Large Scale Distributed Systems

José Orlando Pereira

Departamento de Informática
Universidade do Minho

Reliable event dissemination

- Reliably send to multiple destinations (group)

- Informally: all destinations deliver all messages
- Senders and receivers fail: all correct destinations deliver the same messages
  - average % of destinations
  - % of messages to all destinations

Agreement / Atomicity

Performance metrics

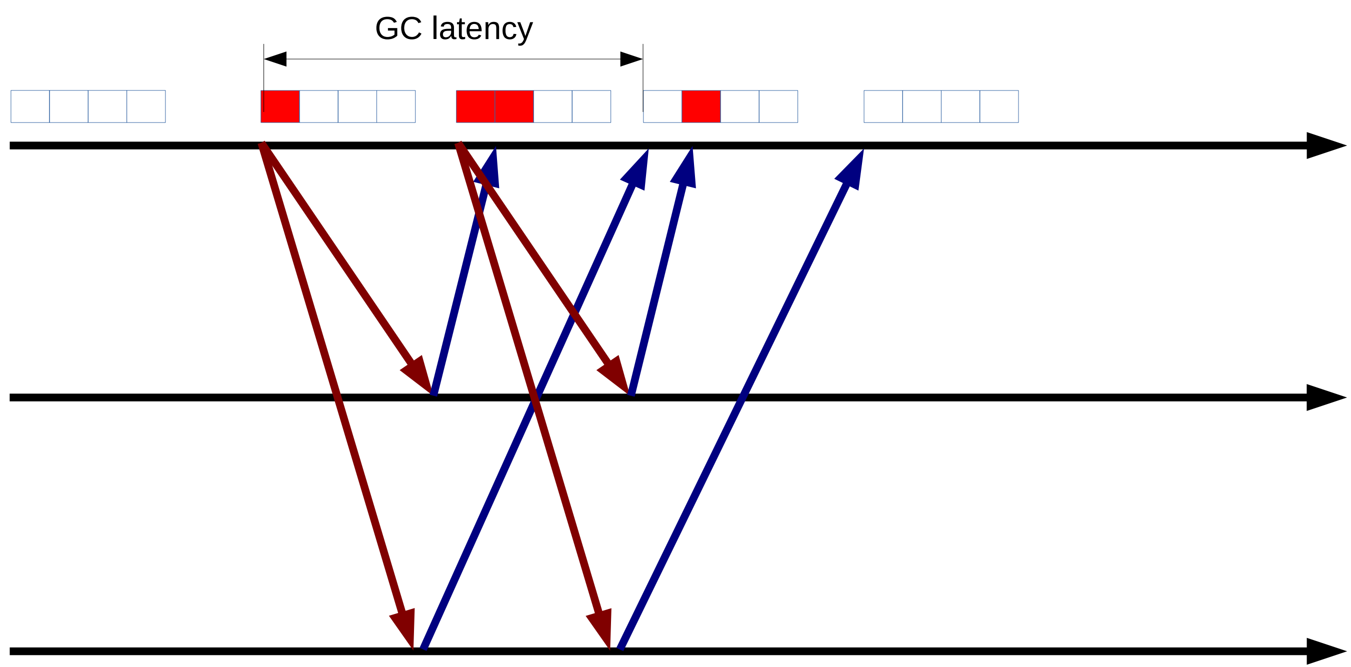
- Bandwidth
  - Payload and control messages
  - System total and maximum in one node/link

Tradeoff!

- Latency
  - Time and network hops to delivery
  - Average vs last delivery

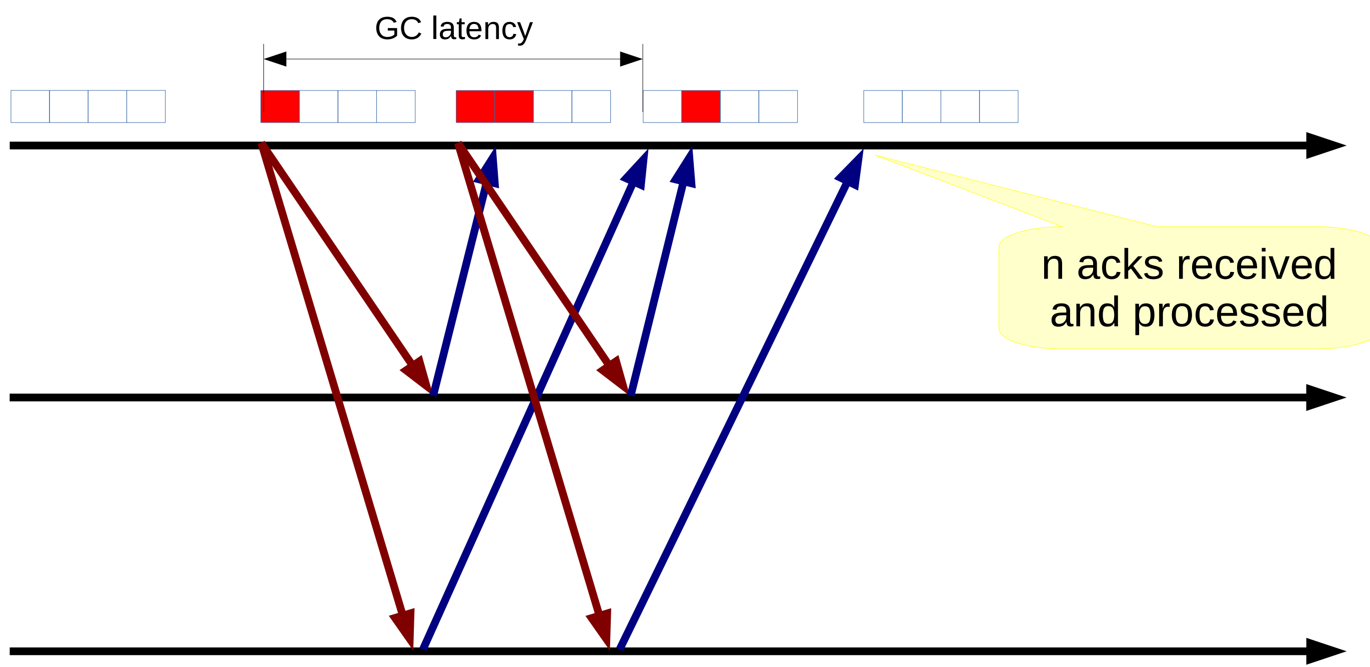
General approach

- Buffer and retransmit until acknowledged



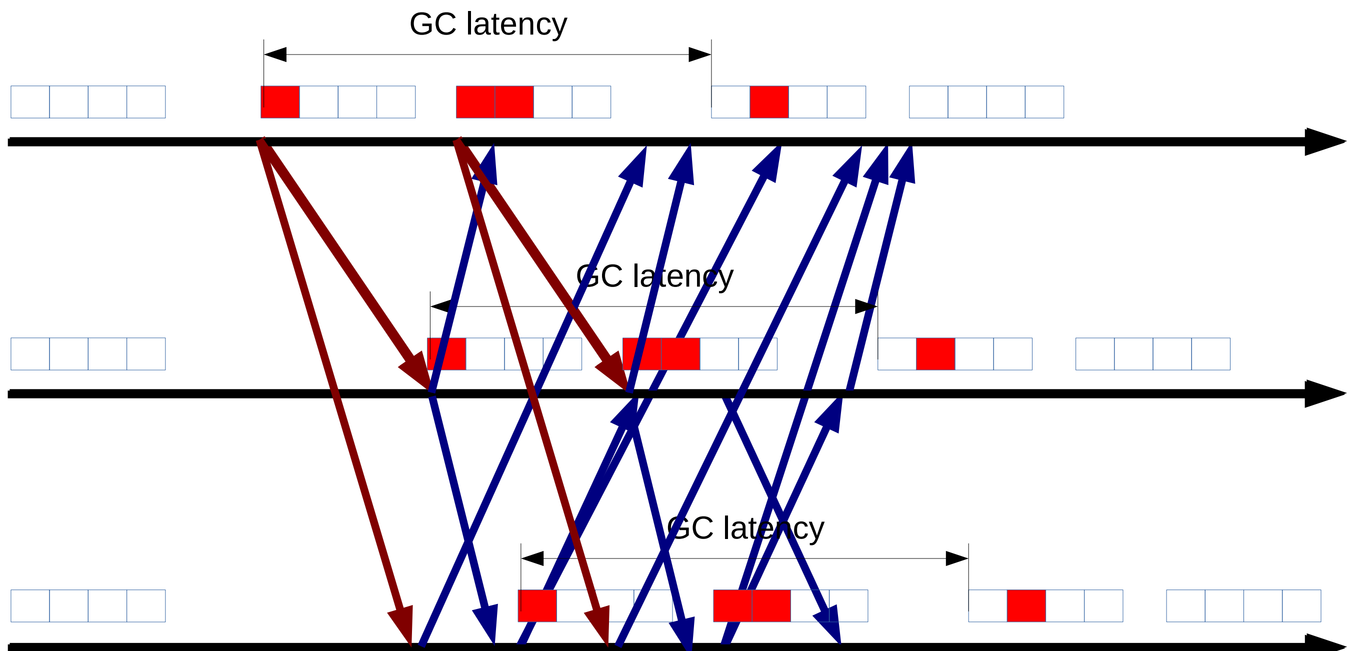
Acknowledgments

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



Agreement

- Acks need to be sent to all destinations resulting in “O(n^2) ack implosion”:

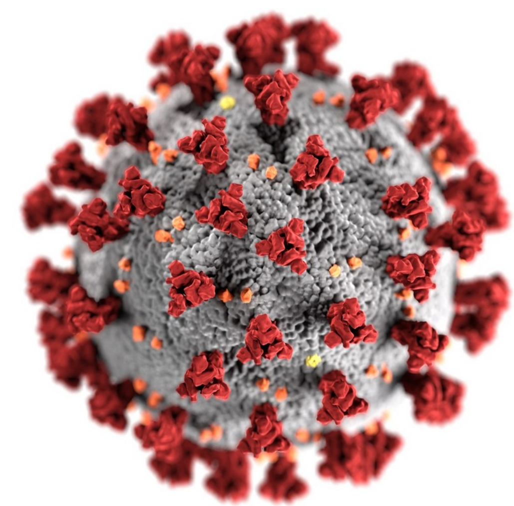


Trees

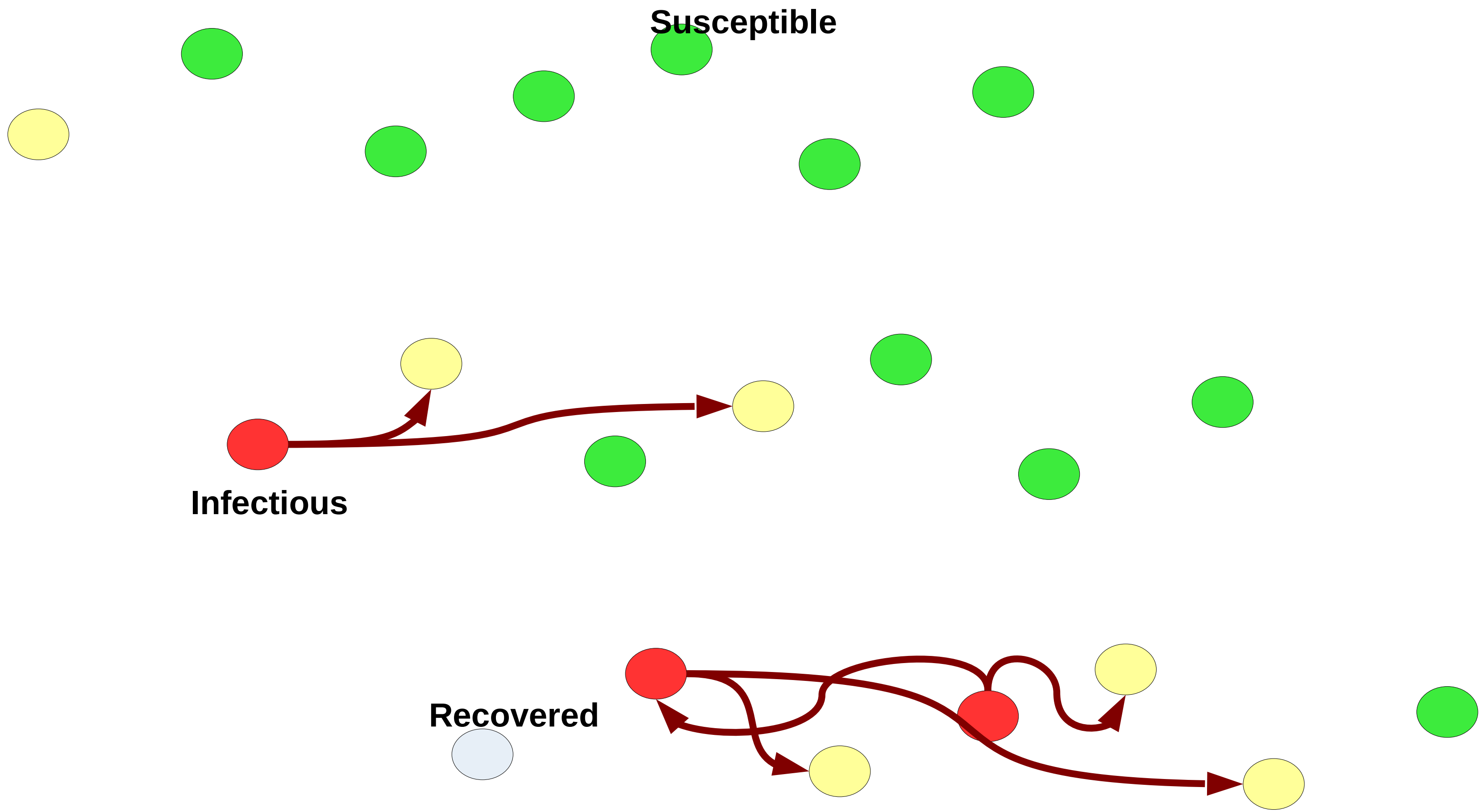
- Build a spanning tree over the destination nodes
  - Maintenance overhead
  - Brittle, as a single node/link failure leads to partitioning
- Use the tree for payload and feedback (acknowledgments)
  - Global vs local acknowledgment
- Trades latency for bandwidth!
  - More latency means more memory committed in buffers

Epidemics

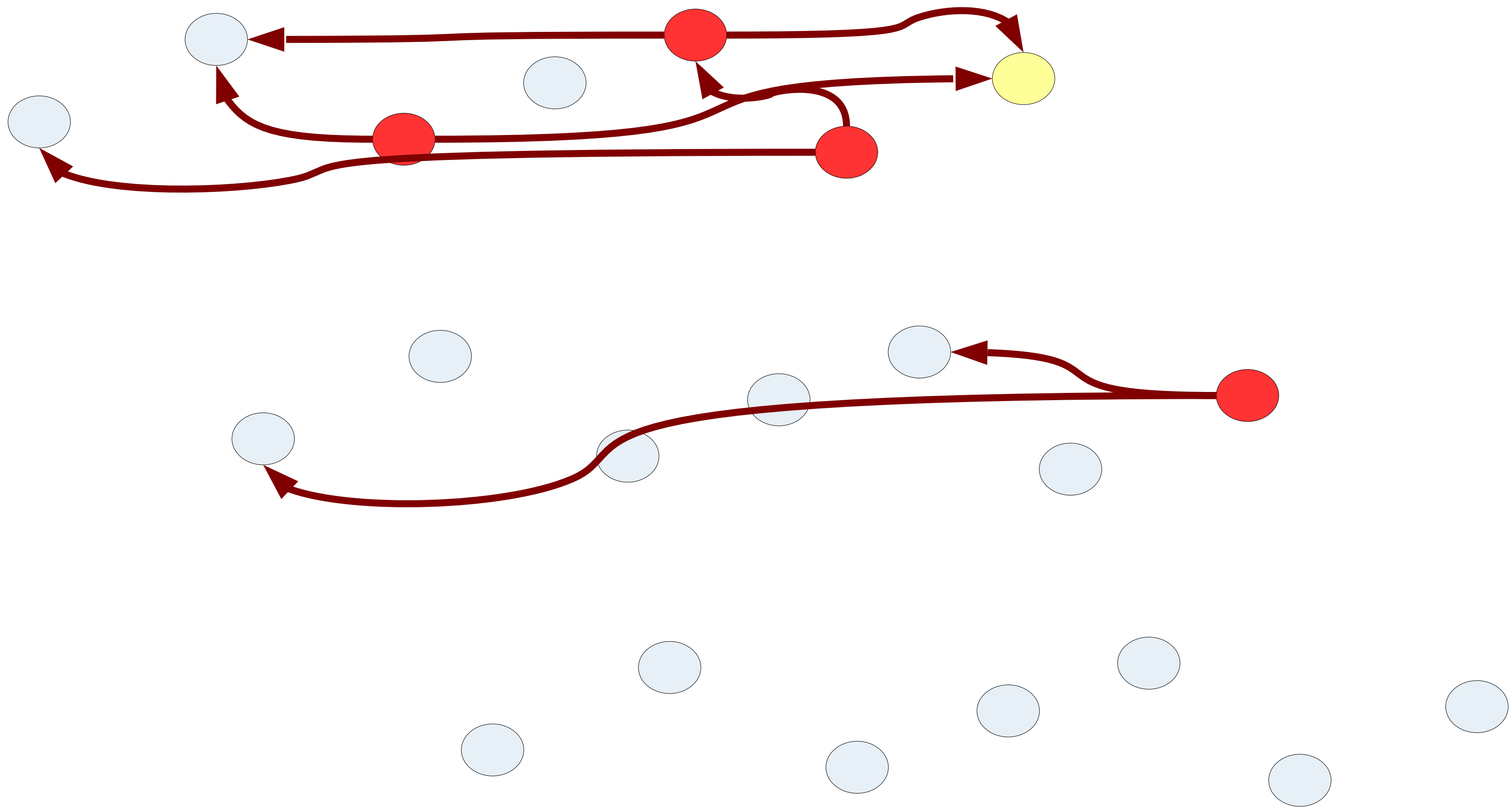
- Epidemic spread in a population
- SIR model:
  - Susceptible
  - Infectious
  - Recovered



Epidemics



Epidemics



Analysis

- Probability of atomic infection p:
  - $f = \log(n)+c$
  - $p = \exp(-\exp(-c))$
- Duration of epidemic when infecting the entire population order of $\log(n)$

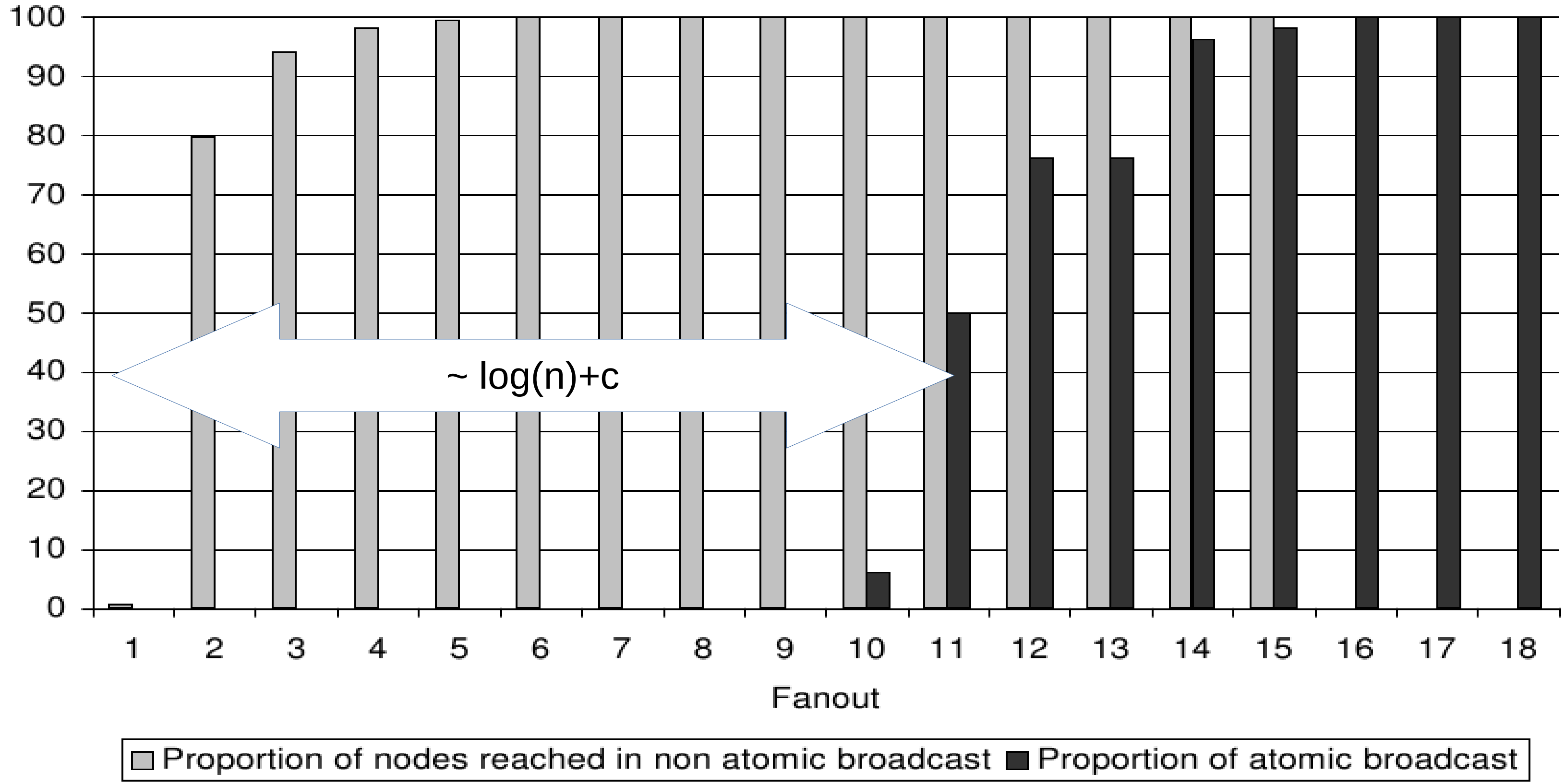
Roadmap

- Epidemic event dissemination and efficiency
- Distributed aggregation

Epidemics and Information dissemination

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

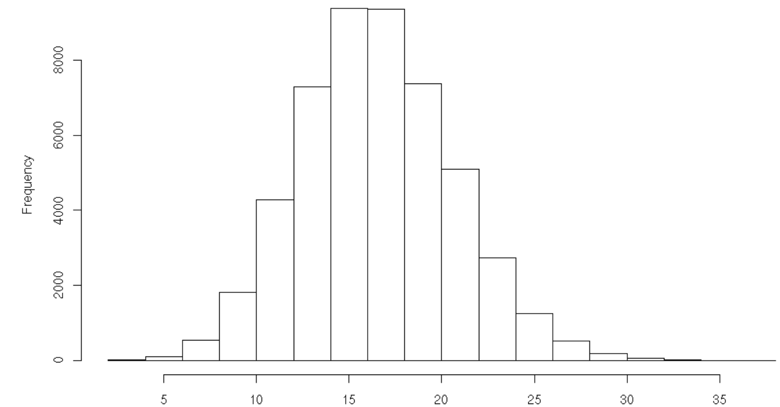
Fanout vs Reliability

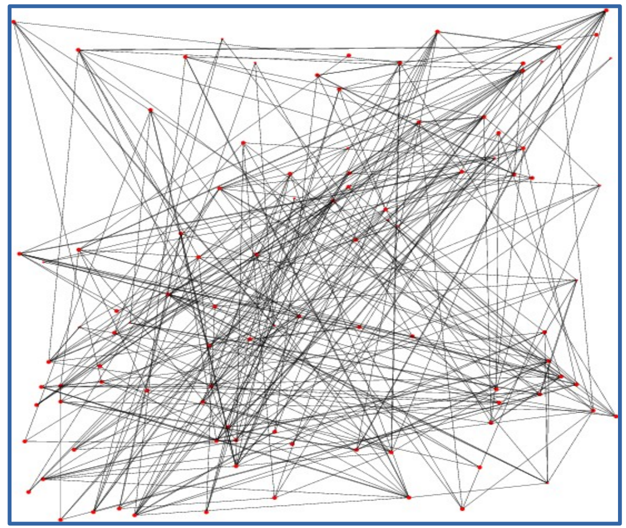


(50000 destinations)

Load-balancing and redundancy

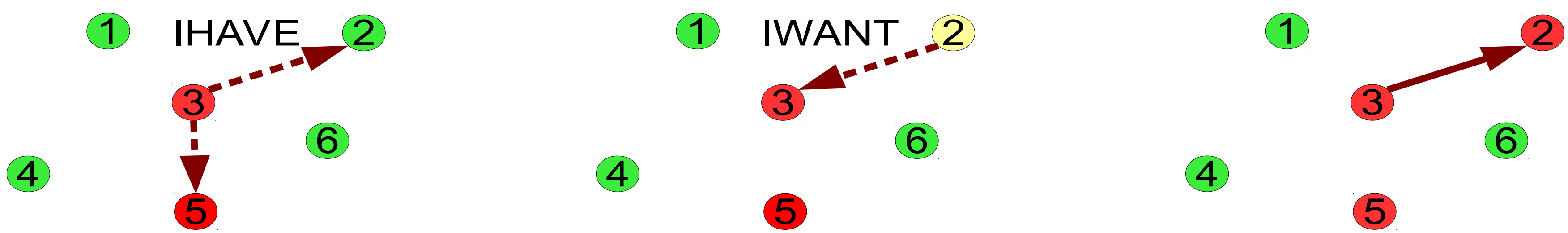
- Distribution of traffic on nodes and links:
  - Normal
  - Mean = fanout
- *f-1* useless payload copies for each destination!
- Not using nodes and links with higher capacity
- Using expensive links as much as cheap ones





Lazy dissemination

- Defer sending the payload (lazy transmission)
  - Equivalent to eager transmission (minor increase in probability of failure)
  - Saves bandwidth, assuming that the payload is much larger that a control message with the event id

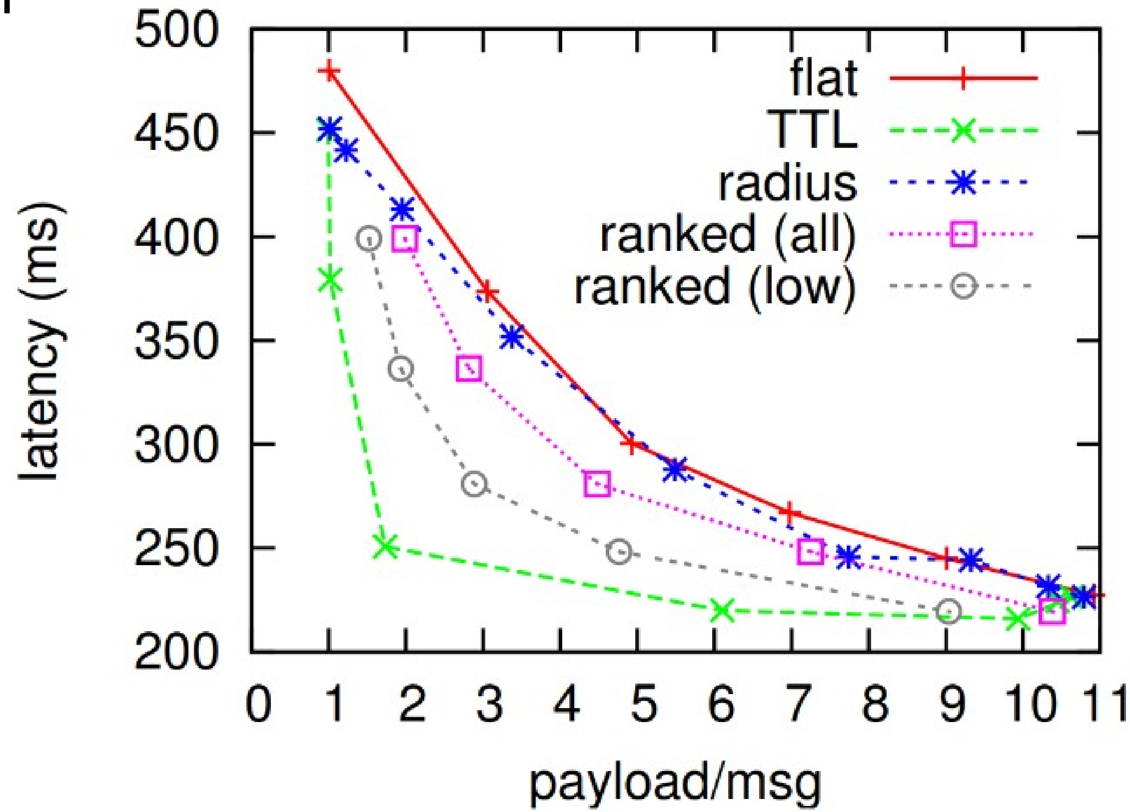


Emergent structure

- Manipulate eager vs. lazy transmission such that the protocol achieves efficiency goals with high probability
  - Using cheap network links
  - Using large capacity nodes
  - Reduces redundant payload copies
- Decision based on local information:
  - Stateless: Decisions are independent
  - Stateful: Decision based on previous observations

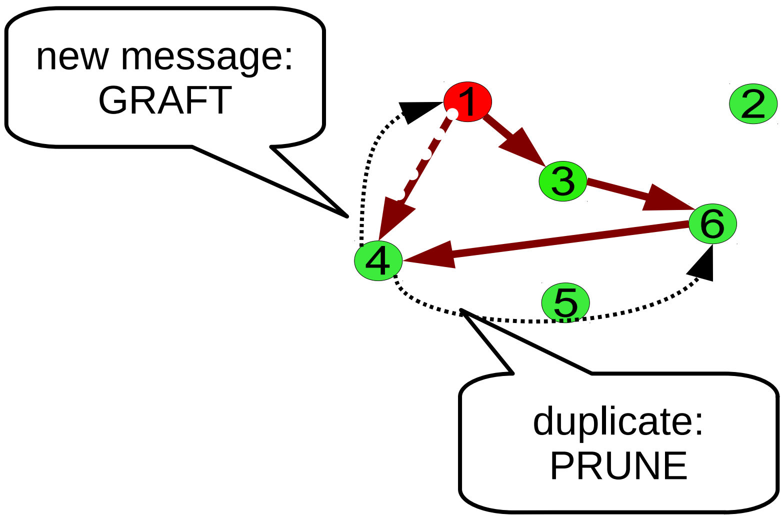
Time-to-live

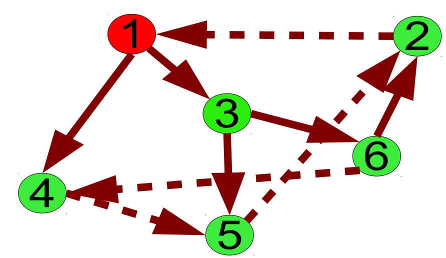
- Observation: Probability of a transmission being a duplicate is:
  - 0 in first round
  - minimal in 2nd round
  - …
  - 1 in latest round
- Use eager for first k hops achieves <2 payloads with good latency



Push-Lazy-Push Tree (aka “plumtree”)

- Start by using eager transmission:
  - When receiving duplicates, ask sender to PRUNE
  - When receiving annoucements for unknown messages, as sender to GRAFT





References

- P. T. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulie, “Epidemic information dissemination in distributed systems,” IEEE Computer , vol. 37, no. 5, pp. 60–67, May 2004. http://dx.doi.org/10.1109/MC.2004.1297243
- J. Leitao, J. Pereira, and L. Rodrigues, “Epidemic Broadcast Trees,” in 2007 26th IEEE International Symposium on Reliable Distributed Systems (SRDS 2007), Oct. 2007, pp. 301–310. http://dx.doi.org/10.1109/SRDS.2007.27

Distributed aggregation

- Computation of aggregate functions:
  - Max, Top-K (idem for Min, ...)
  - Average
  - Count, sum
- Avoid collecting all values at a single node

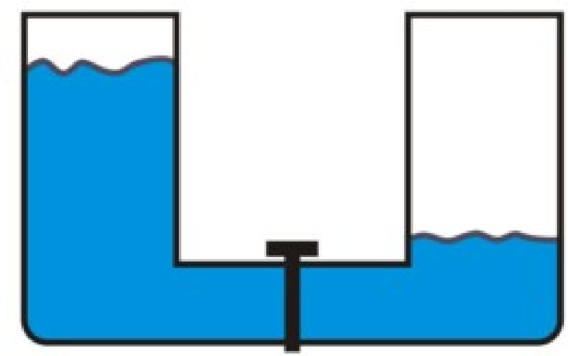
CAI: Comutative, Associative, and Idempotent

Distributed aggregation (max, ...)

- Each node i starts with estimate $e = v_i$
- Exchange estimates with node j
- When receiving e_j, make $e_i = f(e_i, e_j)$
- Right answer after enough time for each “opinion” to impact the whole network: T = log n + c times

Distributed aggregation (average)

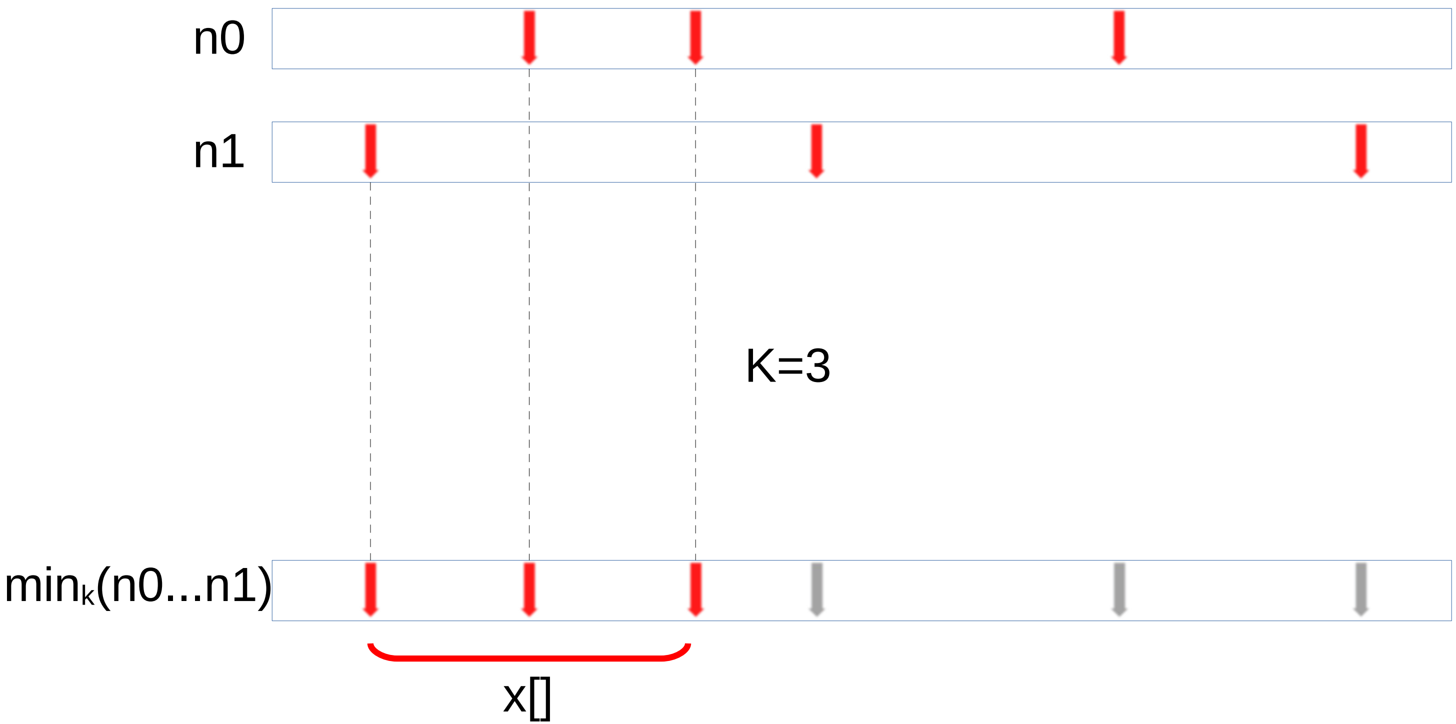
- Each node i starts with estimate $e = v_i$
- Exchange estimates with node j
- When receiving e_j, make $e_i = f(e_i, e_j)$
- Right answer only if exchanges are atomic
- Strategies to avoid duplication/leaks:
  - Use multiple samples (newscast)
  - Keep a current account (flow)



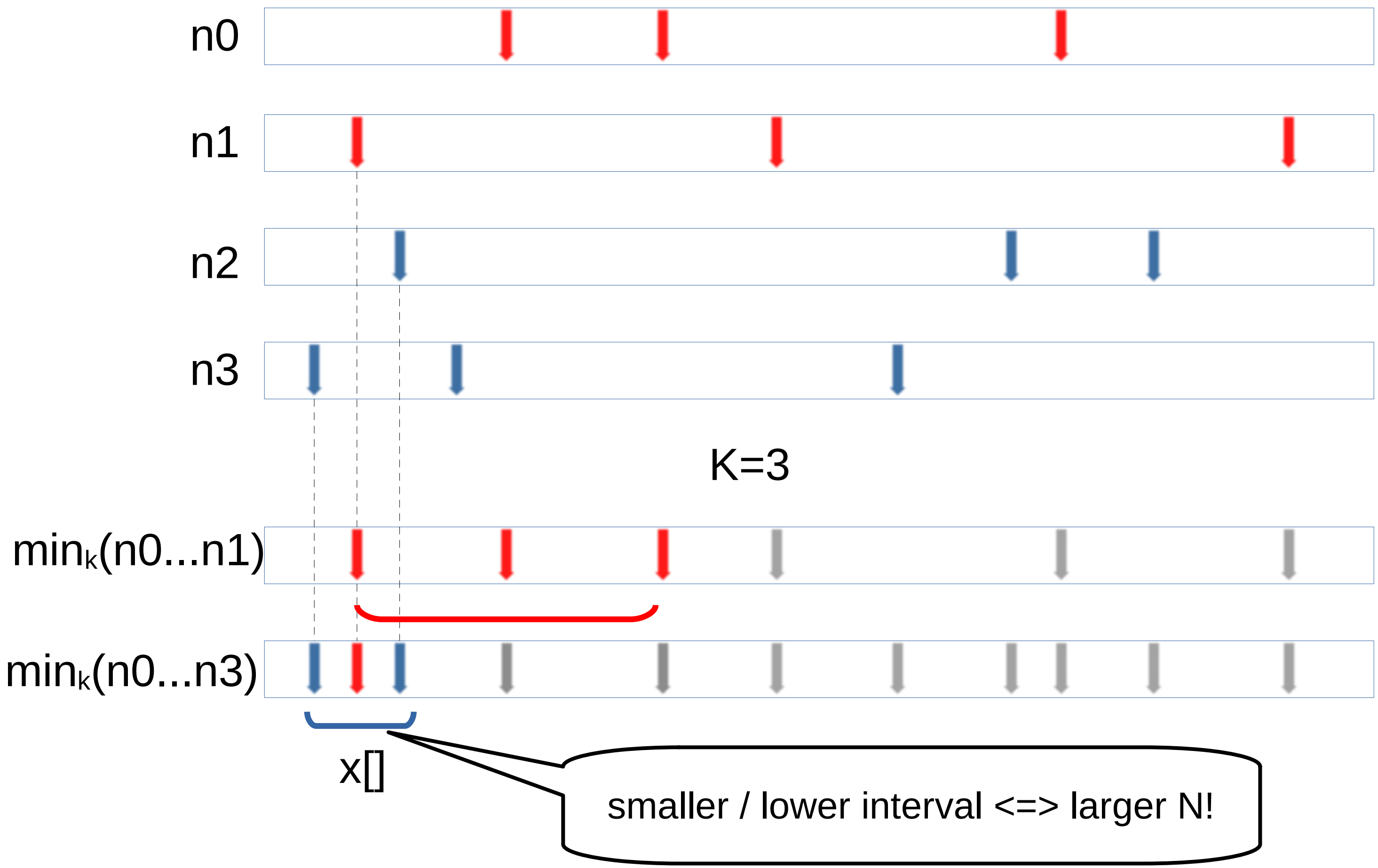
Distributed aggregation (count)

- Count can be obtained from average:
  - one node starts with value 1, all others with 0
  - count = 1/average
- Can be obtained from max/min:
  - extrema propagation

Extrema propagation (intuition)



Extrema propagation (intuition)



Distributed aggregation (count/sum)

- Each of N nodes selects a vector x[] of K random values
- Nodes exchanges vectors and keep minimal values seen
  - Eventually, all nodes have the same values!
- Repeat until vector x[] is unchanged for T rounds
- Estimate count N from resulting vector x[]

- Can be generalized to compute Σv_i, therefore can also compute averages

References

- M. Jelasity, W. Kowalczyk, and M. van Steen, “An approach to massively distributed aggregate computing on peer-to-peer networks,” in 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2004. https://doi.org/10.1109/EMPDP.2004.1271446
- C. Baquero, P. Almeida, R. Menezes, and P. Jesus, “Extrema Propagation: Fast Distributed Estimation of Sums and Network Sizes,” IEEE Trans. Parallel Distrib. Syst., vol. 23, pp. 668–675, Apr. 2012. http://dx.doi.org/10.1109/TPDS.2011.209