Sistemas Distribuídos

José Orlando Pereira

Departamento de Informática
Universidade do Minho

2018/2019

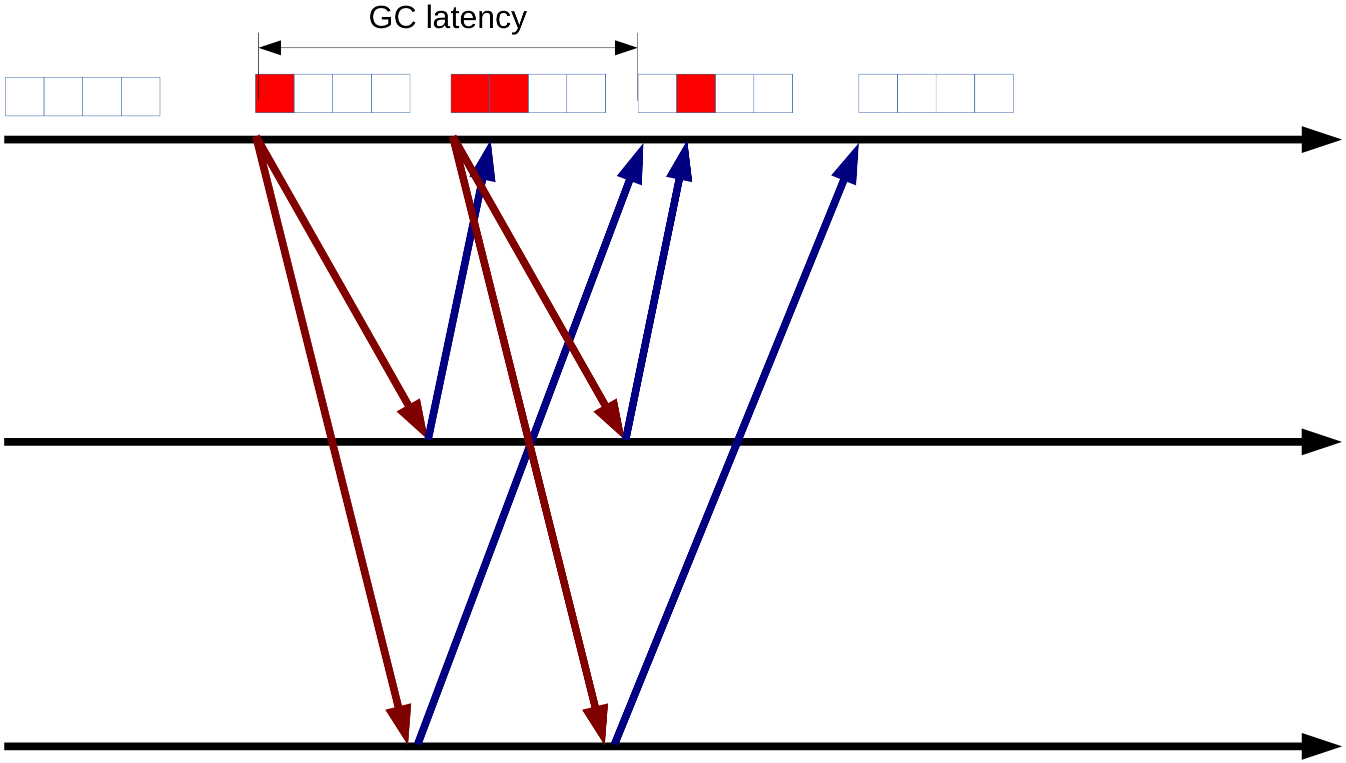
Application Level Multicast

- Send to multiple destinations: group
- Reliability:
  - Deliver all messages sent

    vs
  - All deliver the same messages (agreement)

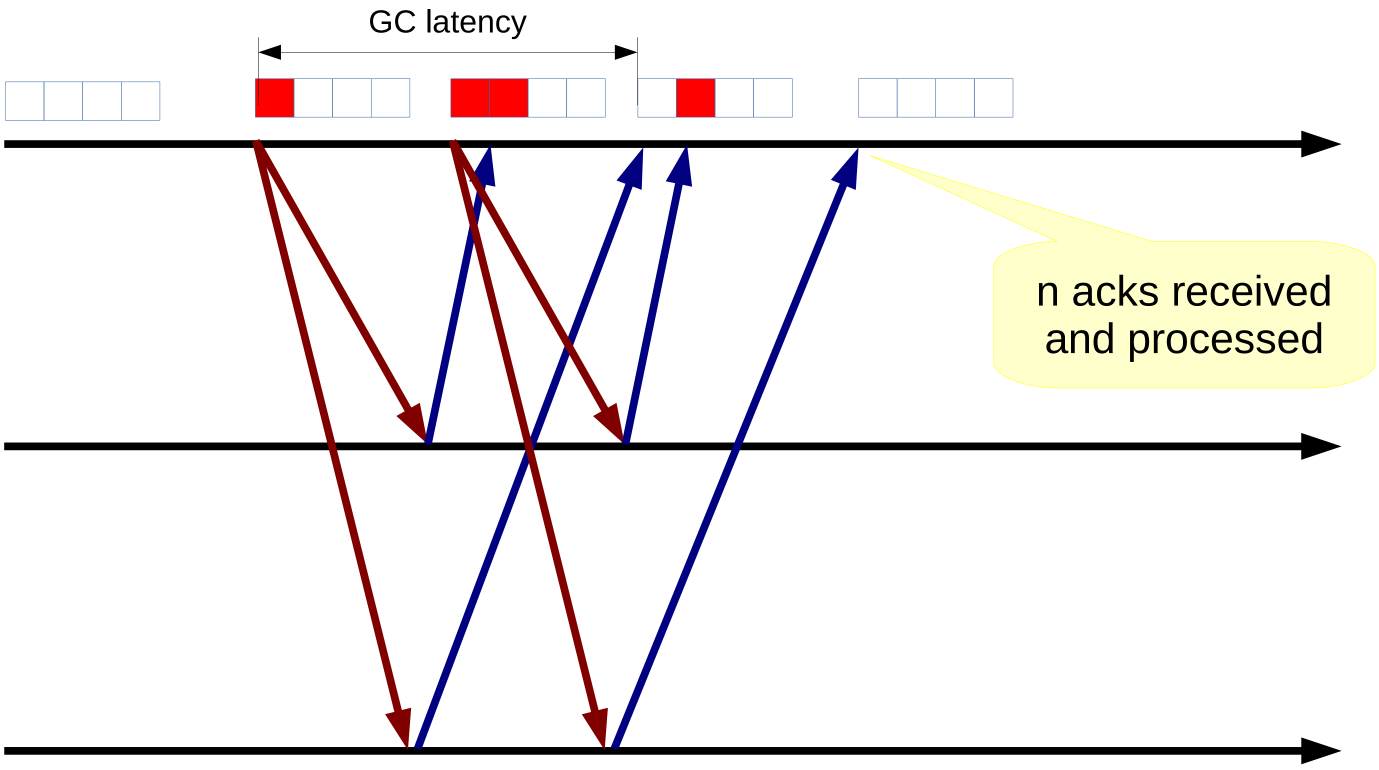
General approach

- Buffer and retransmit until acknowledged



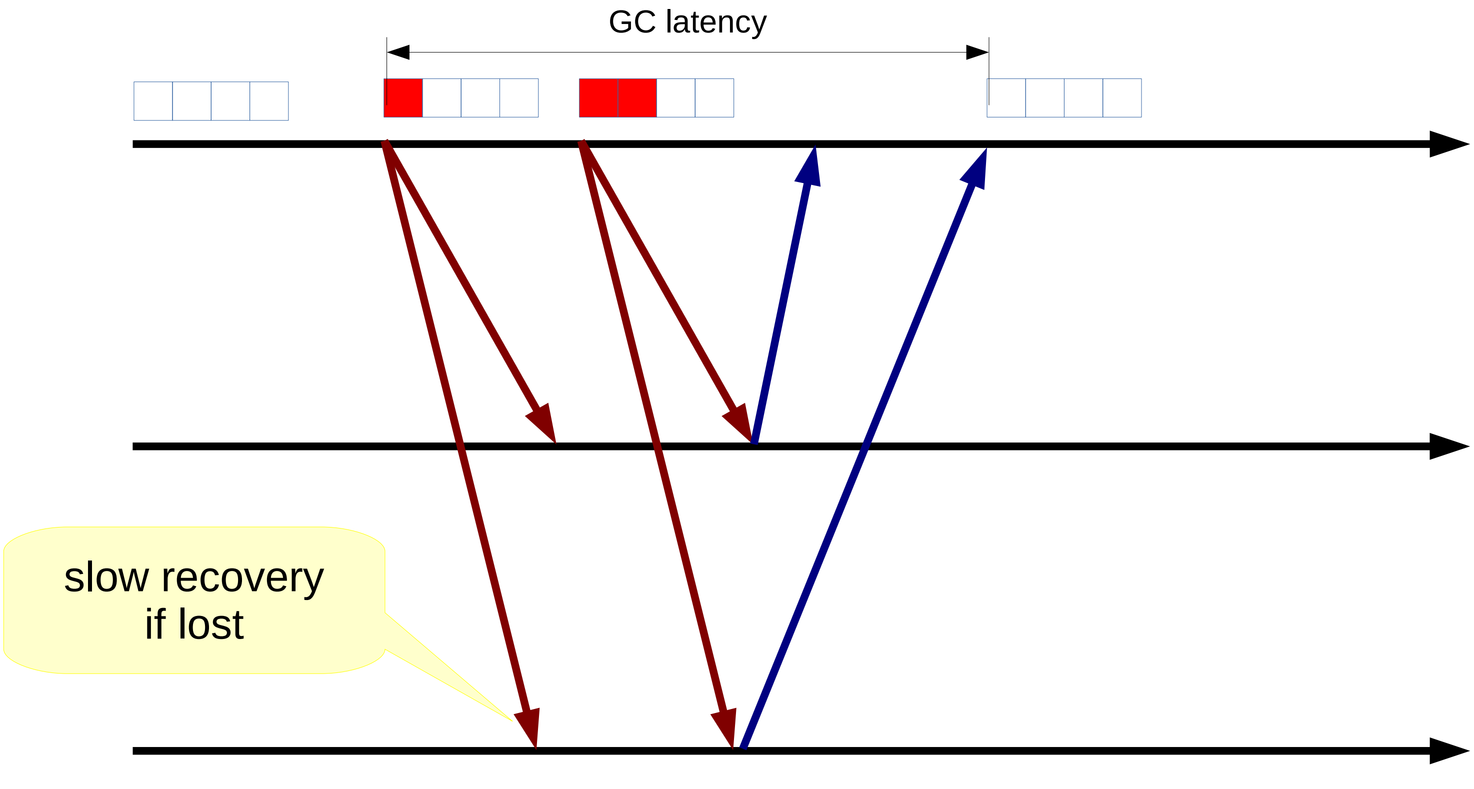
Acknowledgments

- Not scalable to large number of destinations due to “ack implosion”:



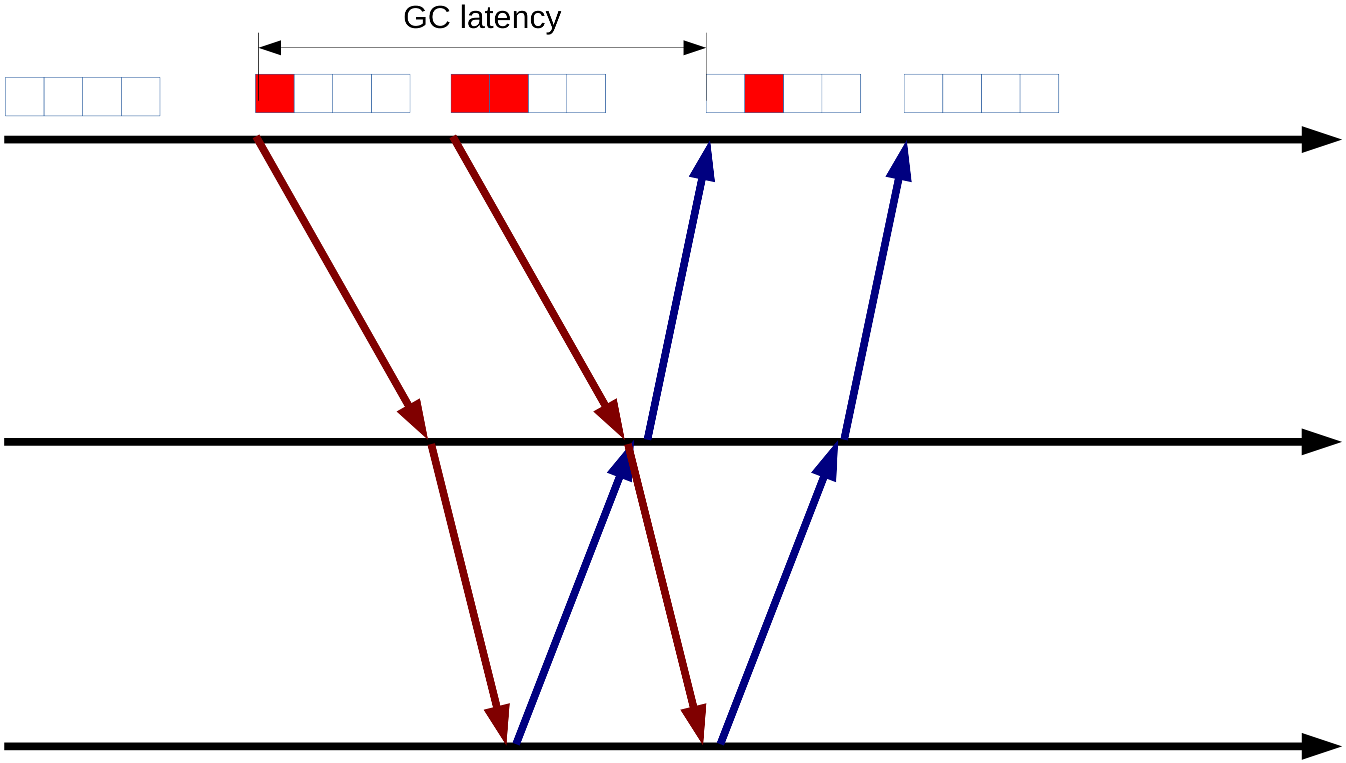
Group acknowledgment

- Not scalable to high throughput due to large buffer requirements and slow recovery:



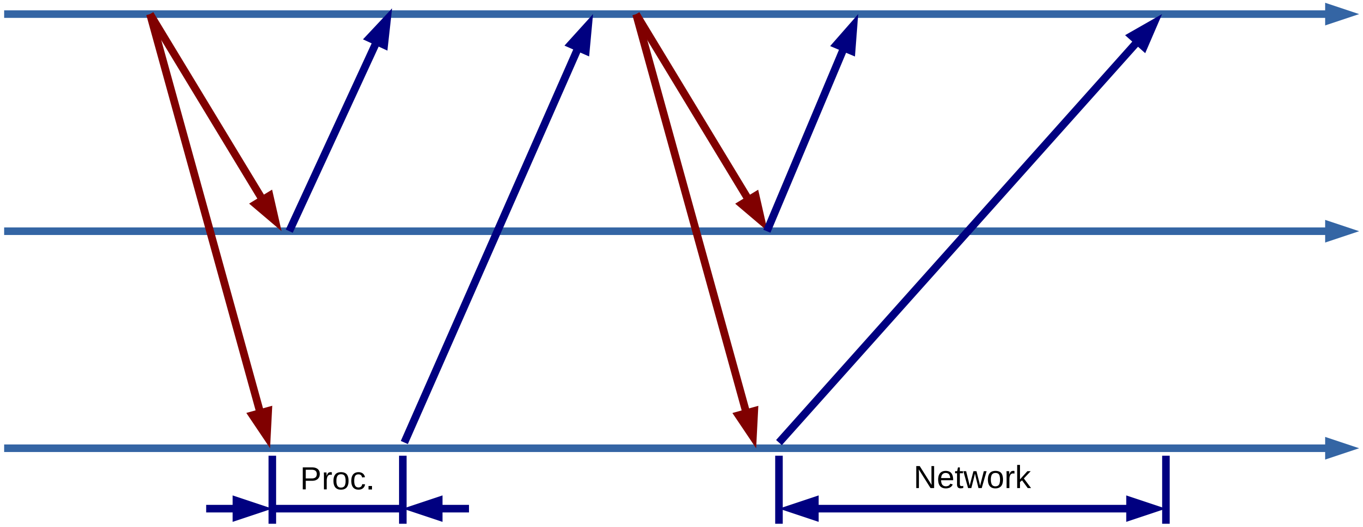
Structured multicast

- Avoids “ack implosion” but increases GC latency:



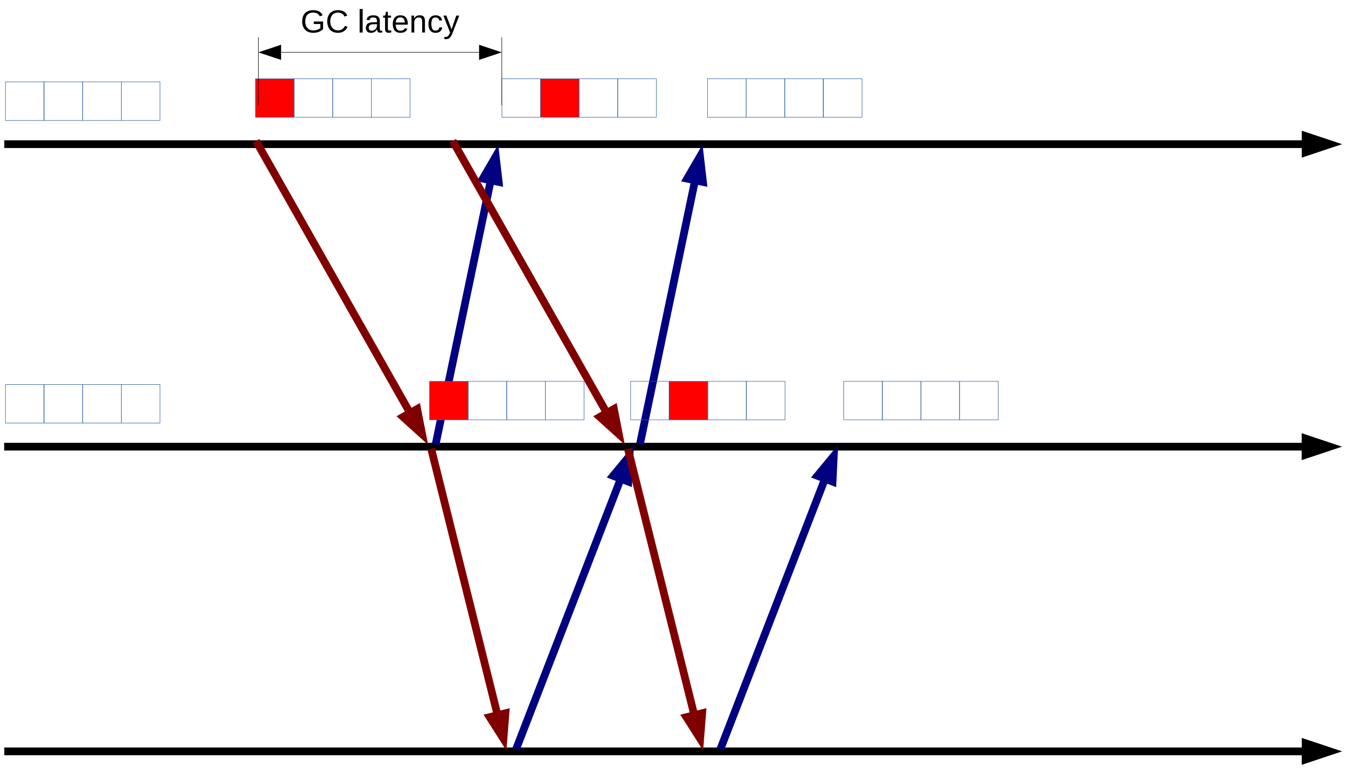
Crybaby

- A single slow receiver delays GC, regardless of structure:



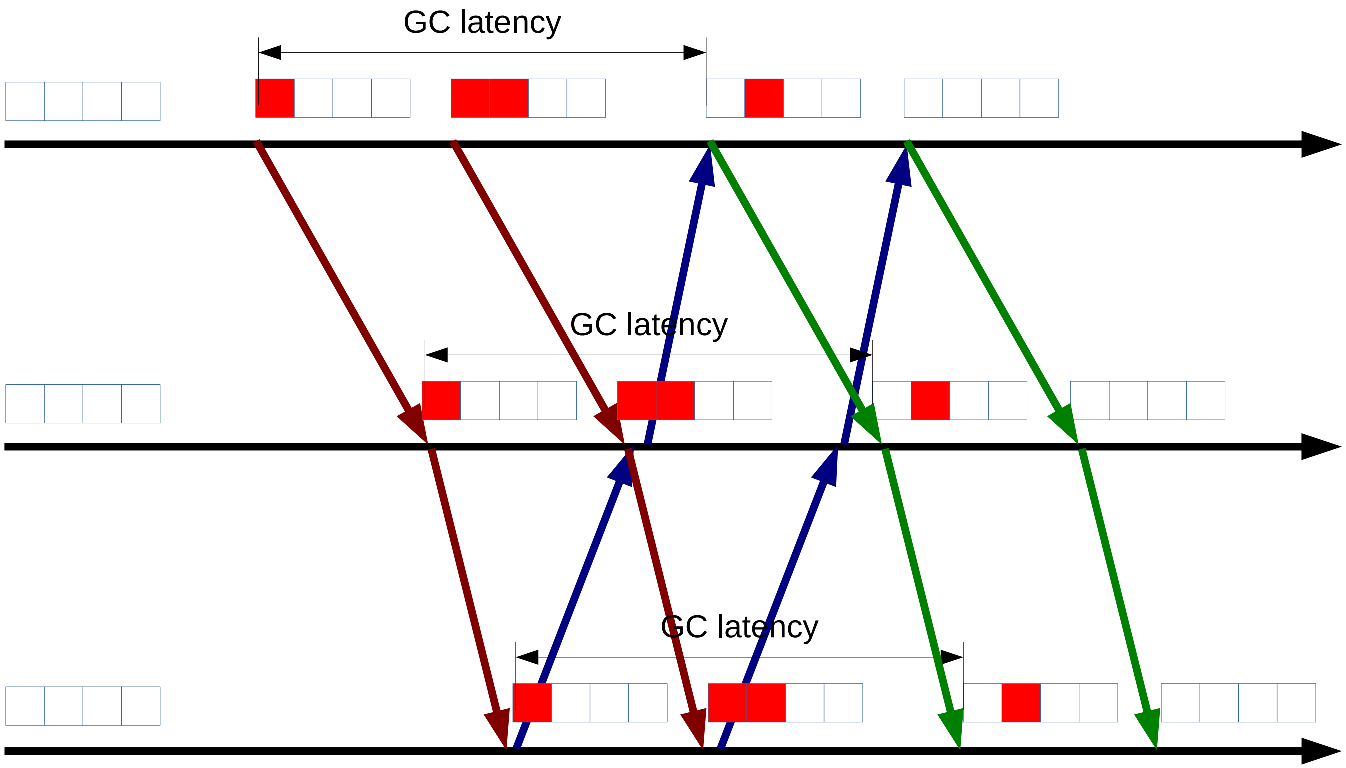
Local acknowledgment

- Local acknowledgment decreases GC latency by using additional buffering:



Multicast with Agreement

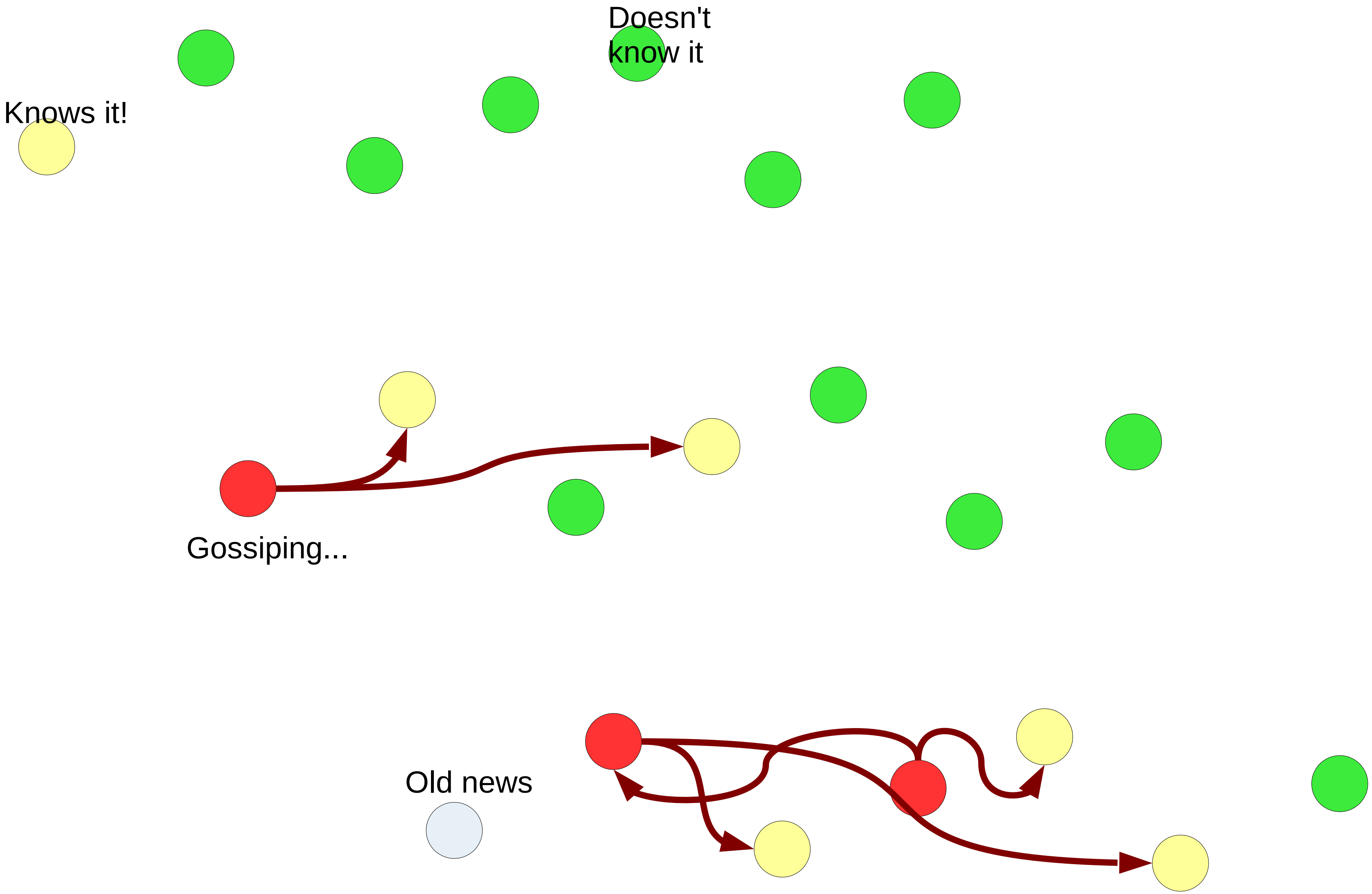
- Two rounds of acknowledgment required to avoid “O(n²) ack implosion”:



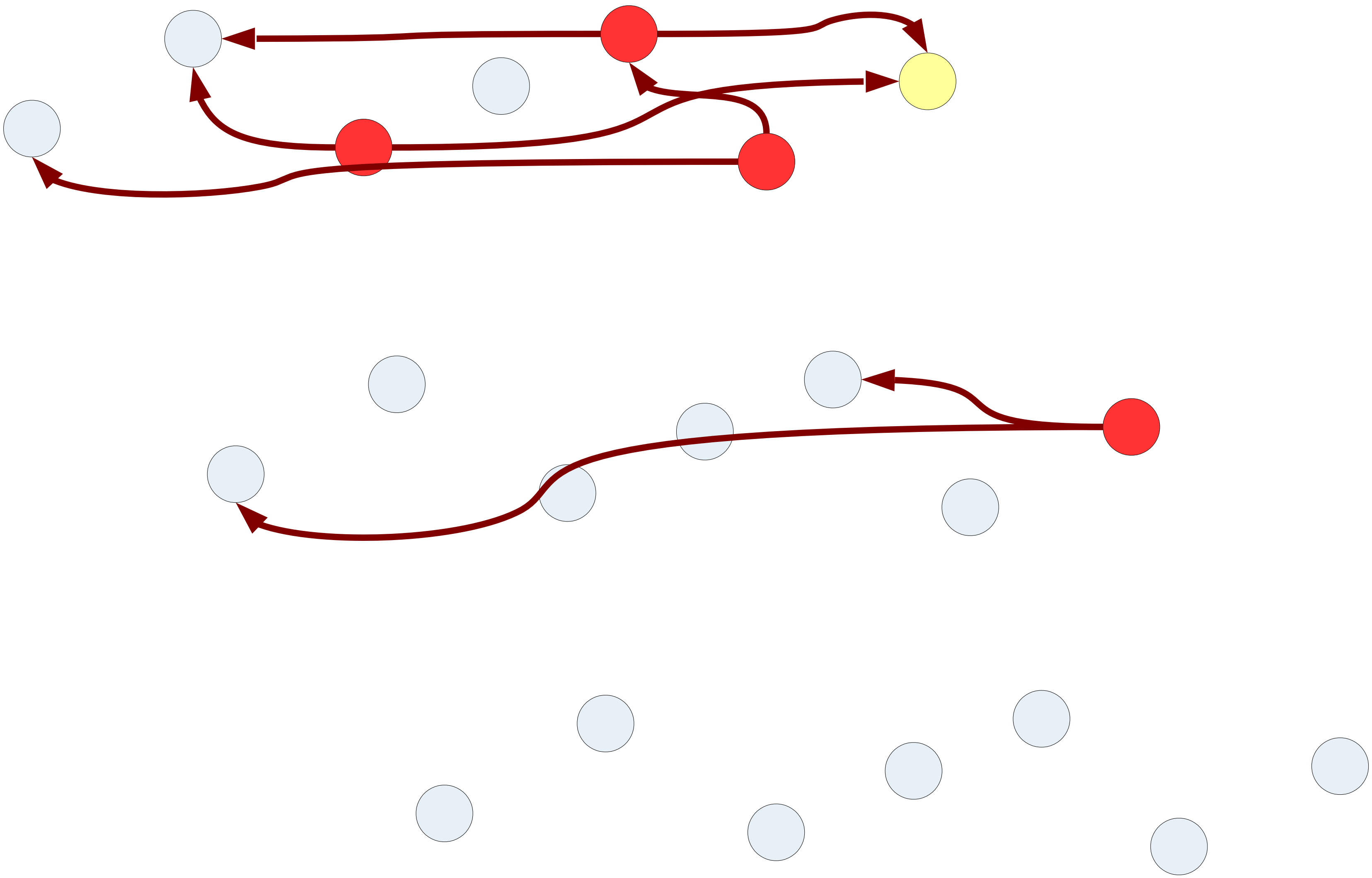
Gossip

- Simple protocol to multicast a message:
  - Select a small subset of random targets
  - Forward message only to those targets
  - Discard message
- Upon receiving a new message, act as the sender

Gossip



Gossip



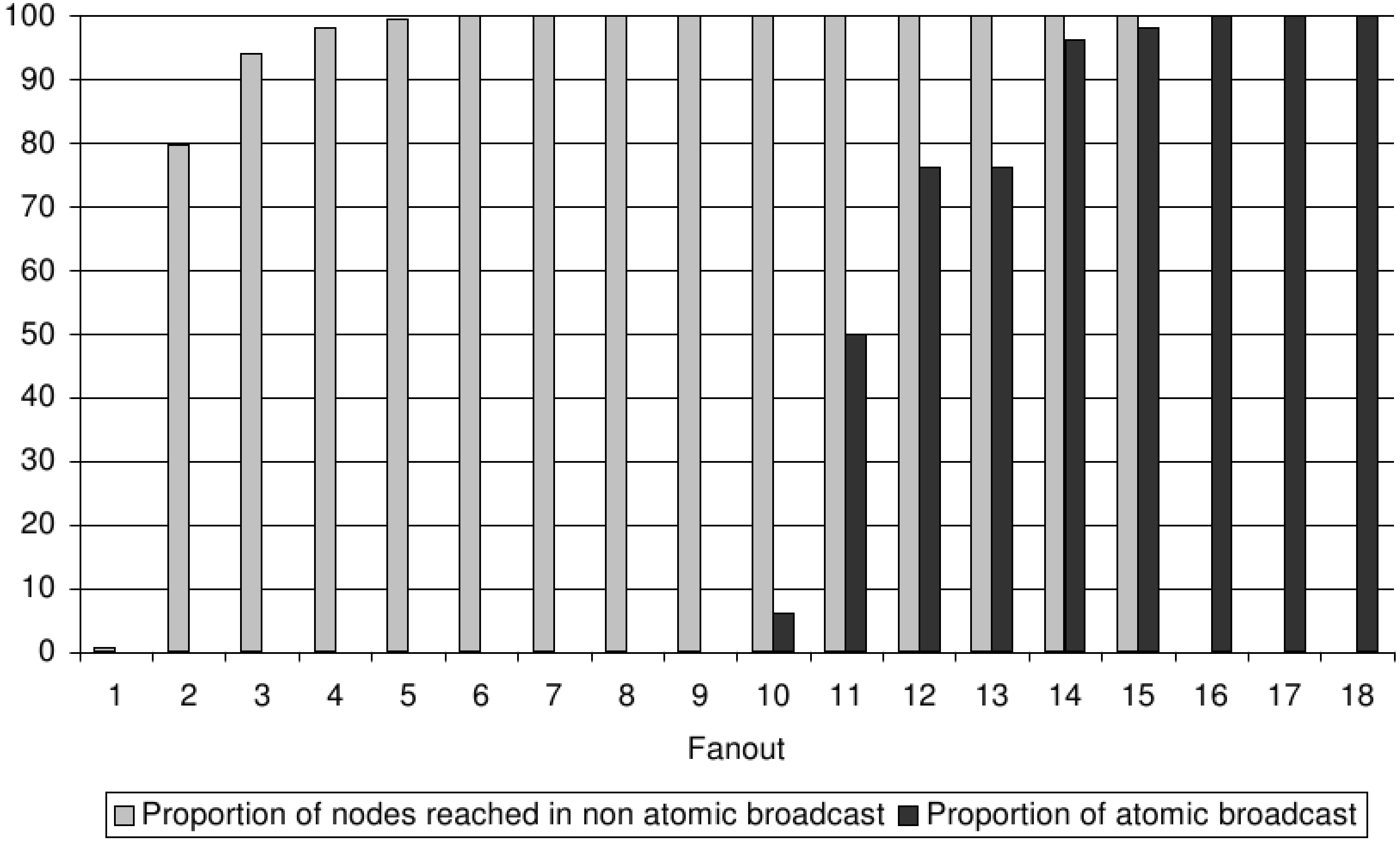
Gossip and Epidemics

- Similarity with epidemics:
  - Sender = contagious = spreads rumor
  - Receiver = infected = knows rumor
  - Ignores duplicated = dead = old news...
- Interesting parameters:
  - *n* – size of the population
  - *f* – number of targets

Fanout vs Reliability

- Infected rate π:
  - $\pi = 1\text{-}exp(\pi f)$
  - Independent of n!
- Probability of atomic infection p:
  - $f = log(n)+c$, $p = exp(-exp(-c))$
  - Depends on n!
- Duration of epidemic when infecting the entire population order of log(n)

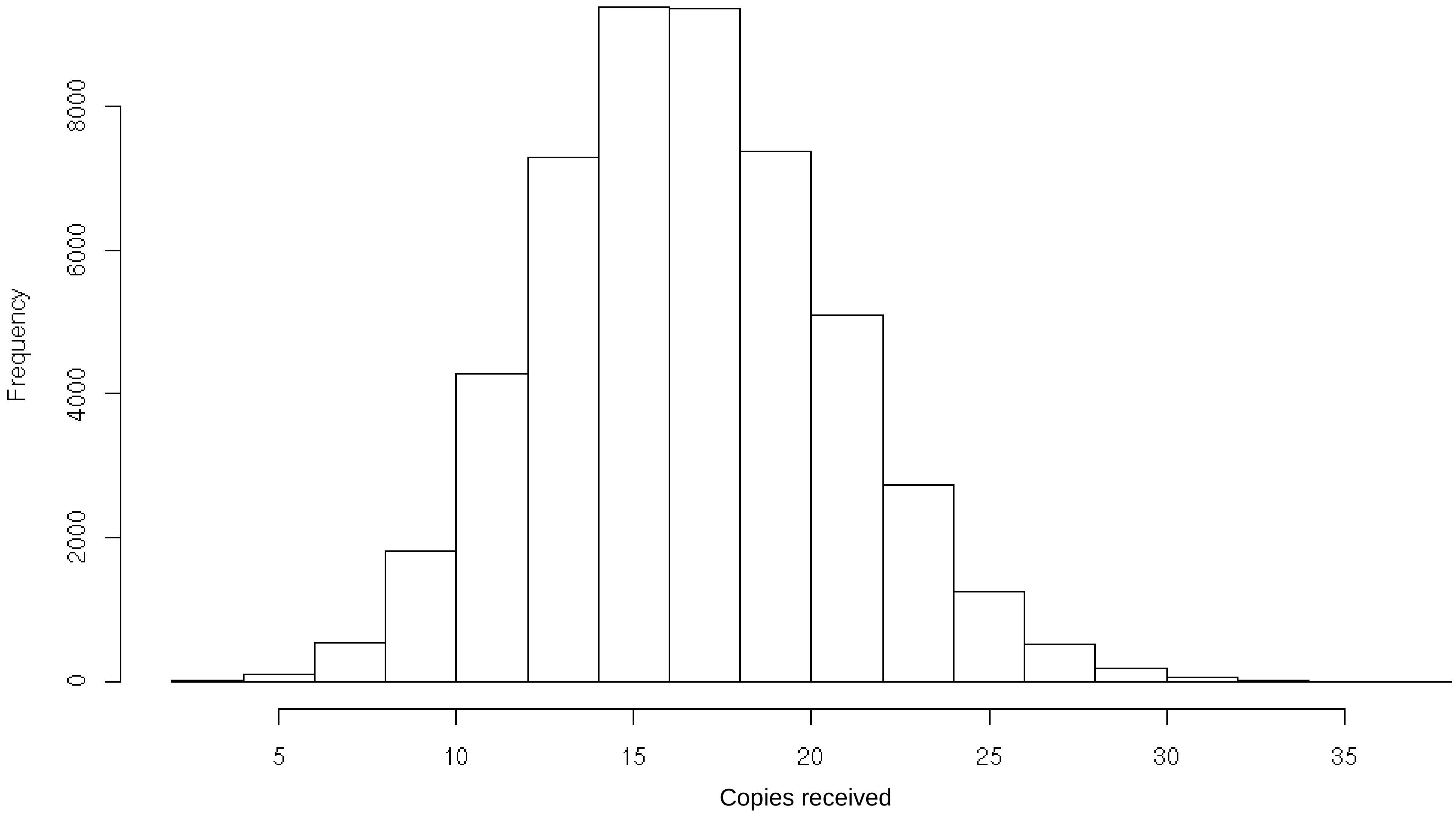
Fanout vs Reliability



(50000 destinations)

In From Epidemics to Distributed Computing.
P. Eugster et al. IEEE Computer, 2005.

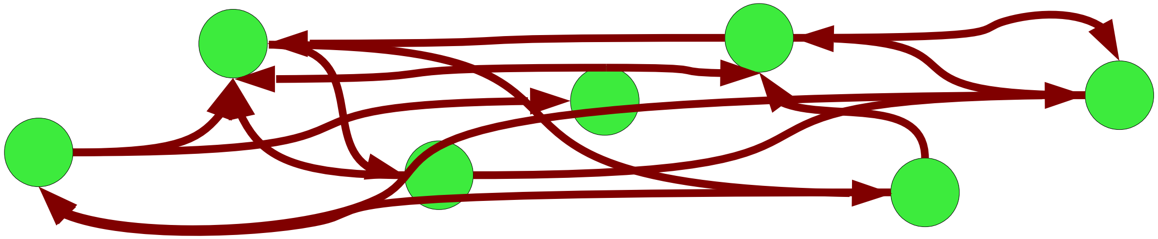
Redundancy



(50000 destinations)

Peer Sampling and Overlays

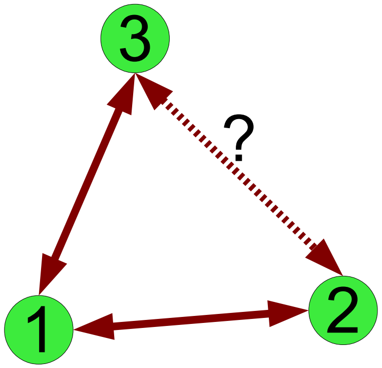
- Need random sample of *f* peers from the set of targets
- Current sample defines implicit overlay network:



- Cannot assume knowledge of all targets to draw from:
  - Potentially very large list
  - How to update it dynamically (churn)?

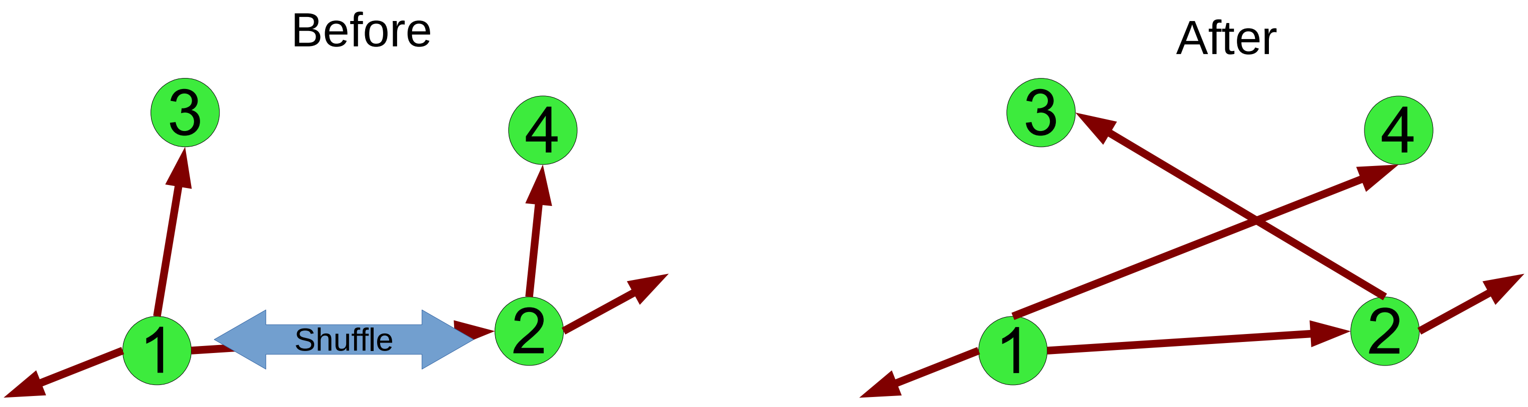
Desirable Overlay Properties

- Connectivity!
  - Convergence to f+c
  - Low variance for load balancing
- Low diameter
  - Latency
- Uniform sample
  - Avoid clustering



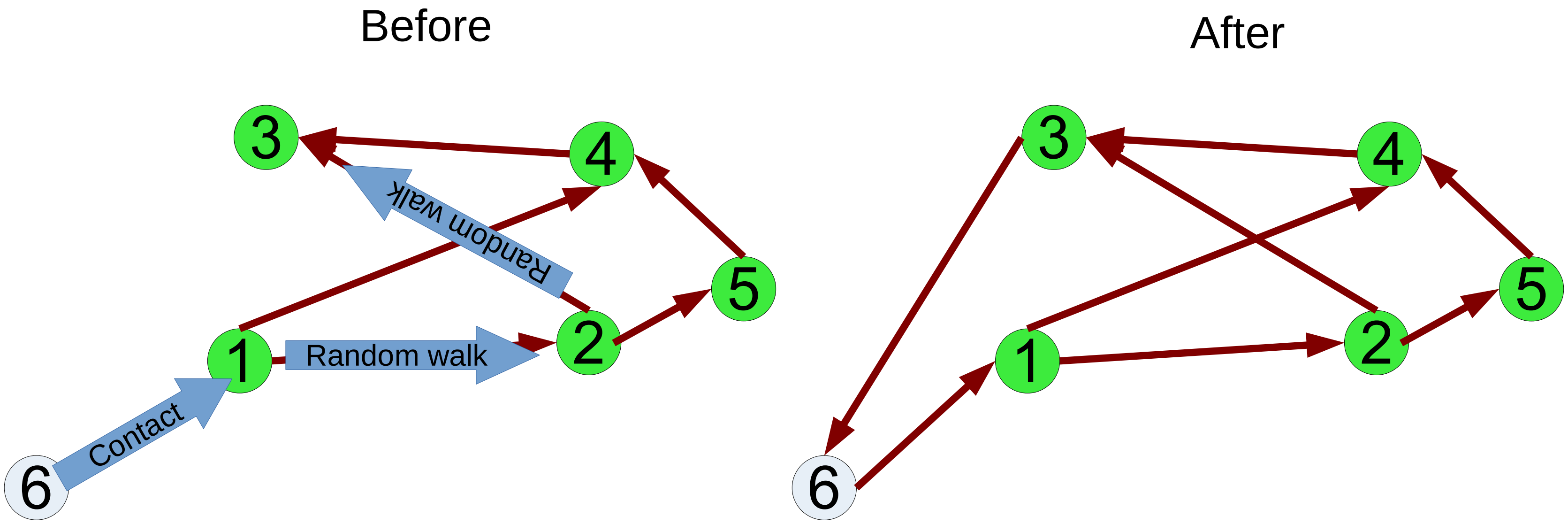
Proactive Overlay Maintenance

- Keep a list of known targets
- Periodically shuffle with known peers:
  - Evict some local targets at random
  - Adopt some remote targets at random



Reactive Overlay Maintenance

- To join a group, use a contact point
- Initiate *f+c* random walks in the overlay from the contact point
- Upon arrival, insert new member in local target set


Before

After

Fault Tolerance

- Pros:
  - Tolerates faults by increasing parameter f:
    - Packet loss
    - Dead processes
  - Immune to the crybaby, since there is no feedback
- Cons:
  - Assumes independent faults

Performance

- Pros:
  - No tree setup/repair overhead
  - Perfect load balancing
- Cons:
  - Redundancy