A case study on gossip beyond gossip: Sorting

Márk Jelasity
MTA and University of Szeged, Hungary

Outline

- Generalizing gossip
- Example non-dissemination applications
  - Peer sampling
  - Topology management, eg sorted ring
- Sorting data along a sorted ring
  - Algorithm description
  - Some results

Generalizing gossip

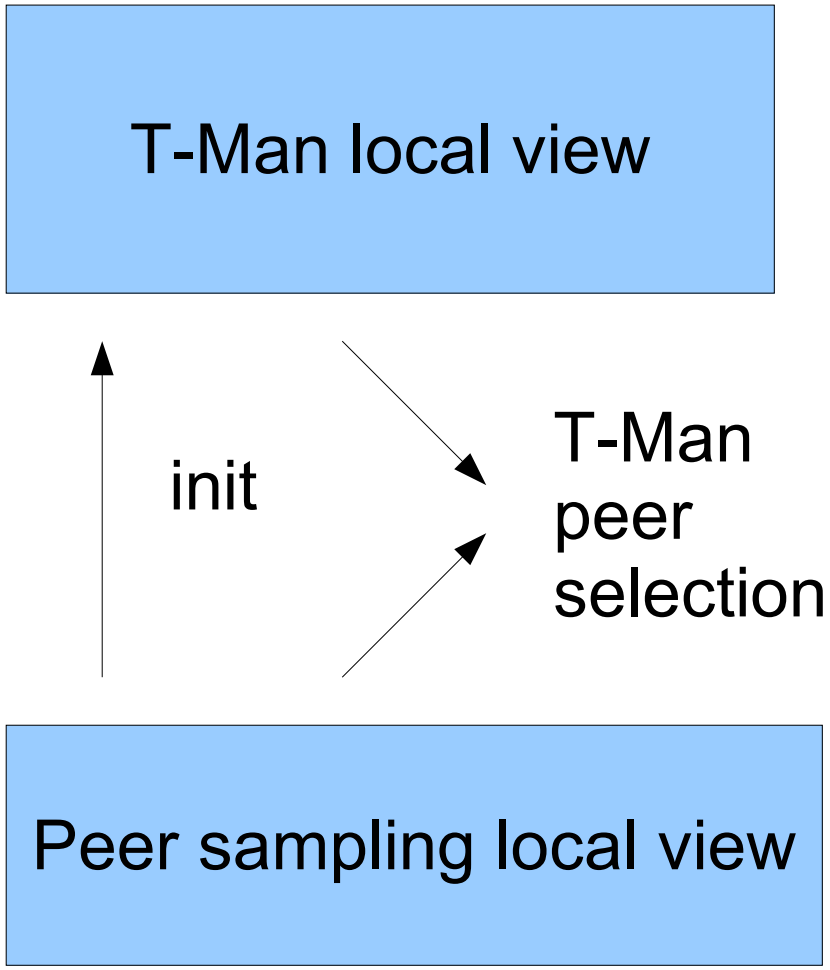
- Gossip: periodically do the following
  - Select peer to gossip with
  - Send some information to peer (push)
  - Optionally receive information from peer (pull)
  - Process new information (update own state)
- Dissemination
  - Peer is selected at random
  - Information is an update or news
  - Processing is simple storage

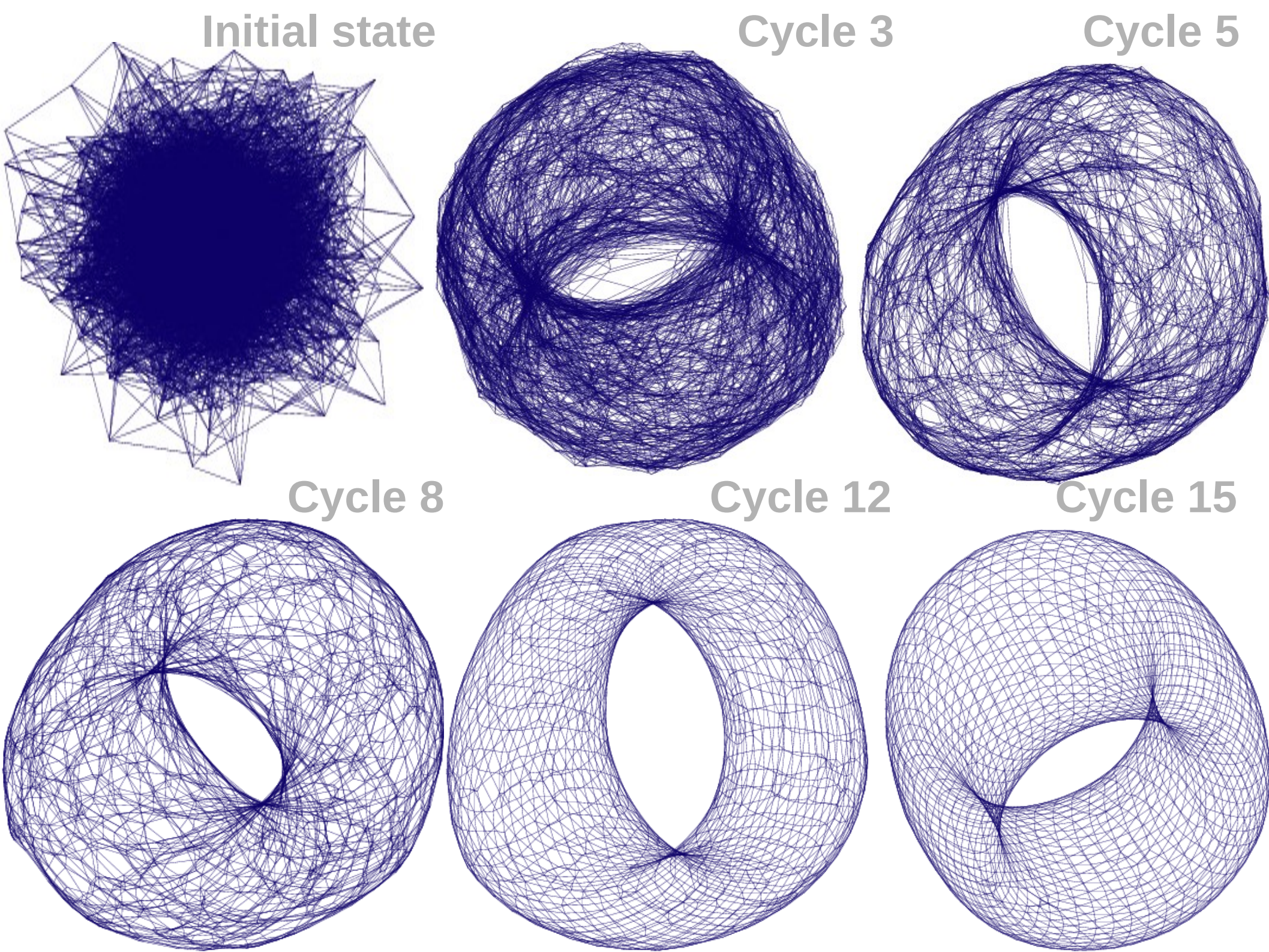
Peer sampling

- Peer sampling service
  - Peer is selected from local partial view
  - Information to gossip is the partial view itself
  - Processing is creating the new partial view
- Topology management (T-Man)
  - Peer is selected from partial view that defines the topology of the overlay we are building (T-Man view)
  - Information is a subset of T-Man view
  - Processing is creating the new partial view: bias towards “close” peers creates a wide range of topologies

“layering” gossip protocols

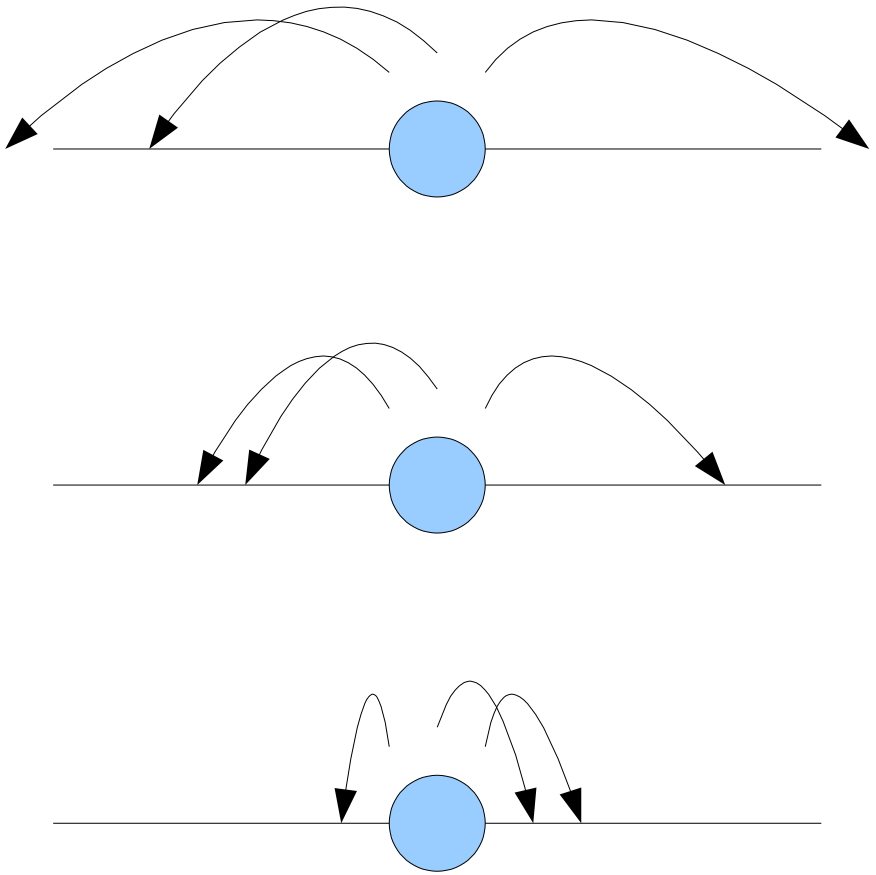
- Gossip protocols at a node can use each other's state (local view) for peer selection and perhaps other things too
- For example
  - The local view of the peer sampling protocol contains random samples: huge number of applications (among others, topology building)
- Much more complicated scenarios work too




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

Sorting

- Sorted topology
  - Generic overlay builder T-Man can create ring sorted by ID
  - Starts with random topology
  - View gradually converges to right neighborhood, while it is exponentially "shrinking"

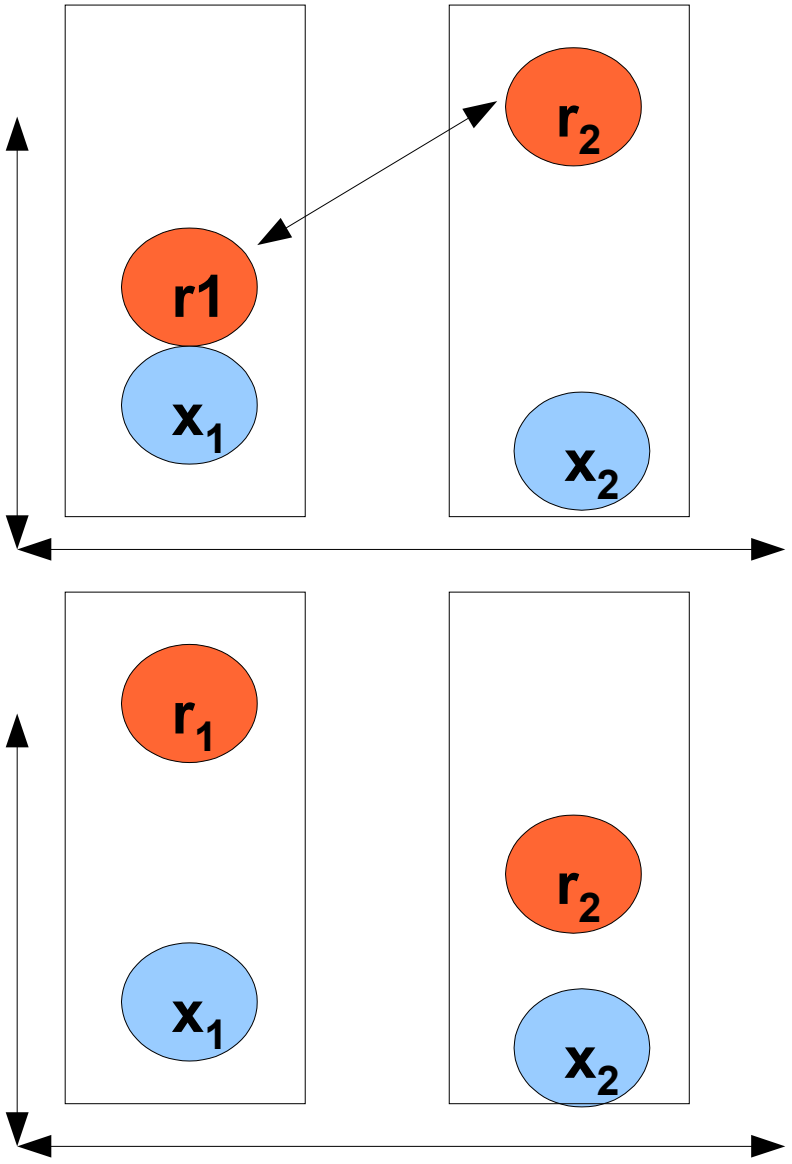


Sorting

- Another sorting problem
  - Sort data along some node ID-s (or any node attribute)
  - Analogous to sorting an array
    - **Array cell index is the index along which we sort**
    - **But in our case the index can be arbitrary**
  - Note that no overlay is needed by this problem statement
- Approach
  - Three layer approach: use T-Man to build a sorted ring using the ID-s and in parallel sort the data too

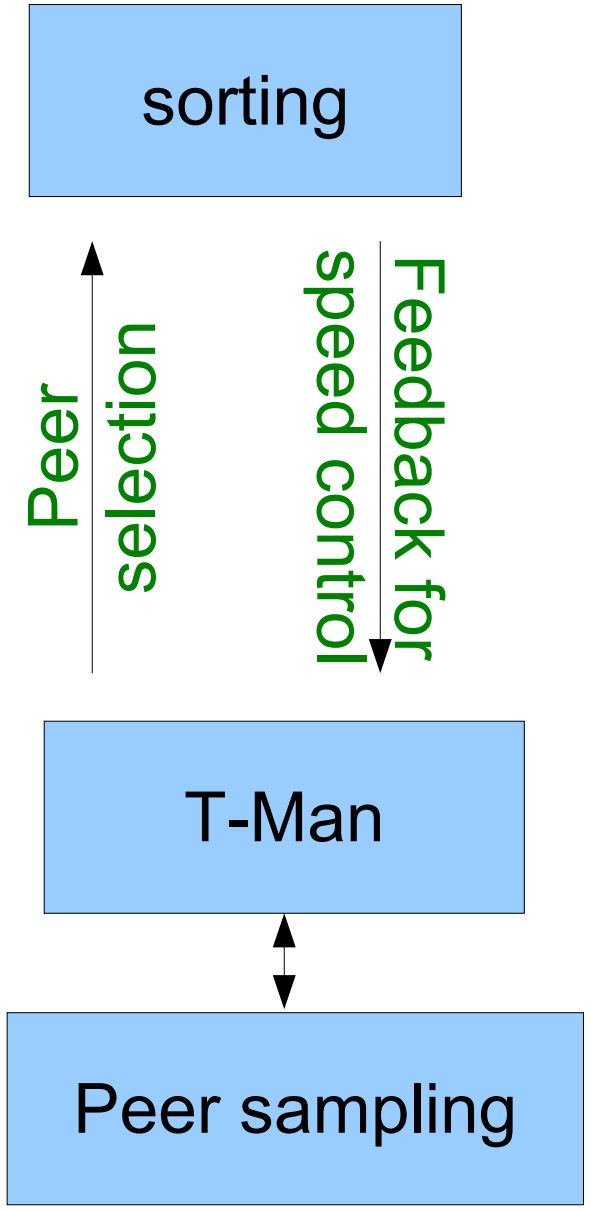
Main idea

- Select peers from shrinking T-Man view
- When possible, swap values
  - Possible when index is smaller and value is larger or vice versa



Three gossip layers

- Three gossip protocols cooperate
  - Usual T-Man + peer sampling unit
  - Sorting uses T-Man for peer selection
  - Sorting controls convergence speed of T-Man to get the right pace of shrinking
    - **T-Man gossips only when sorting no longer finds good peers**



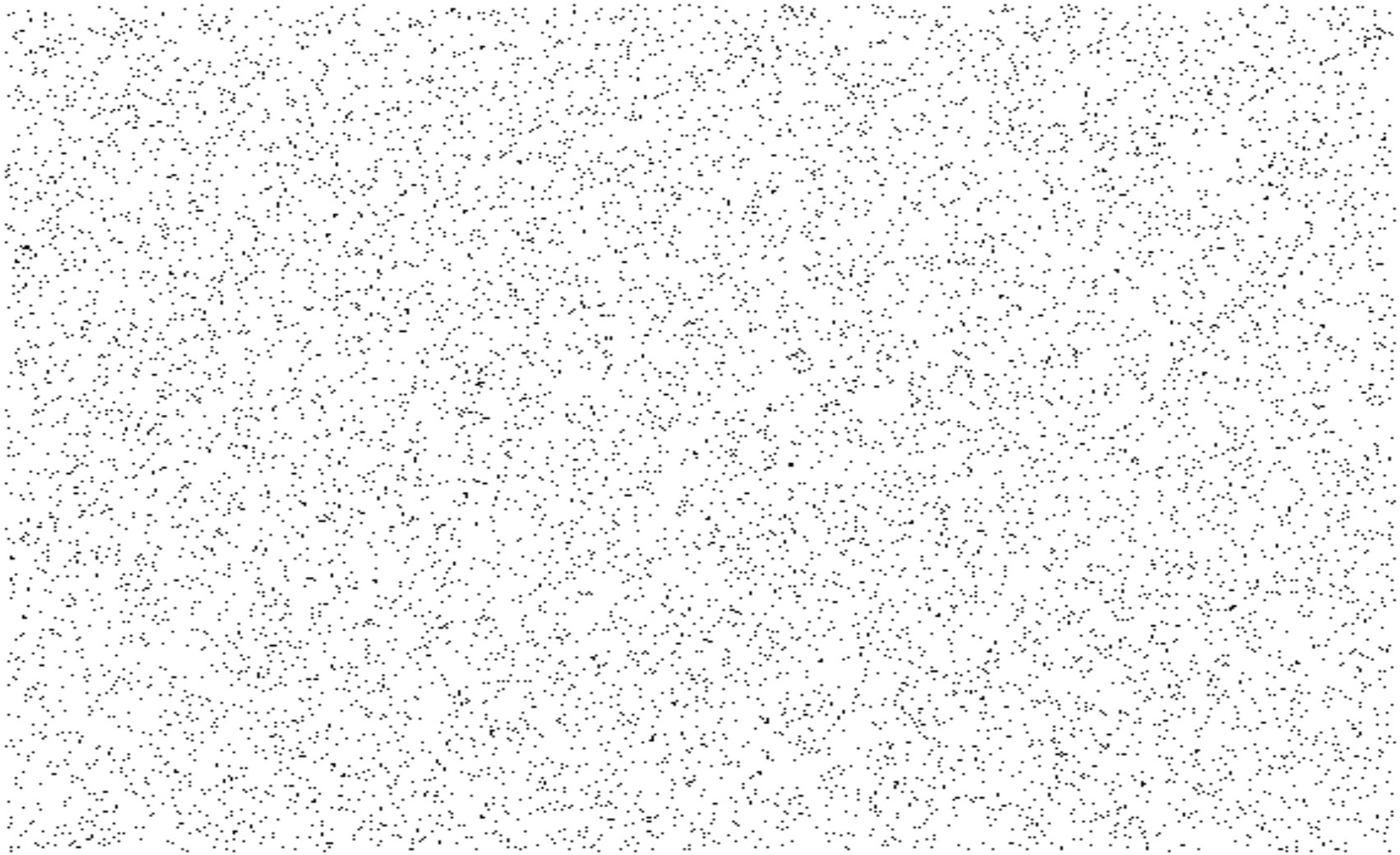
Preliminary simulator-implementation and model

- All protocols start at once, “more or less” in synch
- No failure and concurrency is implemented, but
  - No bottlenecks and hot spots can occur due to ideal load balancing
  - We already know that at least peer sampling and T-Man is modeled closely enough; so very likely sorting too
- In case of node or message failure or long delay we can loose data even with reliable transport
  - In target applications (data mining, self-organization) it is often tolerated to some extent

Parameters

- Peer sampling view size is 20, message size also 20
- T-Man view size is 100, message size 10
- In one gossip round, sorting takes the 20 entries from the T-Man view that are closest in the forming ring
  - Probes them in a random order
  - If a suitable peer is found, an exchange is performed and rest of the 20 is not probed (often 2nd , 3rd try works)
  - If no peer is found, a T-Man gossip step is performed to get new entries in the view

Animation with N=10,000

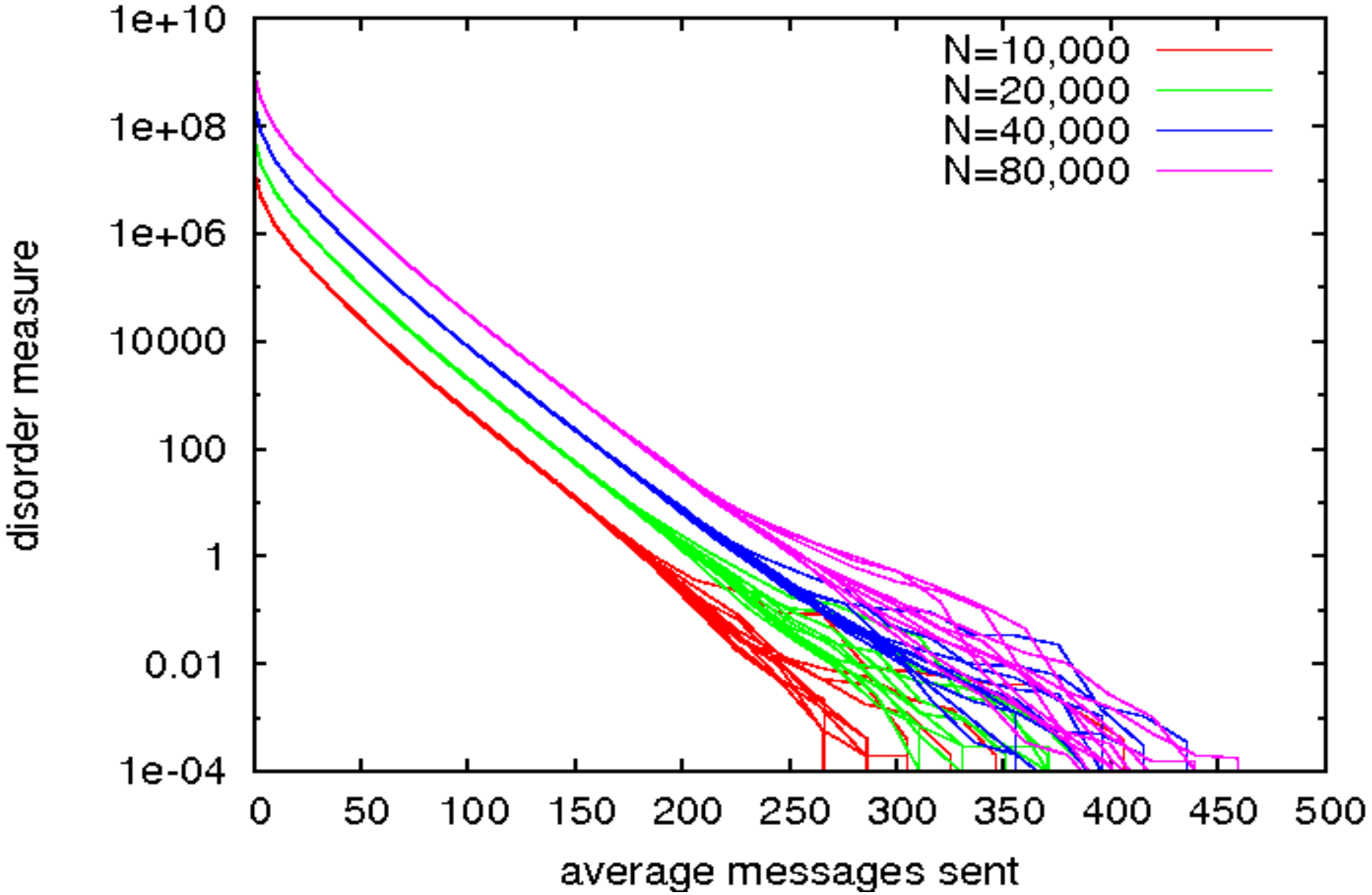


Measure of performance

- A variance like measure to characterize goodness of sorting

$$\sigma(t) = \frac{1}{N} \sum_{j=1}^{N} (j - i_j(t))^2$$

Logarithmic speed



Summary

- Experimentally O(log N) speed to achieve perfect sorting
  - Very simple
  - In the lack of failure potentially competitive to sorting networks (constant within O can most likely be reduced significantly)
  - In the presence of failure potentially graceful degradation
- Remotely similar to Shell sort, only probabilistic
- Lots of open questions