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)
  - phenomena such as forest fires, branching processes and diffusion are all similar mathematically
- extremely simple locally, powerful and robust globally
- In computer science, epidemics are relevant
  - for security (against worms and viruses)
  - for designing useful protocols (we look at this here)

Outline

- Information dissemination
  - Brief intro to seminal work by Demers *et al* (1987), that first coined the term gossip and epidemic protocols
  - point is: gossiping is simple, fast, and robust
- Generalizations of gossip protocols for
  - peer sampling
  - topology maintenance
  - data aggregation
  - modular architectures
- Problems, directions

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: emailing updates
  - problems with detecting and correcting errors (done by hand!)
  - bottleneck with the originating (updated) site
  - not scalable (slow if very large number of nodes)
  - (message complexity quite good though!)

Gossip/epidemics to the rescue

- spread information using gossip
  - all nodes periodically contact random other nodes and exchange information (spread updates)
- SI model (from epidemiology)
  - susceptible: has not received the update
  - infected: received the update (can “infect”)
  - nodes are initially susceptible, and become infected; no other states, no turning back, spreading until all infected
- SIR model
  - removed: infected nodes eventually stop being infective: spreading can stop before all infected

Some Properties of the SI model

- the push model
  - N nodes communicate in rounds (cycles)
  - in each cycle, a node that has the update (infected) sends it to a random other node, that becomes infected too
- “Anti-entropy” for database updates
  - nodes send the (hash of) the entire database (not only a single update)
    - **as a side effect, all new updates are spread according to the SI model**
  - receiving nodes update their own database via merging the unseen updates

Mean-field model of push SI

- Let p_i be the proportion of *not* infected nodes in cycle i
- $1-p_0=1/N$
- Pittel (1987) shows that the model below is quite accurate for predicting p_i

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

Speed and cost of push SI

- Let S_N be the first cycle where $p_i=0$
- Pittel (1987) shows that in probability

$$S_N = \log(N) + \ln(N) + O(1)$$

- This is quite fast...
- But the number of overall messages sent is

$$O(N \log N)$$

Pull and push-pull SI

- With pull, we have

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

- This is *very* fast when p_i is small (end phase)...
- Karp *et al* (2000) show that the number of overall messages sent with push-pull is

$$O(N \log\log N)$$

- But termination is trickier when no updates are available (for anti-entropy does not matter)

SIR for spreading single updates

- For anti-entropy, use a pull or push-pull SI modell
- For the spreading of updates, the termination problem needs to be addressed: rumor mongering with SIR model
- Push approach
  - when a rumor (update) becomes “cold”, stop pushing
- Pull approach
  - same as push, only stop offering update when pulled when it becomes cold

A note on random networks

- Note that gossiping nodes pick another node in each cycle: they do not need to know all the nodes
- The actual communication pattern defines a random graph
  - by looking at these graphs, we can understand the properties of the communication better
  - we can design better gossip protocols if we understand the implications of our design decisions

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, 41(5), 2007.
- Boris Pittel. On spreading a rumor. SIAM Journal on Applied Mathematics, Vol. 47, No. 1 (Feb., 1987), pp. 213-223
- R Karp, C Schindelhauer, S Shenker, B Vocking. Randomized Rumor Spreading. FOCS 2000.

A Gossip Skeleton

- Originally for information dissemination in a very simple but efficient and reliable way
- Later the gossip approach has been generalized resulting in 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

Rumor mongering as an instance

- state: set of active 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
- propagation of one given update can be limited (max k times or with some probability, as we have seen, etc)

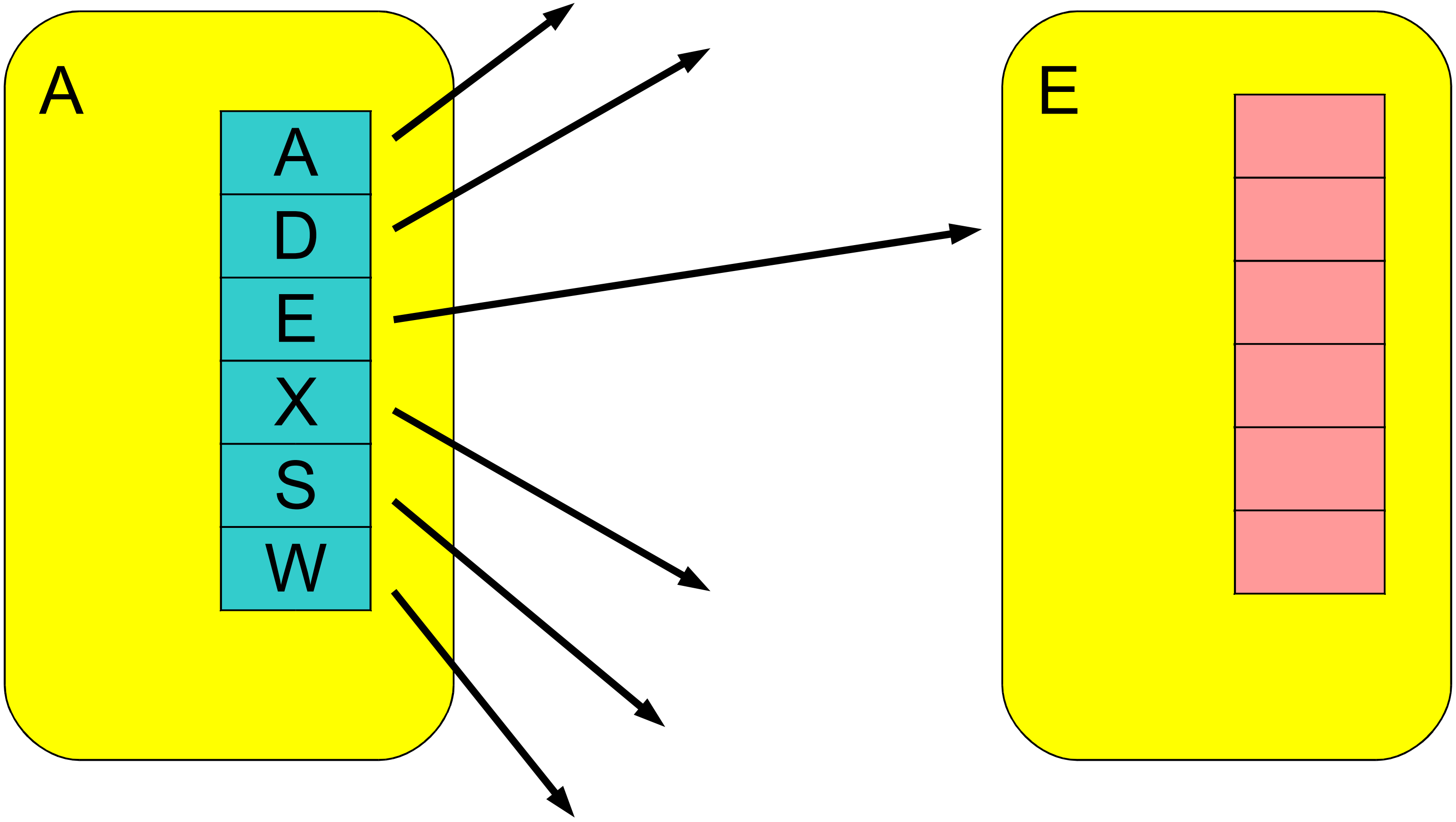
Peer Sampling

- A key method is selectPeer in all gossip protocols (influences 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

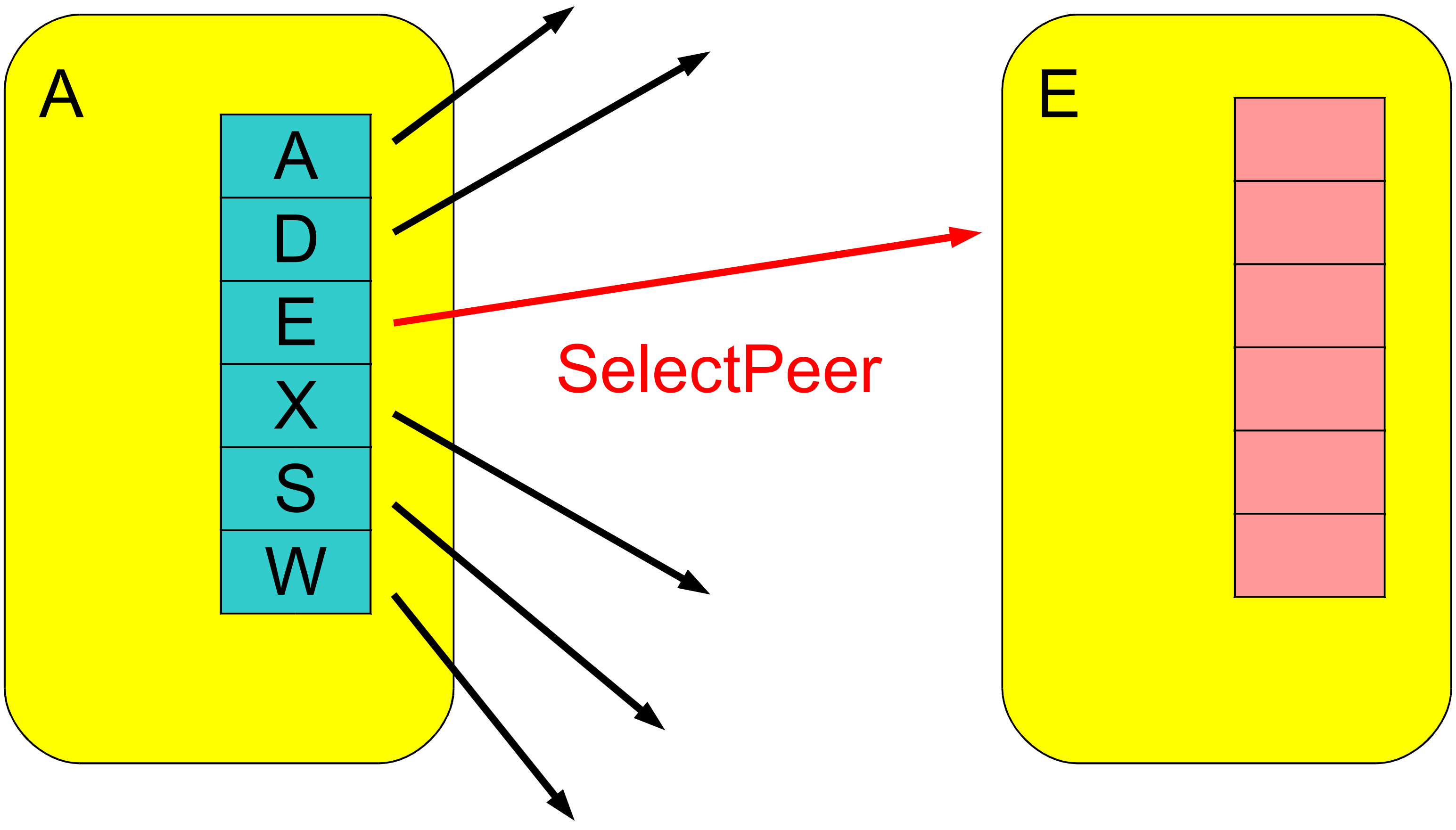
Gossip based peer sampling

- 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

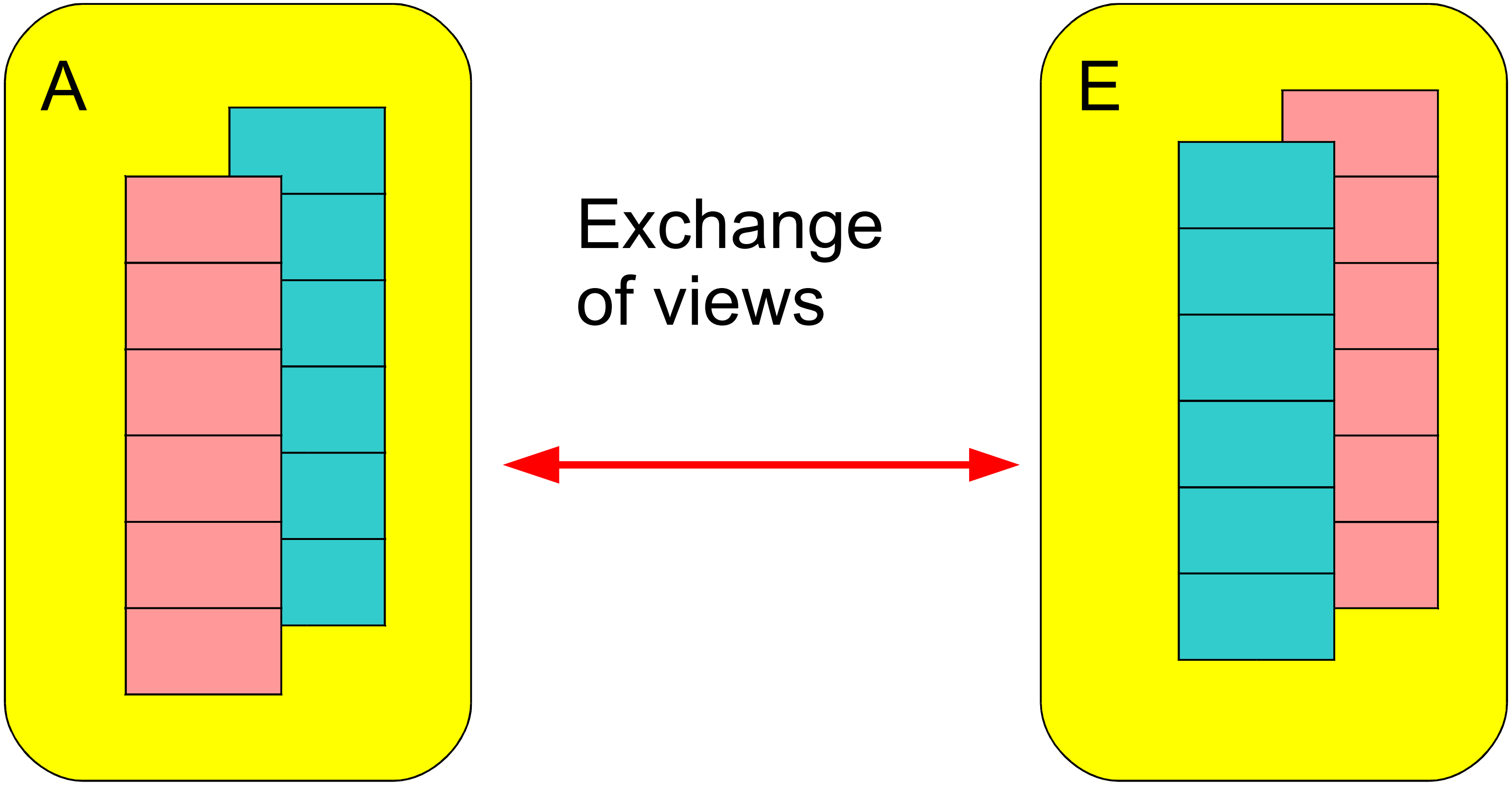
Gossip protocols for topology management



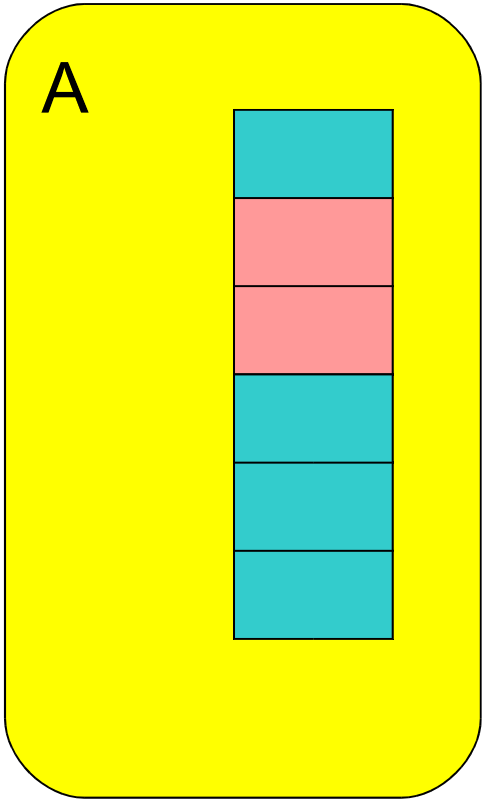
Gossip protocols for topology management



Gossip protocols for topology management

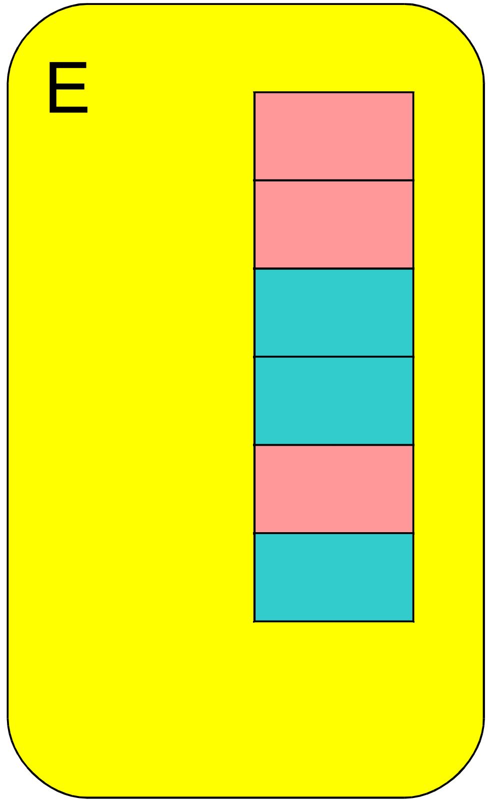


Gossip protocols for topology management



Both sides apply update

thereby redefining topology



References

- Allavena, A., Demers, A., and Hopcroft, J. E. 2005. Correctness of a gossip based membership protocol. In Proceedings of the 24th annual ACM symposium on principles of distributed computing (PODC 05). ACM Press, Las Vegas, Nevada, USA.
- 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

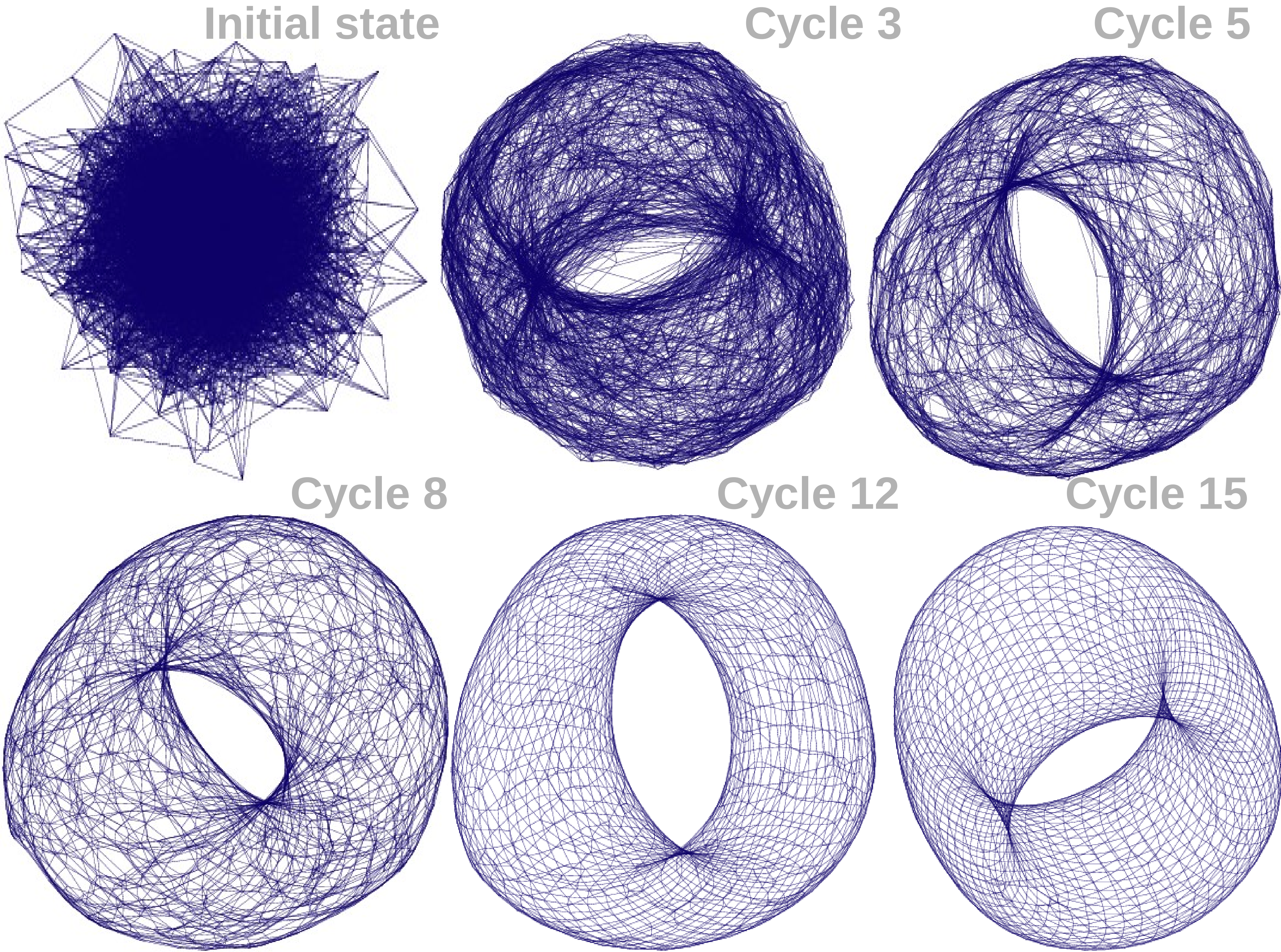
- We saw we can build random networks. Can we build any network with gossip?
- Yes, many examples
  - proximity networks
  - DHT-s (Bamboo DHT: maintains Pastry structure with gossip inspired protocols)
  - semantic proximity networks
  - etc

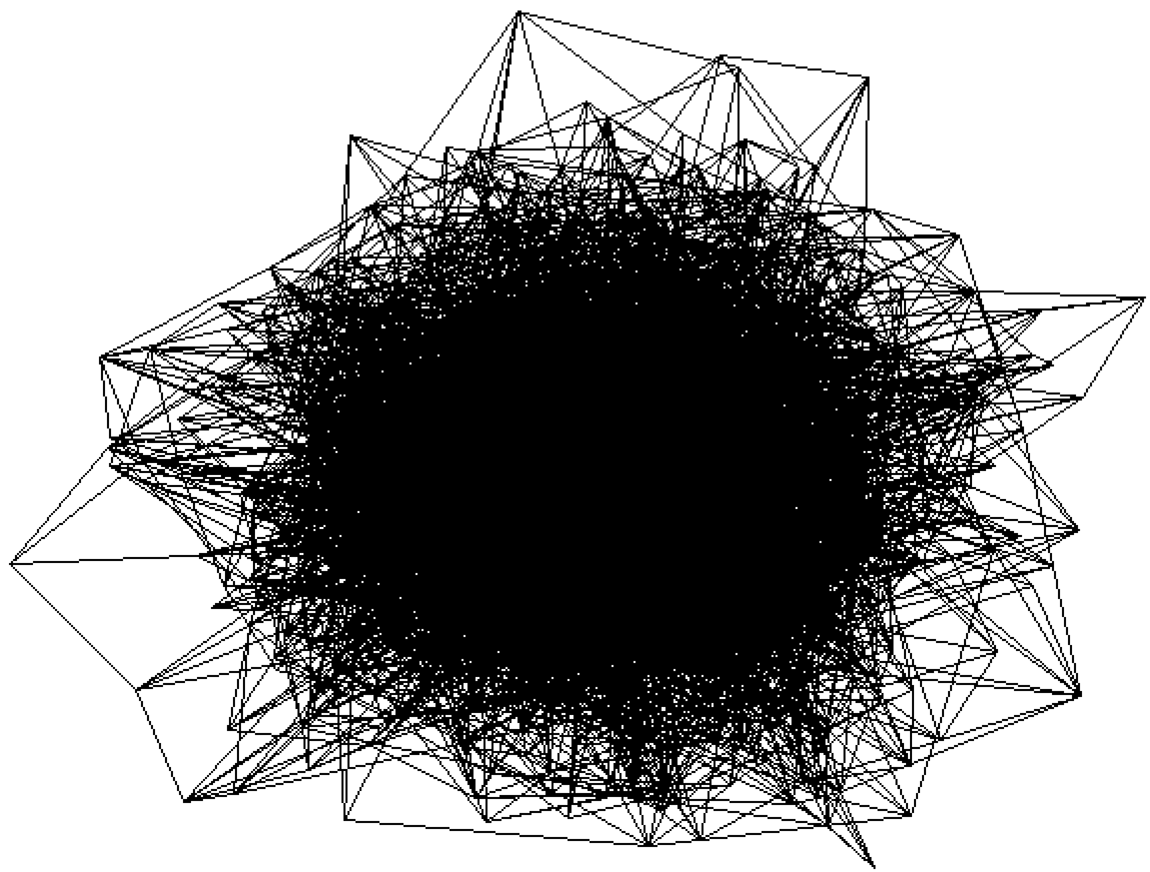
T-Man

- 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
- 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

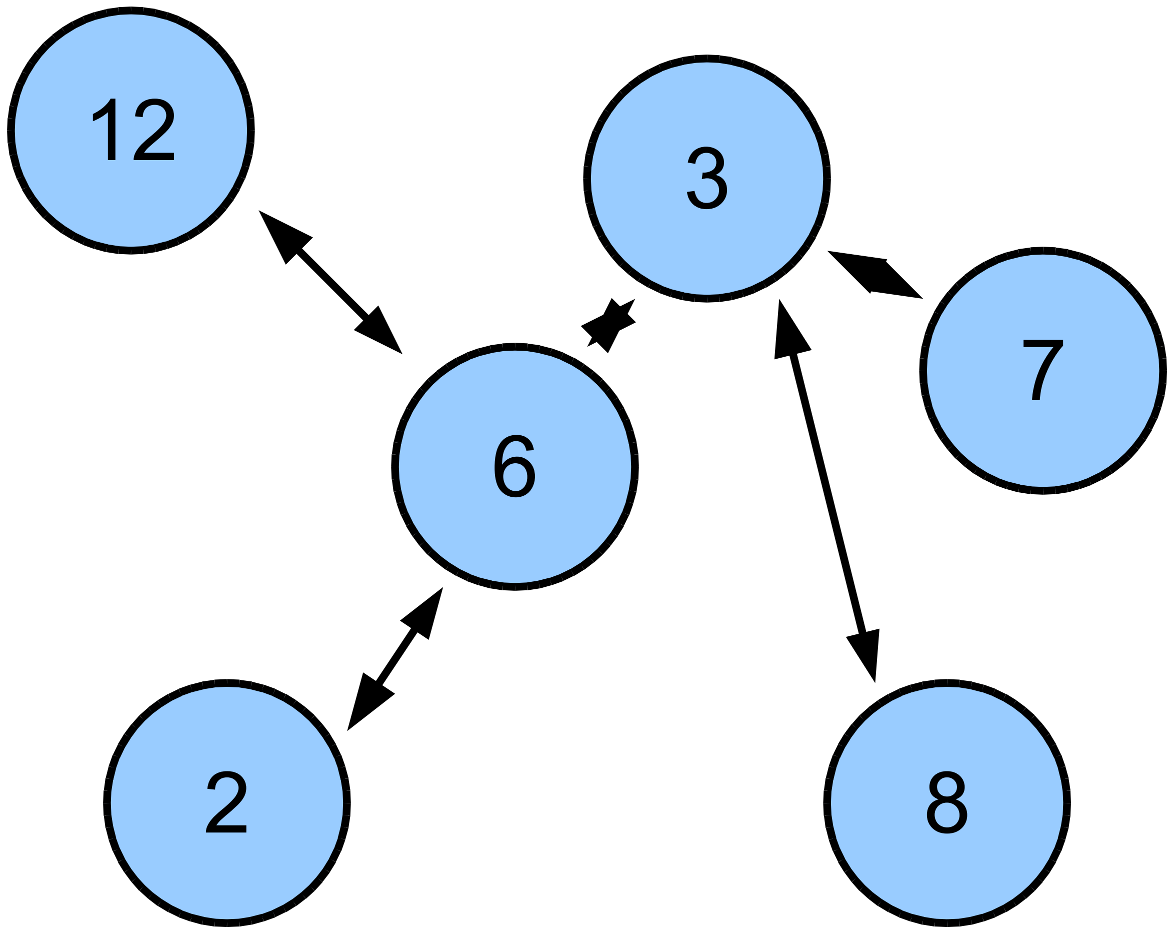
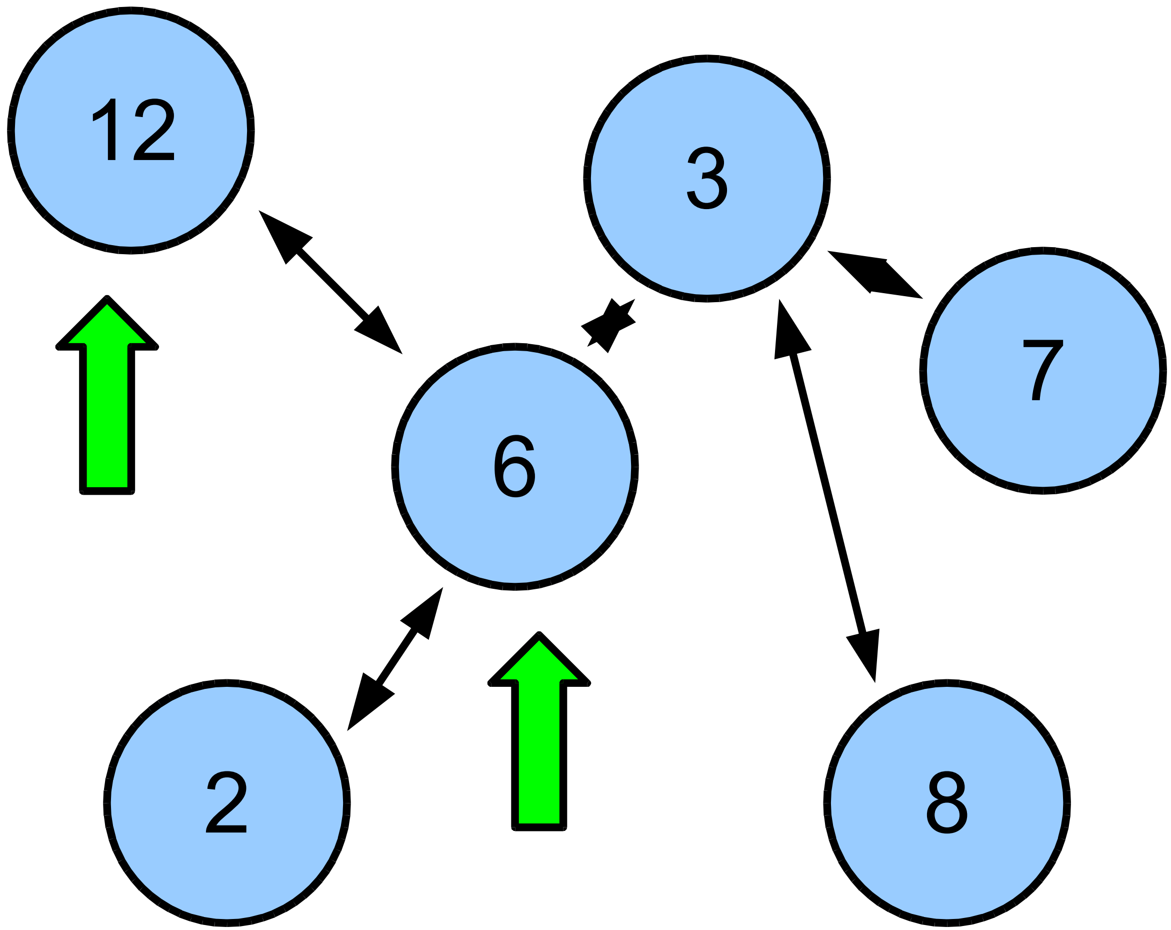
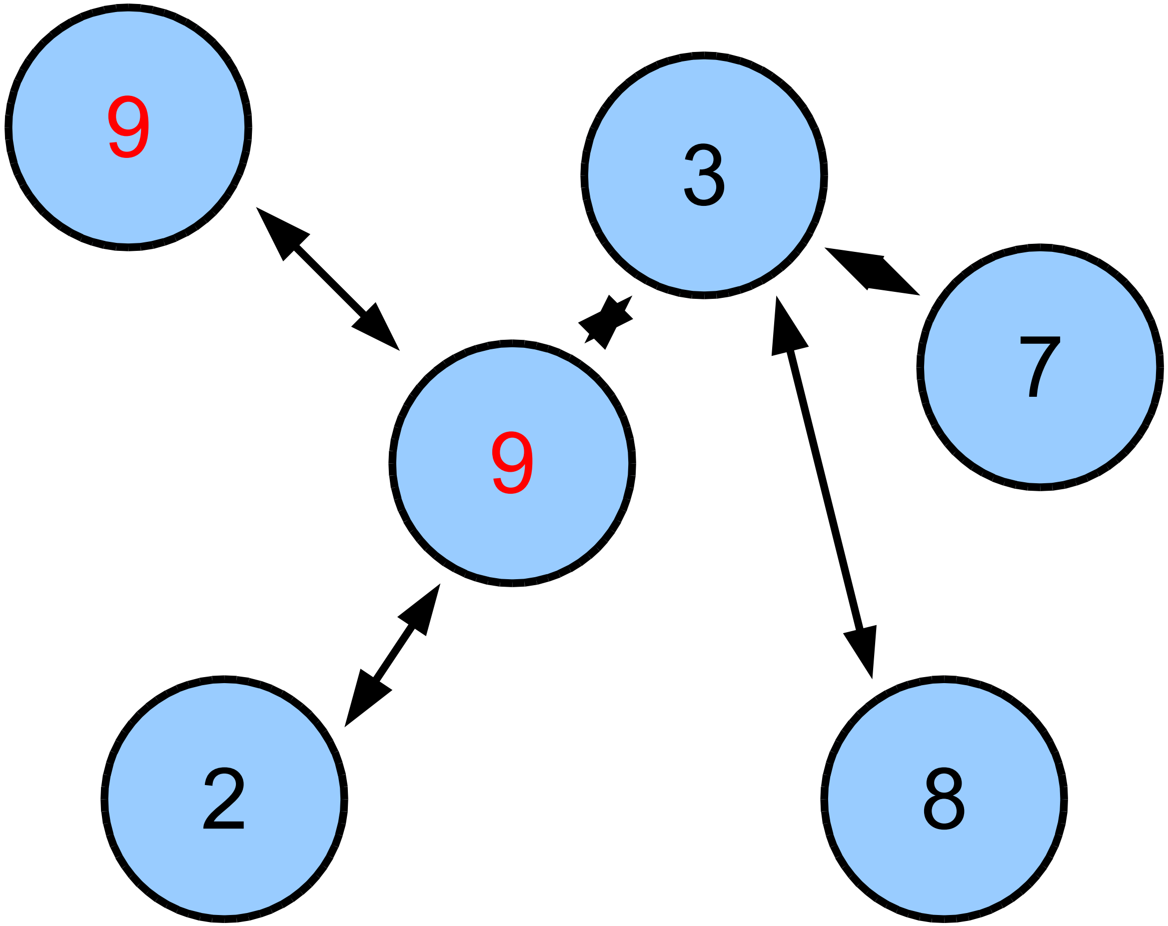


Illustration of averaging



(12+6)/2=9

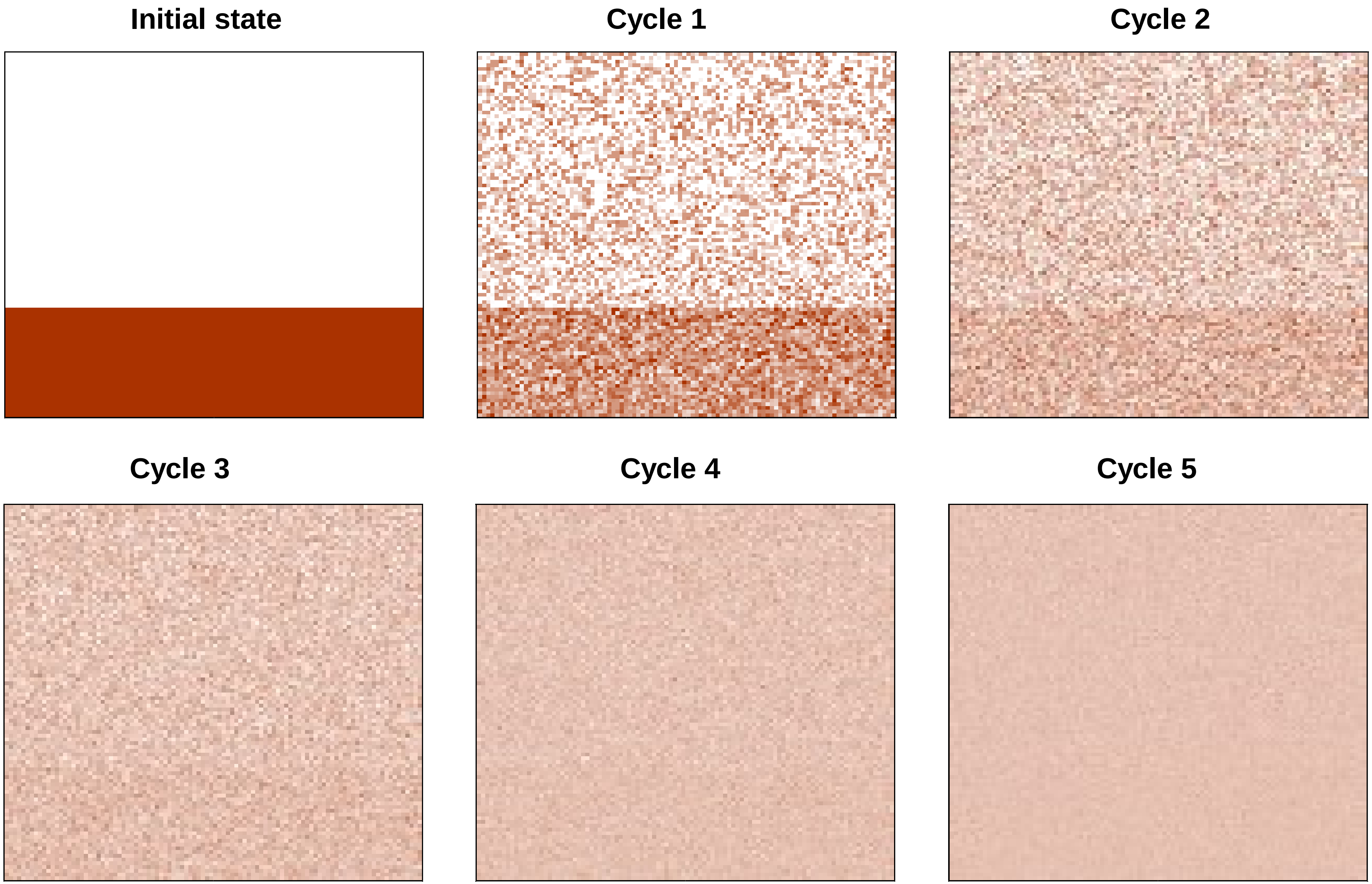
Illustration of averaging



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



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.

Some other examples

- firefly-inspired synchronization
- partitioning (slicing) and sorting in P2P networks
- asynchronous implementation of matrix iterations
  - ranking (PageRank)
  - reputation systems
- emergent cooperation

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

A slide on viruses and worms

- We focused on “good” epidemics but malicious applications are known
  - viruses and worms replicate themselves via similar algorithms using some underlying network such as email contacts or the Internet itself
- The dynamics is described by SIS model
- Underlying networks are typically scale free (power law degree distribution)
  - can be proven: no threshold: it is nearly impossible to completely eliminate a “disease”

Some open problems

- gossip in mobile contact networks and its potential applications (also malware...)
- security
  - gossip is robust to benign failure but very sensitive to malicious attacks
  - current “secure” gossip protocols sacrifice simplicity and light-weight
- interdisciplinary connections: toward a deeper understanding of self-organization and gossip protocols as a special case