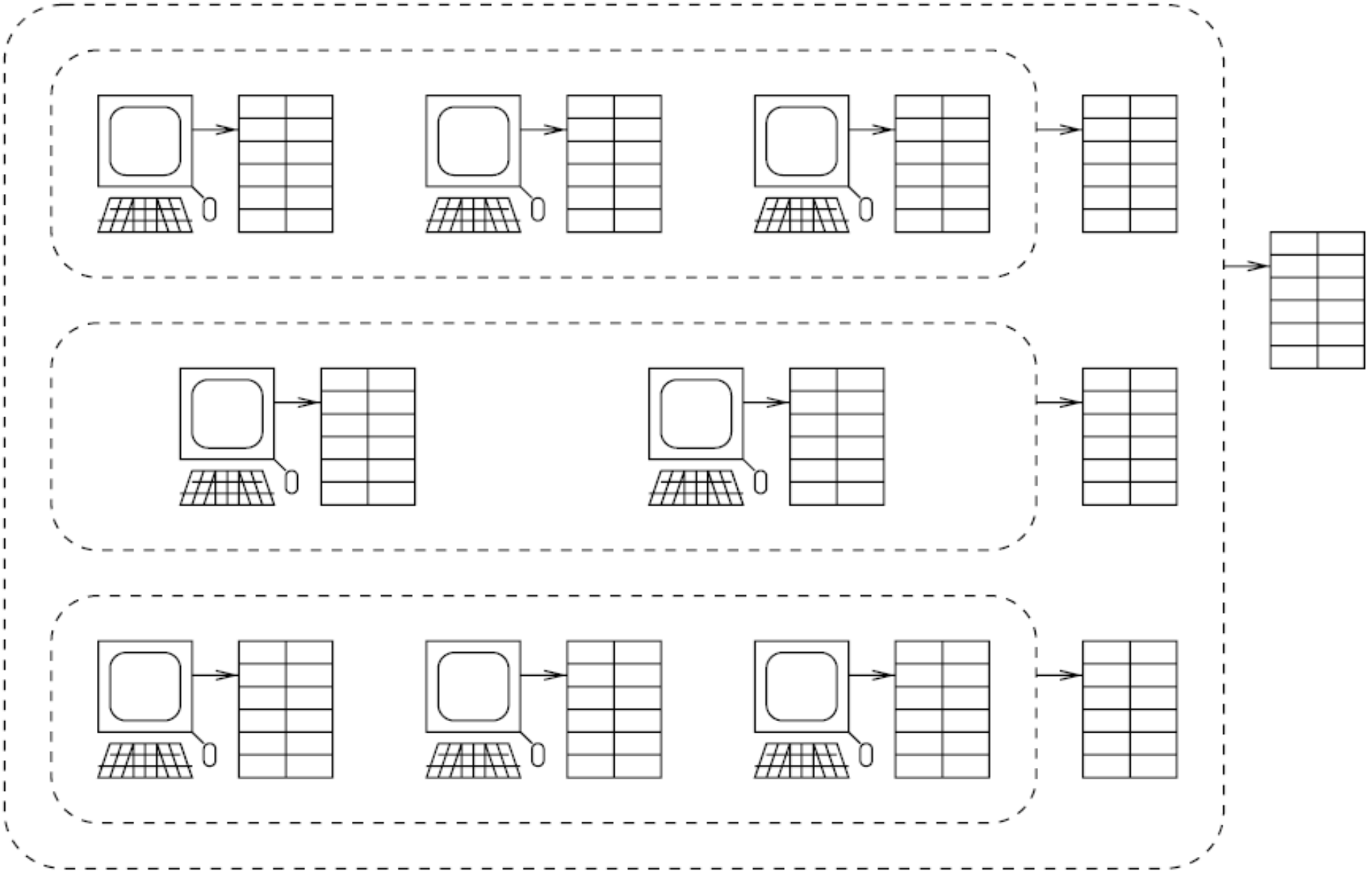
Analogy with DNS

- DNS
  - Directory service based on hierarchical domains
  - Lately more functionality
    - **Round robin DNS, server records, etc**
  - Updates are slow, and vulnerable
- Astrolabe also hierarchical but
  - More efficient
  - More robust
  - More generic
    - **arbitrary info about a domain**
    - **Collected online real time, in a configurable way**

Aggregation as Key Abstraction

- Aggregation is summarizing info
  - Over the entire system or within domains
  - It is of small size (not listing, only summary (O(1)))
- For example
  - Average, maximum, count, etc of some values
- Info is stored in (small) databases: MIBs
  - Management information base
- Aggregation is expressed by a simplified SQL language
- Aggregates are proactively updated at each level

Schematic view of Astrolabe



Astrolabe API

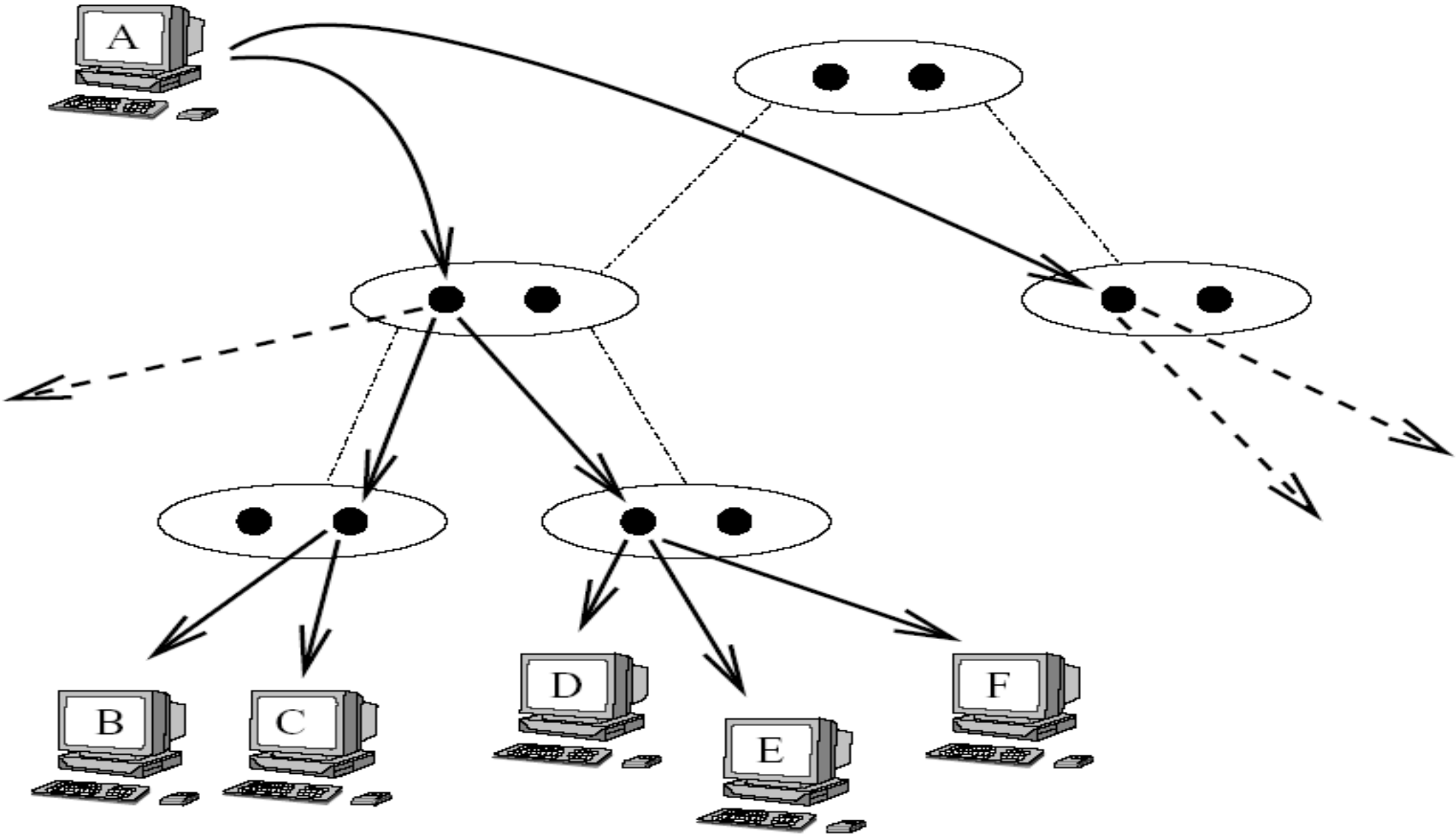
Method	Description
find_contacts(time, scope)	search for Astrolabe agents in the given *scope*
set_contacts(addresses)	specify addresses of initial agents to connect to
get_attributes(zone, event_queue)	report updates to attributes of *zone*
get_children(zone, event_queue)	report updates to zone membership
set_attribute(zone, attribute, value)	update the given attribute

- Can be accessed locally at an Astrolabe host or remotely through RPC
- scope: well defined subset of the tree
- zone: subtree (or leaf)
- updates only on leaf (virtual child zone)

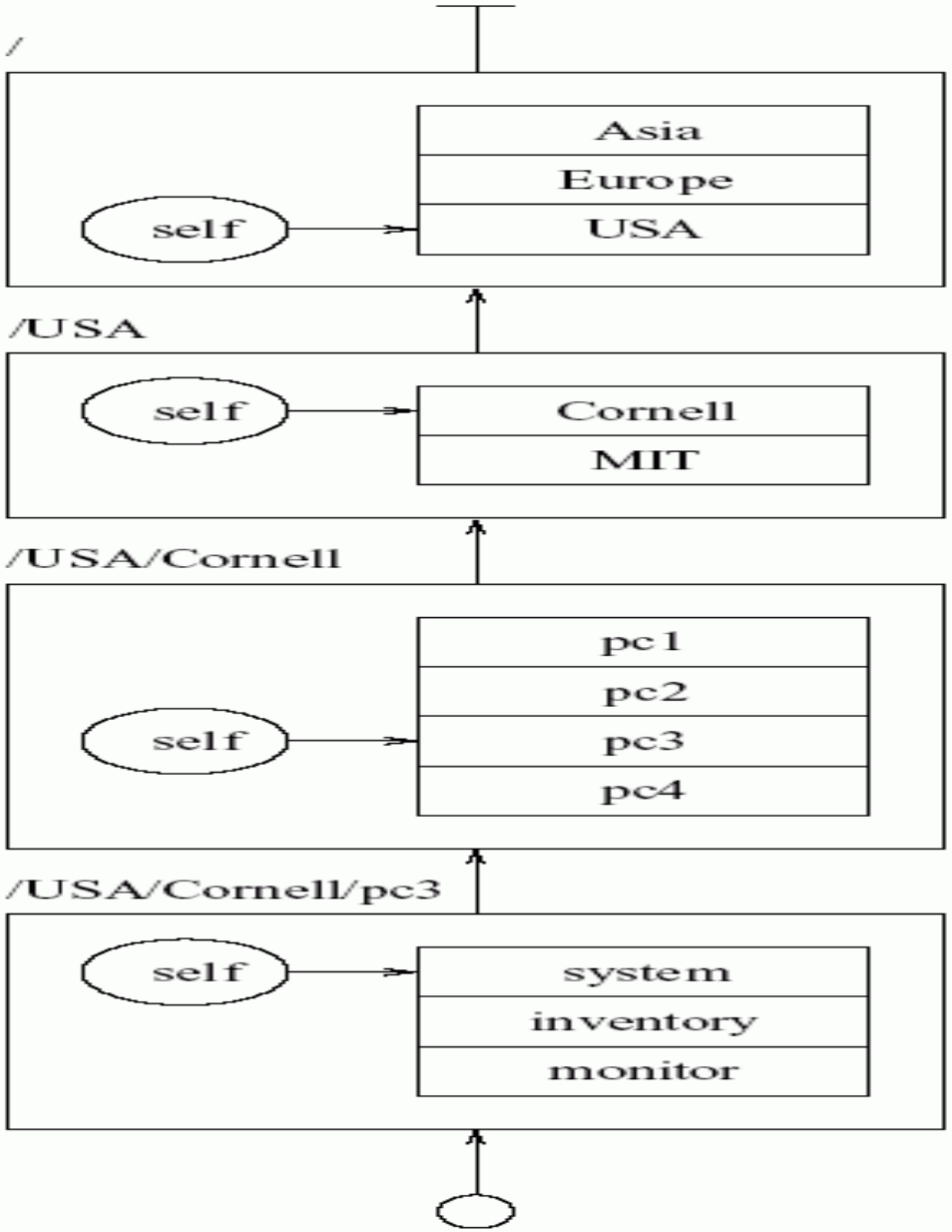
Example Applications

- P2P information diffusion: selectCast
  - Multicast to multicast groups
  - Each zone aggregates members of a group
    - **eg SELECT FIRST(2,game) AS game ORDER BY rate**
  - This way an overlay is superimposed that is used to multicast
  - Having two selected members at each zone allows for redundancy
- Note that the underlying Astrolabe infrastructure takes care of keeping all this up-to-date, scalable and robust

Schematic view of SelectCast



Implementation



- Each agent maintains a copy of the chain MIBs up to the root
- It also replicates the MIBs of all child zones of all the zones in this chain
- So zones are purely virtual and are replicated over all members

Compulsory Attributes

- ID
  - the local zone name within the parent zone
- Issued
  - the timestamp of last update of this MIB
- Contacts
  - Representatives for this zone (who will gossip)
- Nmembers
  - Number of members in the zone
- Servers
  - Small set of agents that implement the API

Gossip

- This set of MIBs is replicated (refreshed) through gossip
- For all zones separately
  - There is a gossip rate (cycle length)
  - Contacts for a zone pick a sibling zone at random
  - Initiate gossip with a contact of the selected zone
  - They run an anti-entropy step (regarding their own level and up)
- Note that most communication is done between sibling leaf nodes

Other issues

- Membership management
  - If a given zone's MIB is not refreshed for some time, it is removed
  - Joins are dealt with
    - **Setting a contact node explicitly**
    - **Or doing IP broadcast, etc**
- Communication
  - Issues with firewalls
    - **Application level gateways (ALGs), etc**
- Security
  - Through certificates
    - **Each zone has a certificate authority (CA)**

References

- Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining. ACM Transactions on Computer Systems, 21(2):164–206, May 2003. (doi:10.1145/762483.762485)
- Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing (PODC'87), pages 1–12, Vancouver, British Columbia, Canada, August 1987. ACM Press.