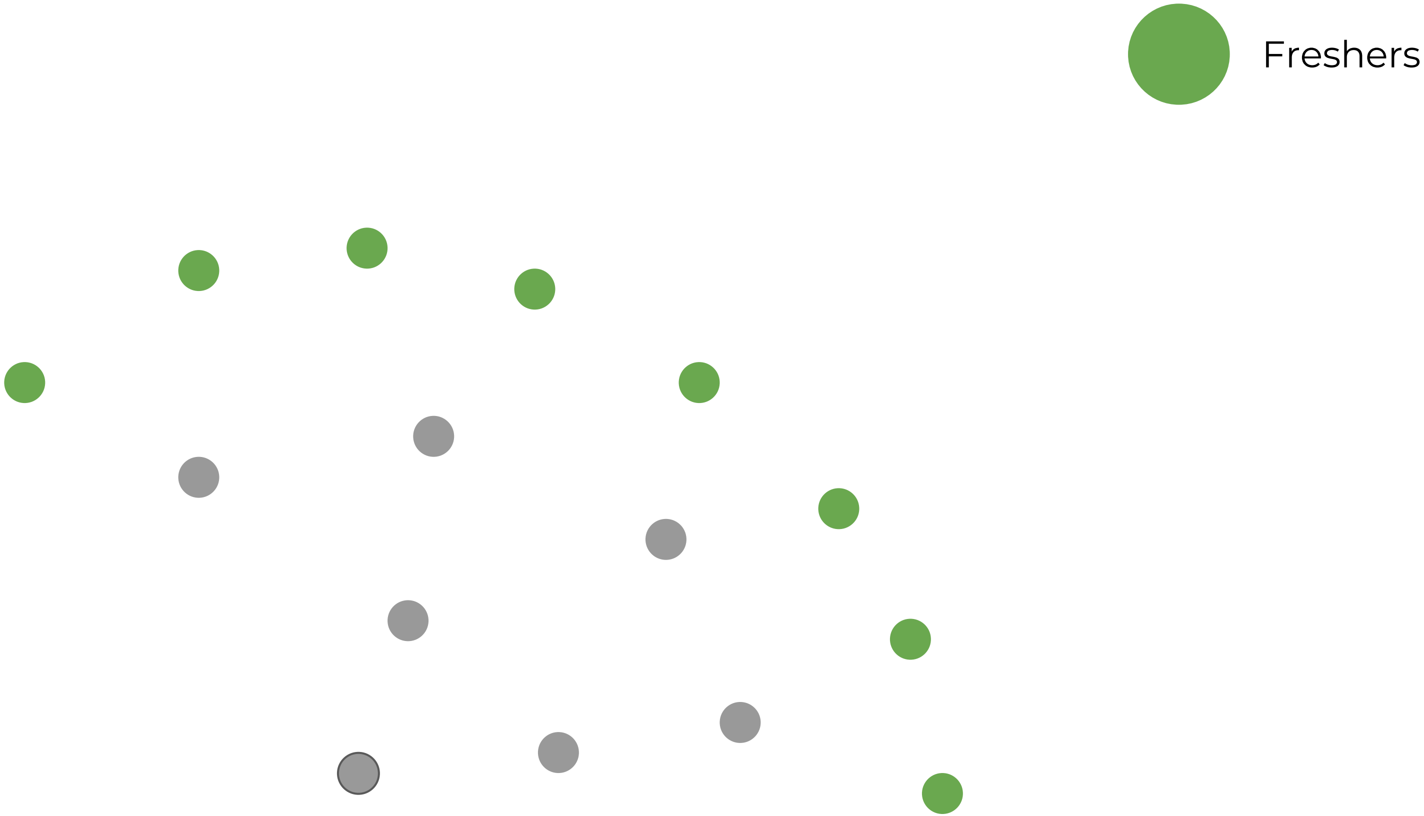
Making servers gossip

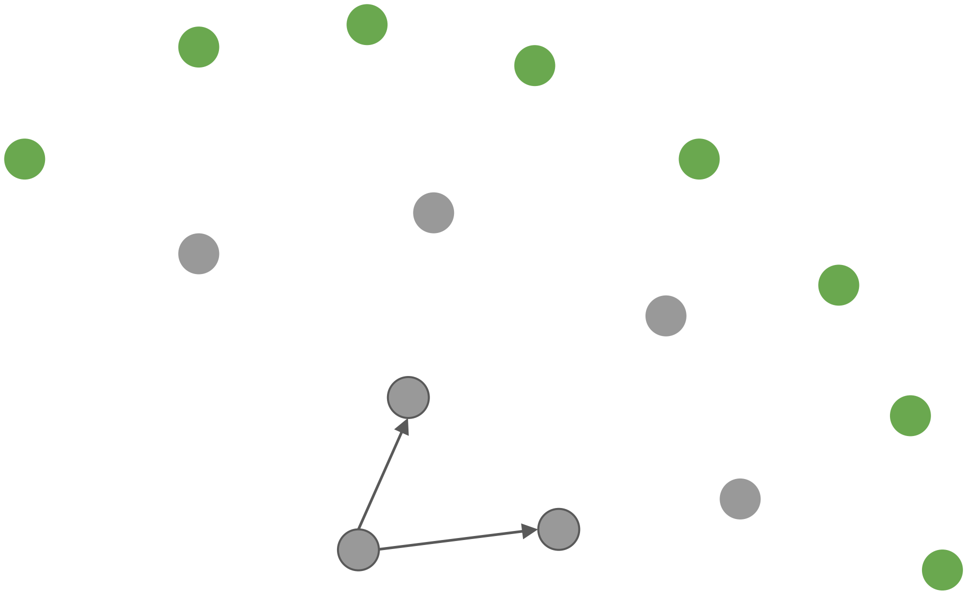
Tomás Fernandes

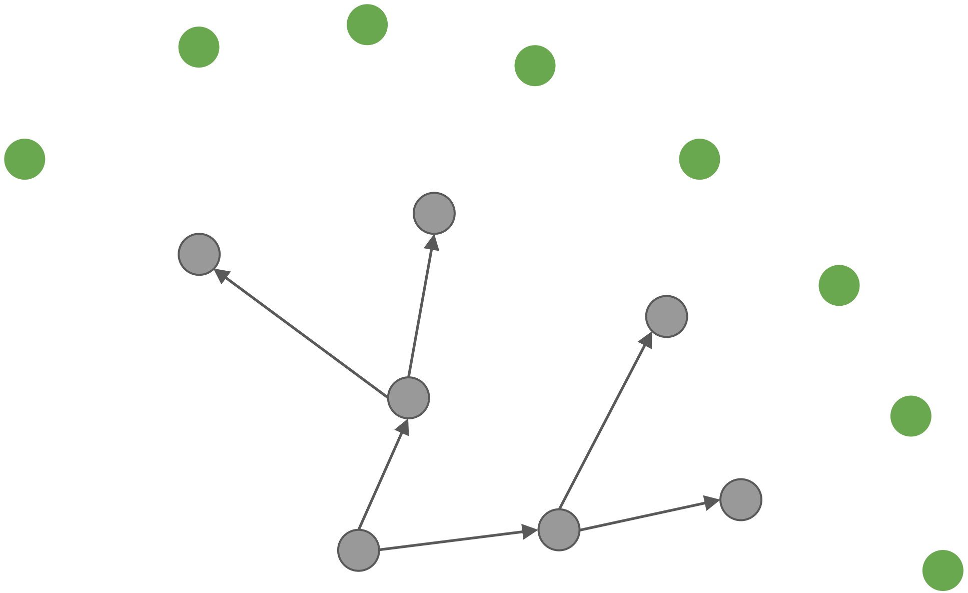
“UWCS is giving free pizza during Friday Night Gaming”

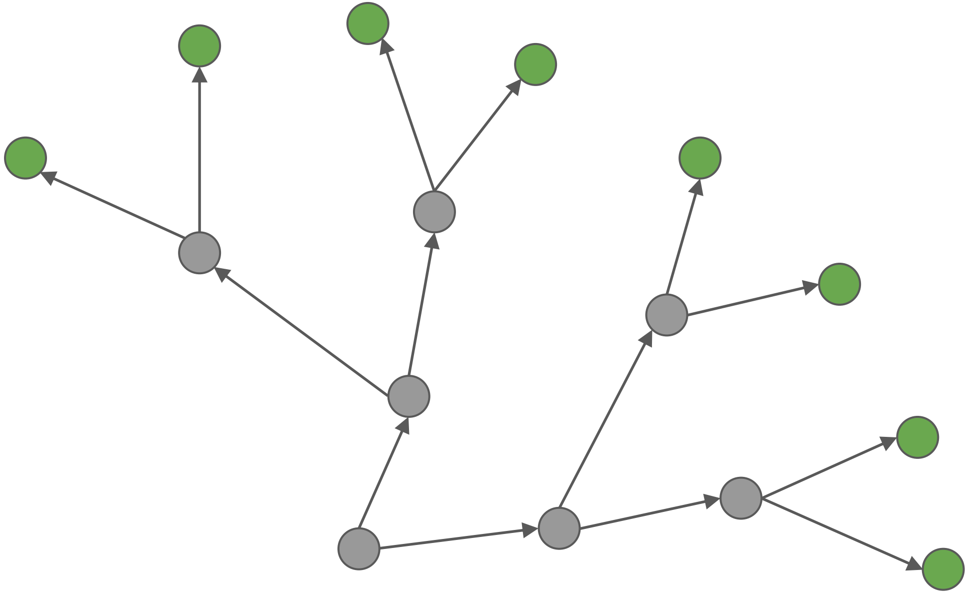
- Some Computer Science student, circa 2022



Freshers







UWCS goes bankrupt!

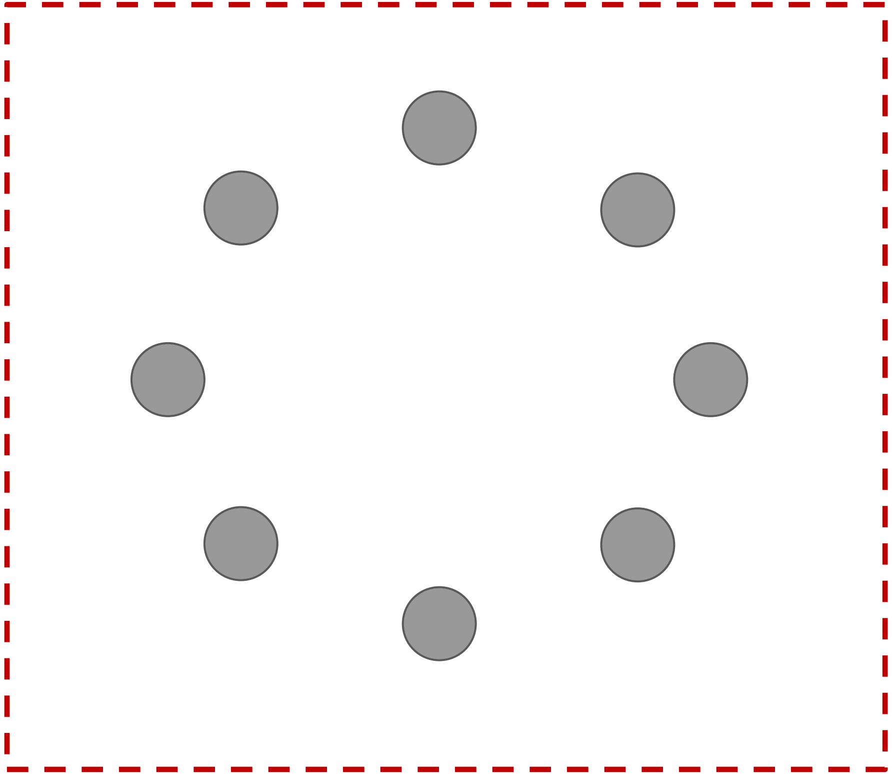
UWCS goes bankrupt!

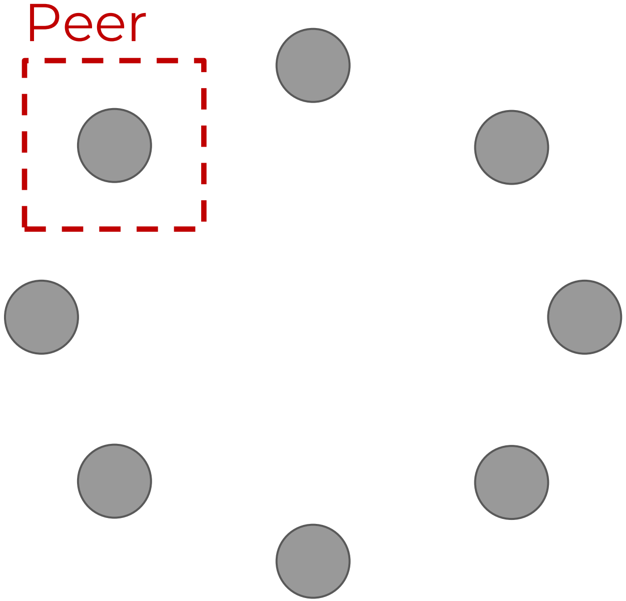
So what?

Gossip protocol

Inspired by **human gossip** and **epidemics**

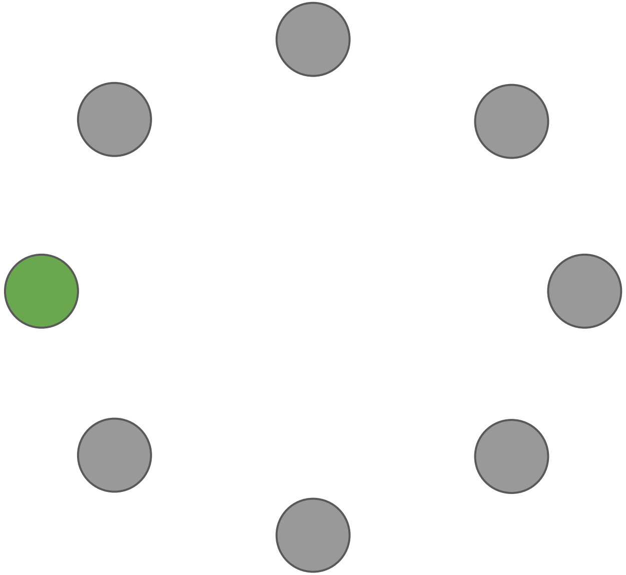
Overlay / Topology



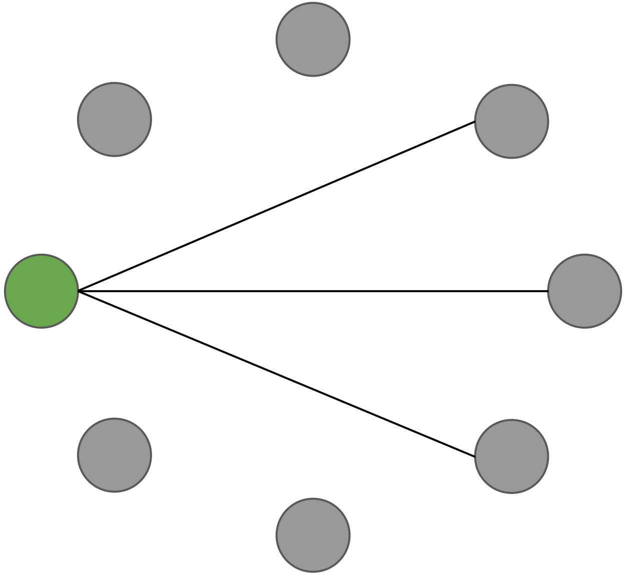

Peer

Round 1

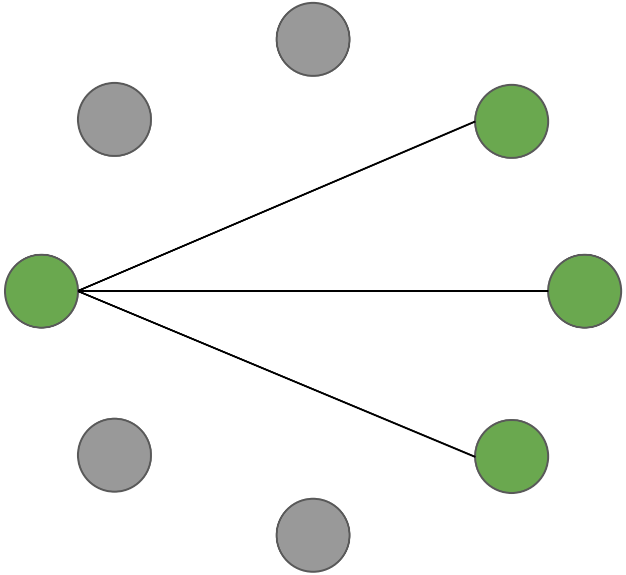
Fanout = 3



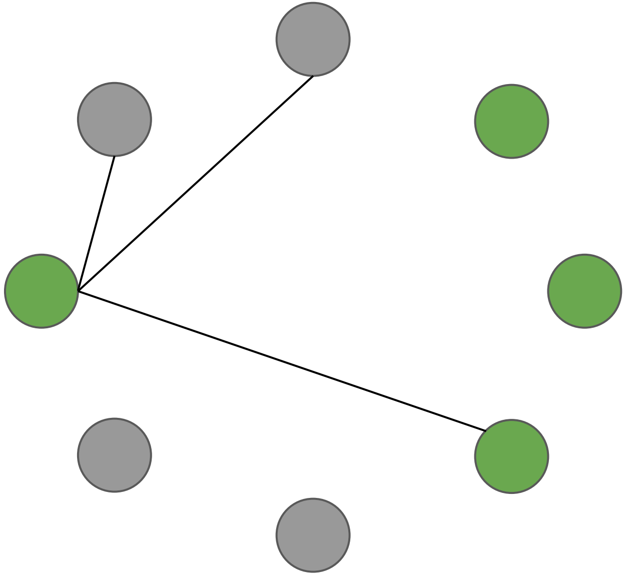
Round 1



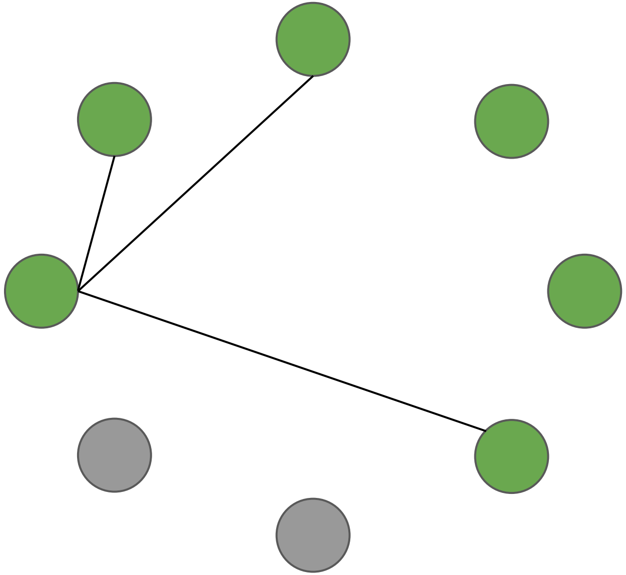
Round 1



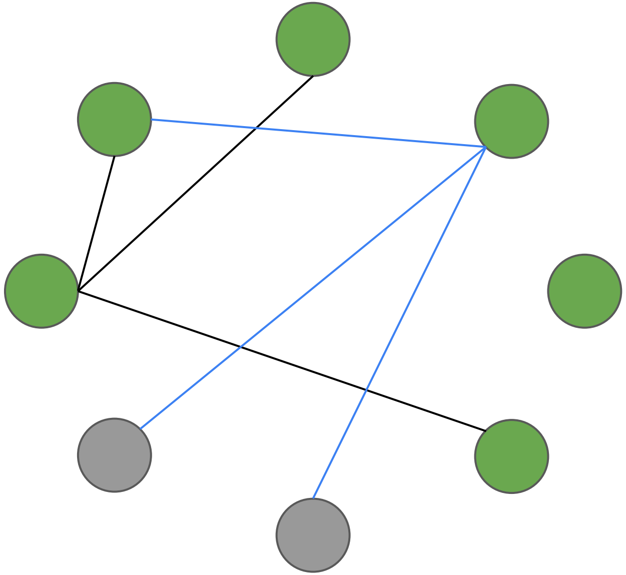
Round 2



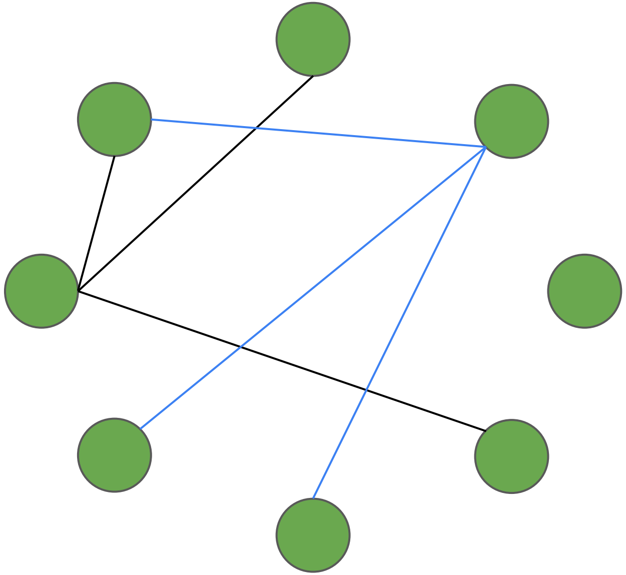
Round 2



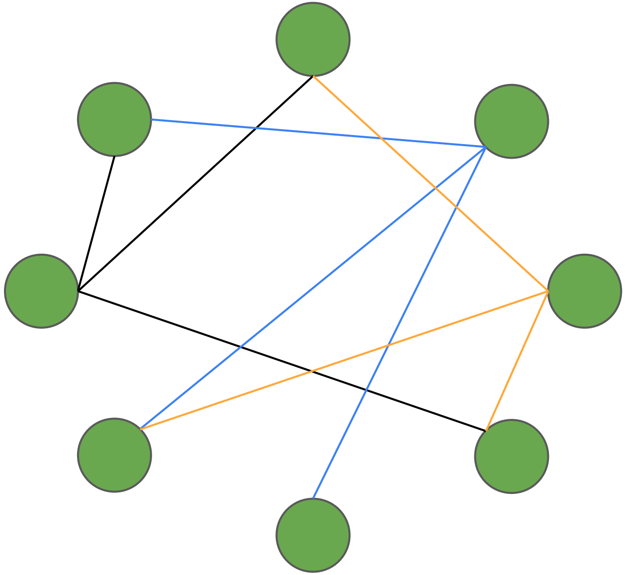
Round 2



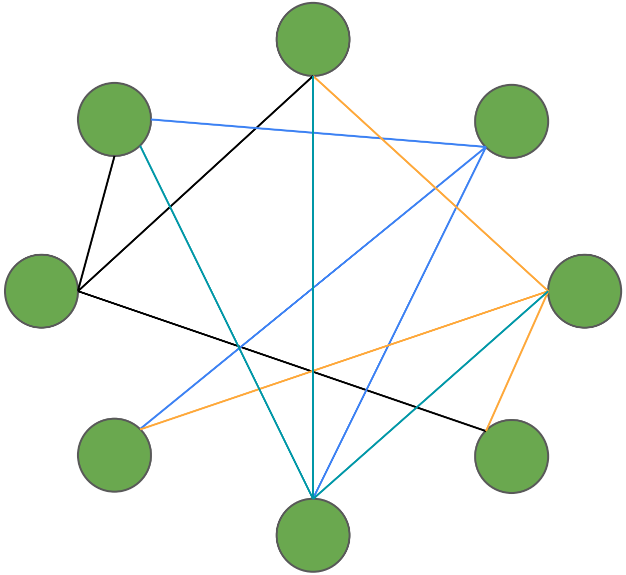
Round 2



Round 2



Round 2



Scalable!

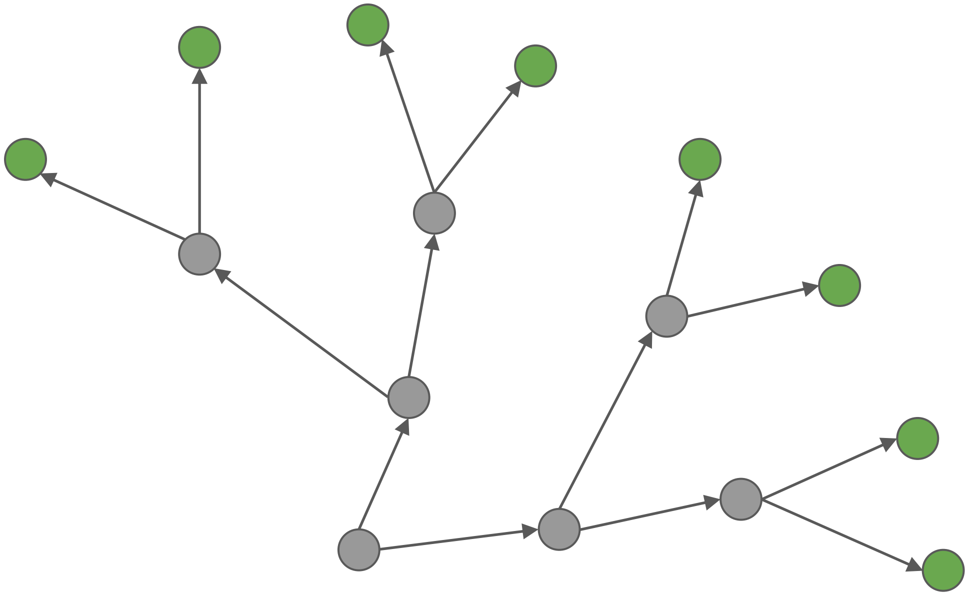
$$O(\log n) = {\sim}1.89$$

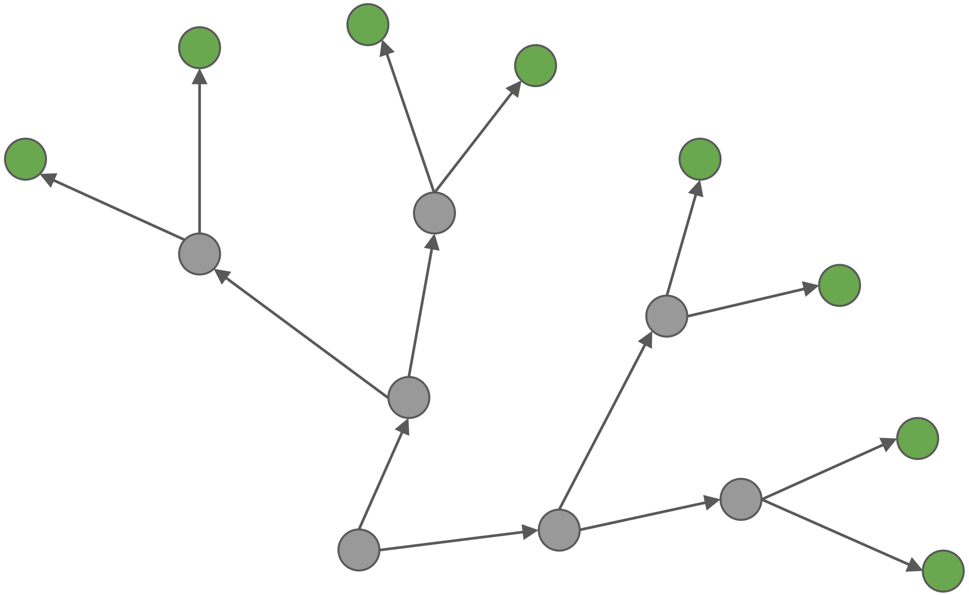
rounds to spread the rumor with everyone

We gossip with a **fixed** number of people

No recovery action is taken

Fast convergence
& eventually consistent

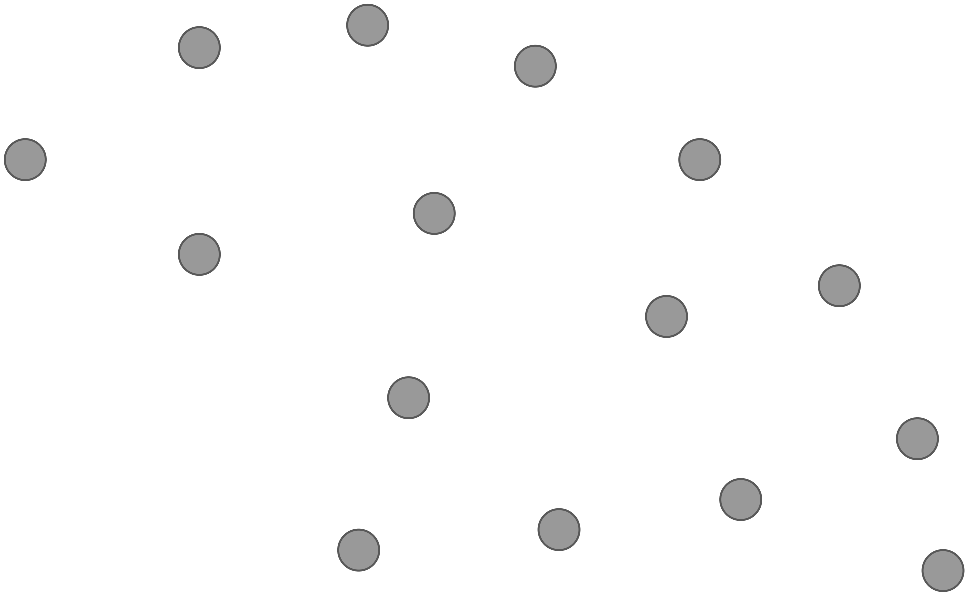


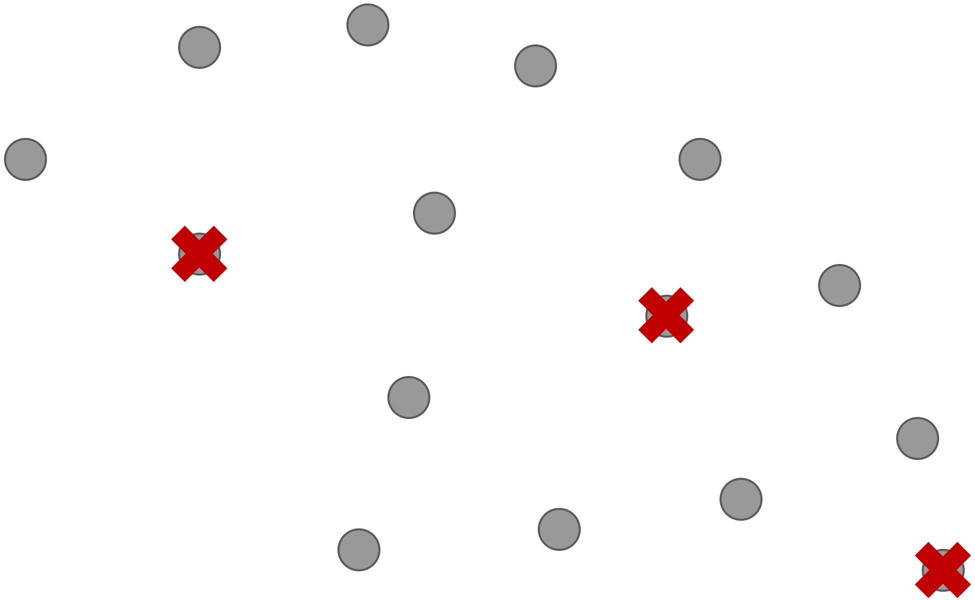


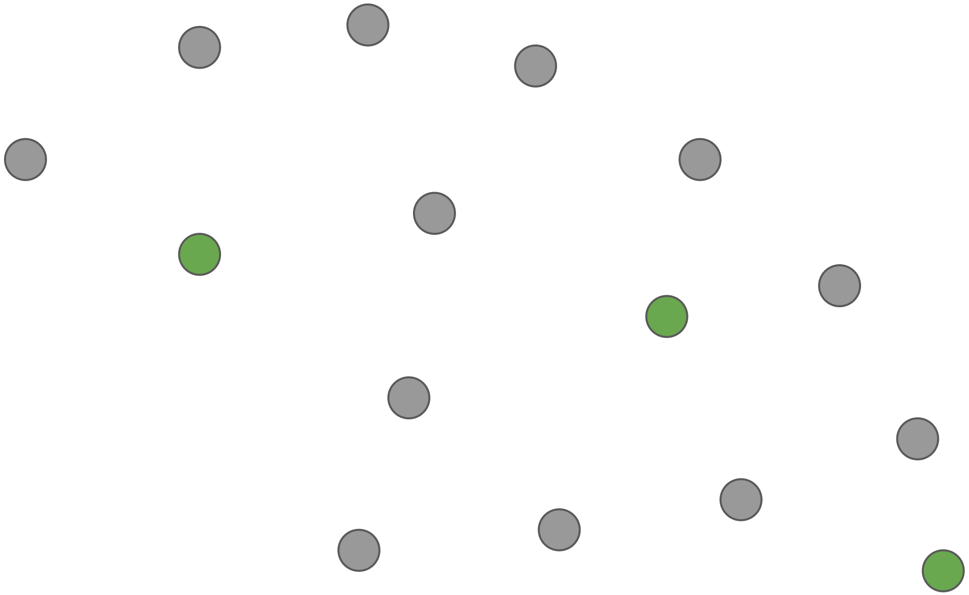
$$2^n$$

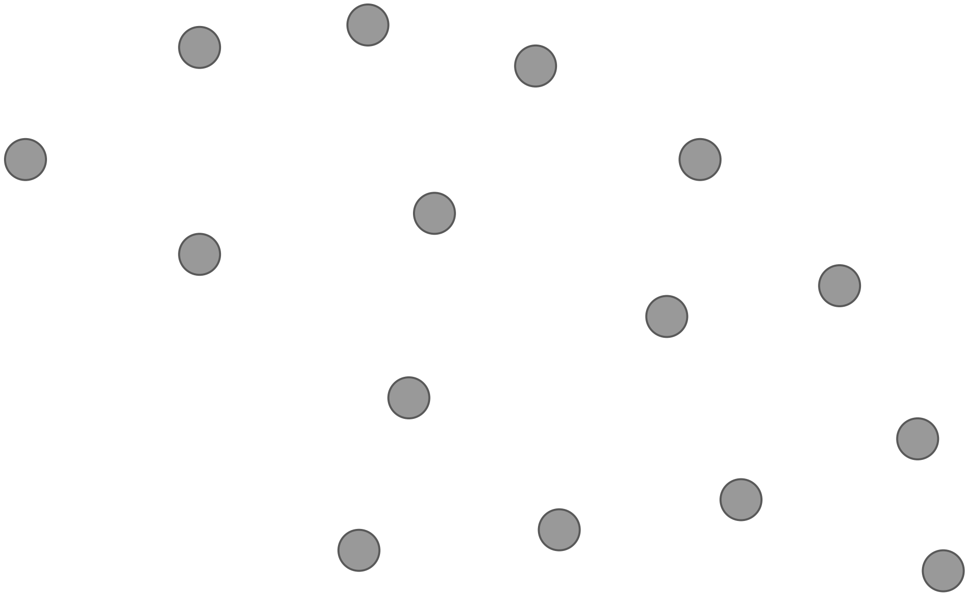
The free pizza rumor spreads exponentially across the Warwick campus

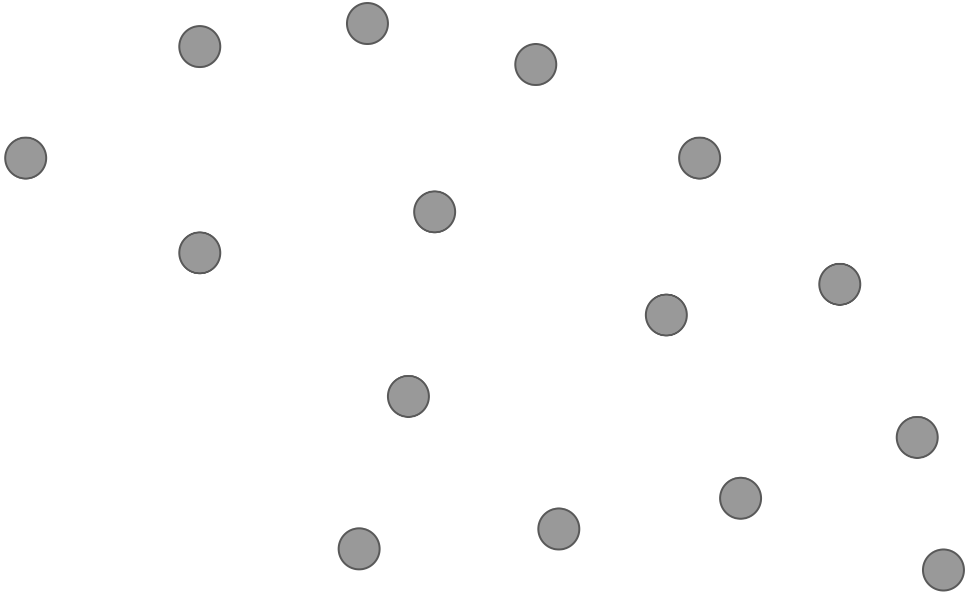
Decentralized





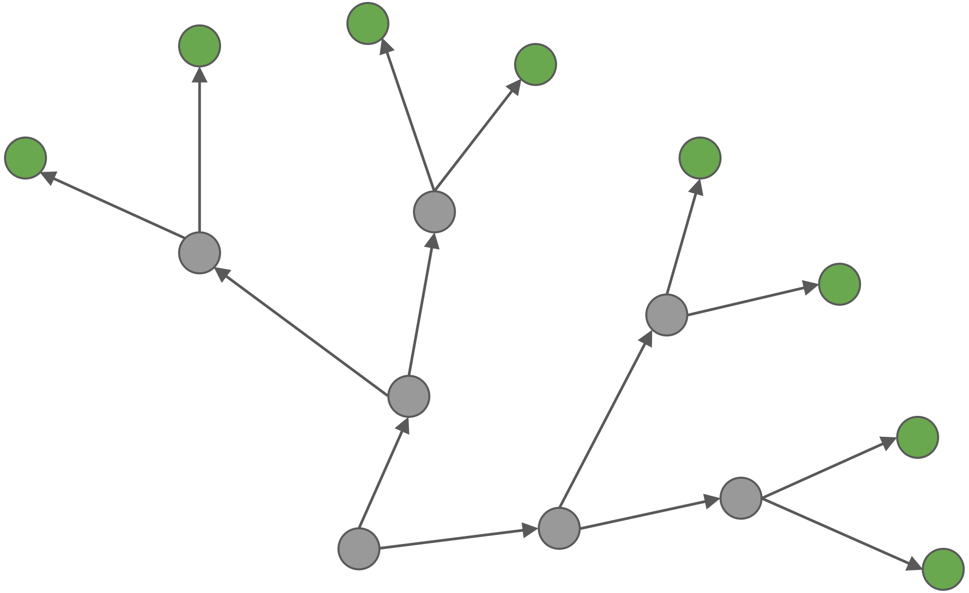


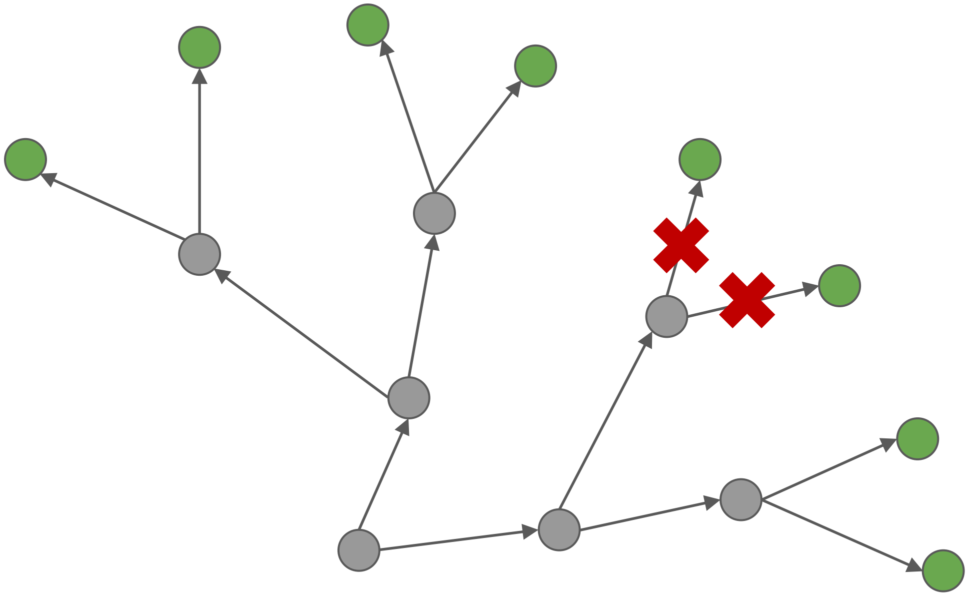


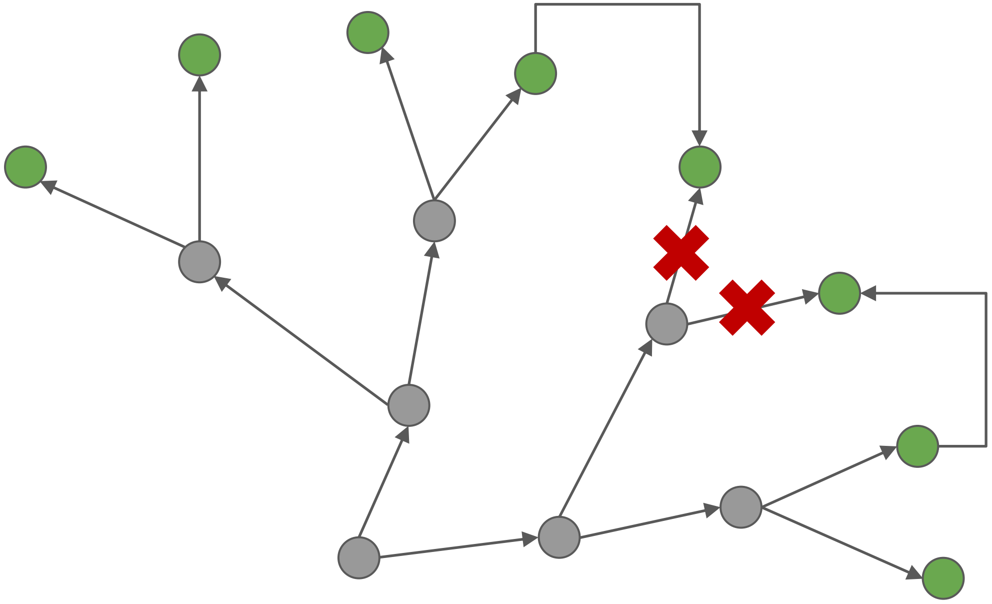


Running the same software, no central coordinators

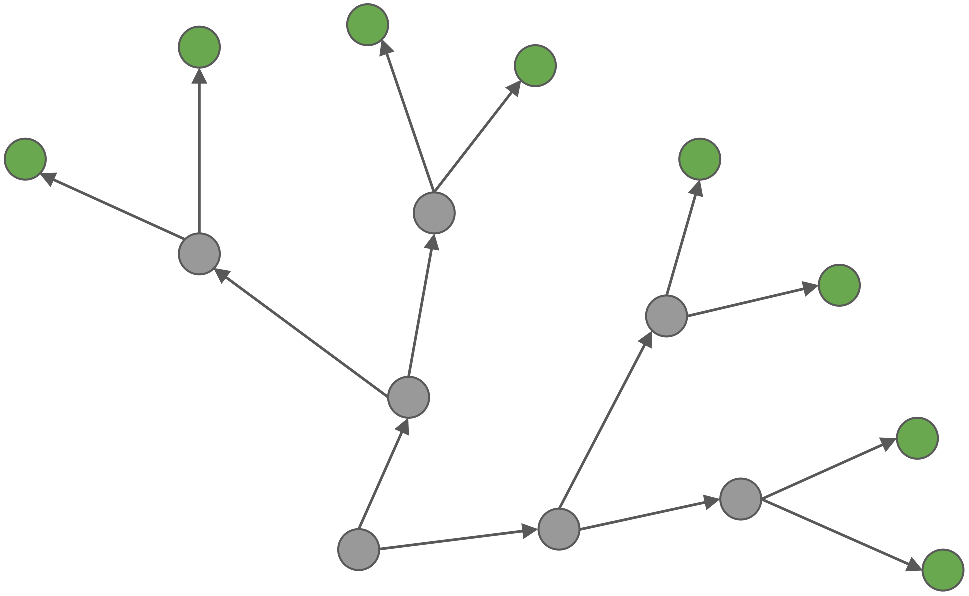
Fault Tolerant

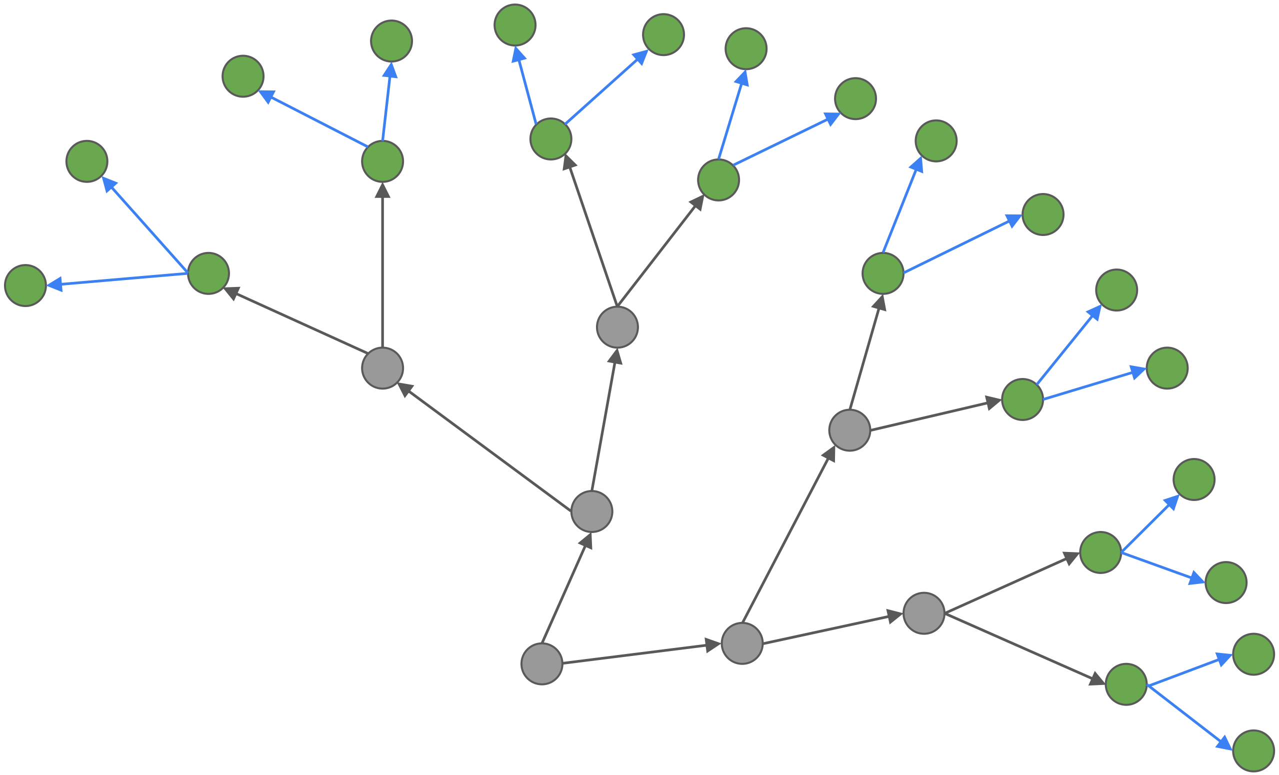


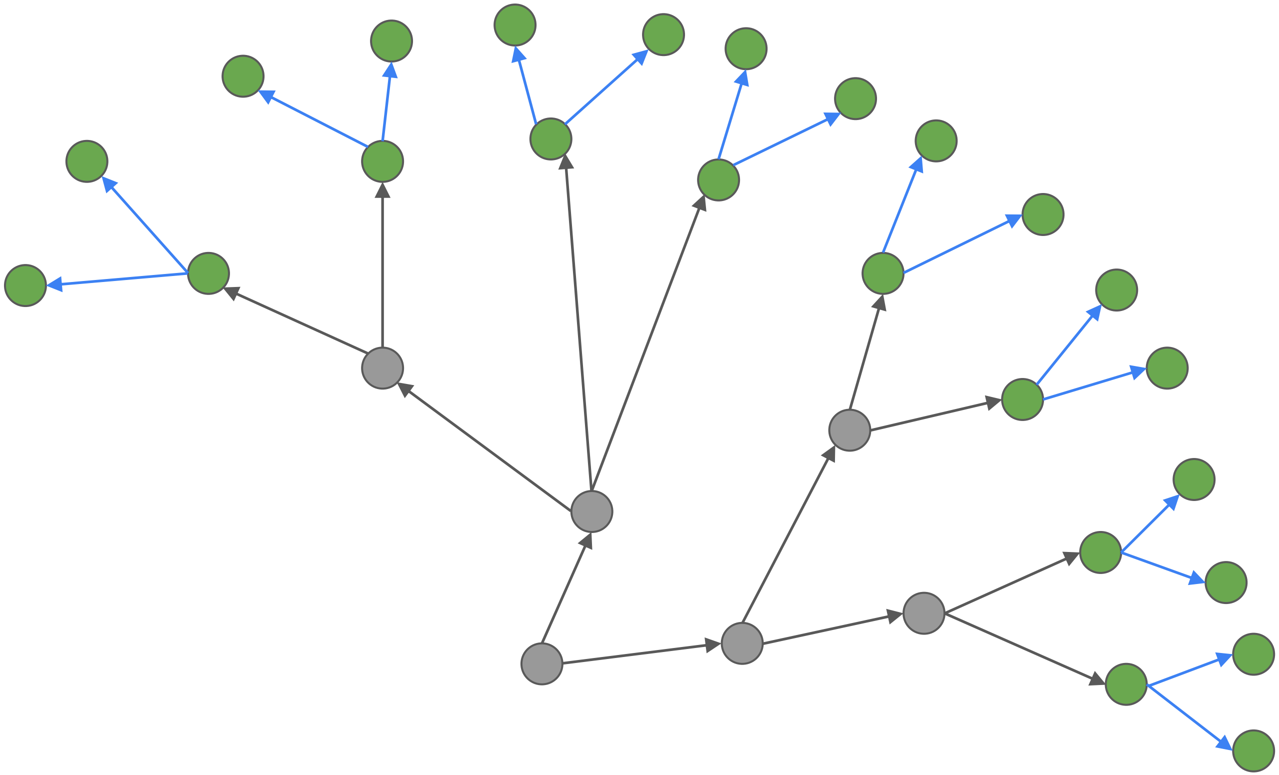




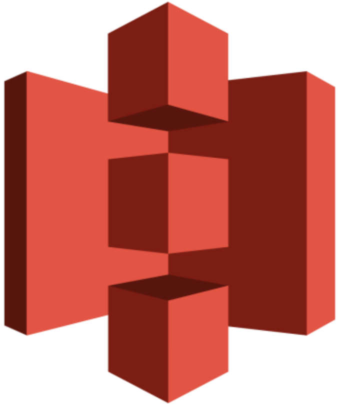
Robust*







Amazon S3



Amazon S3

Scalable service for object-storage offered by AWS

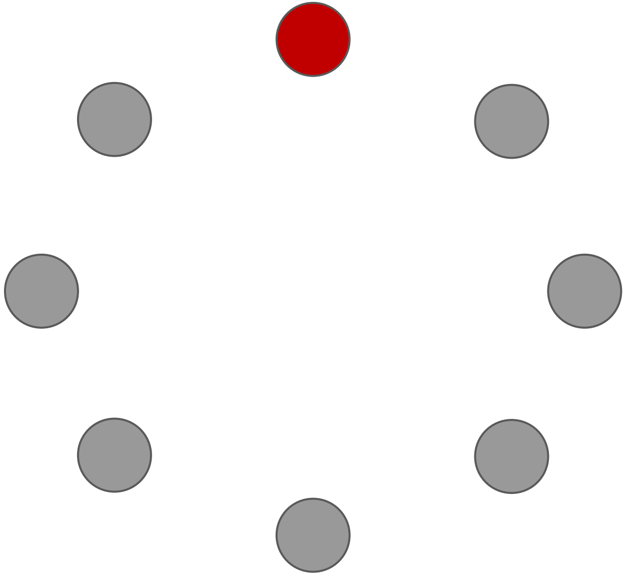
Uses a Gossip protocol to spread the state of each server

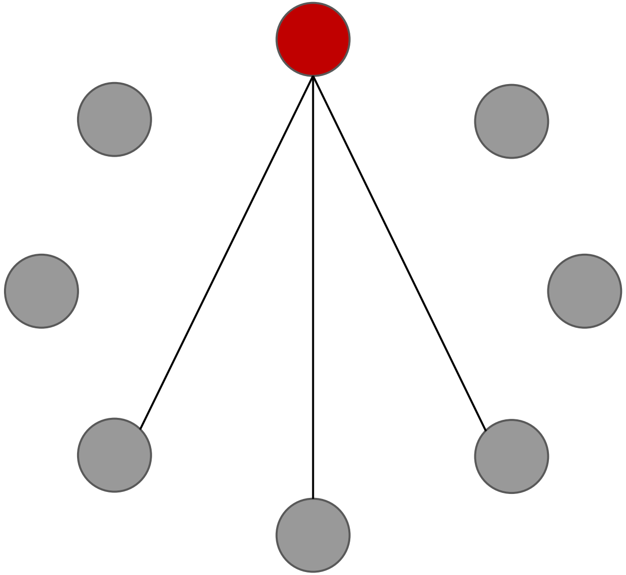
2008 Loss of availability

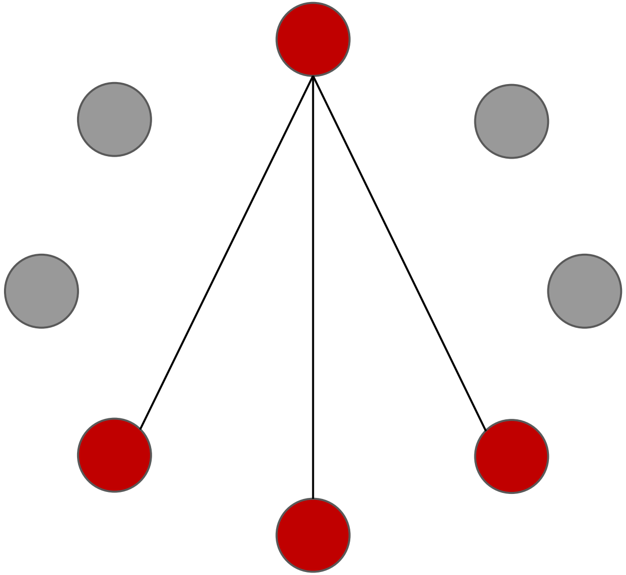
Messages with a corrupted bit were circulated

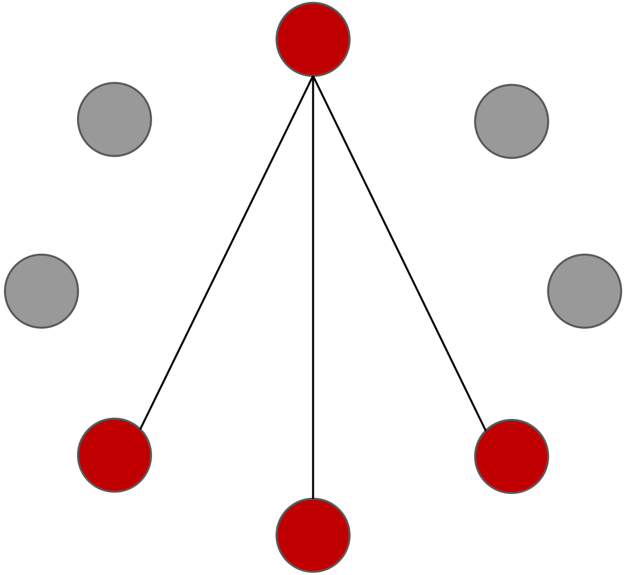
Amazon had to shut down communication between S3 servers

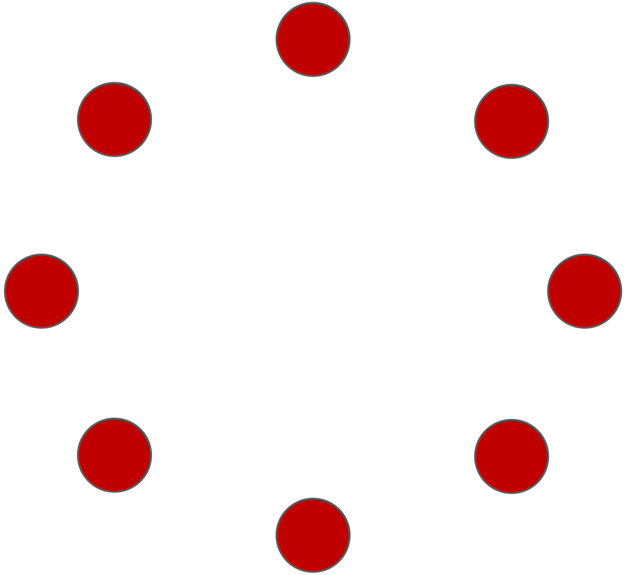
The state was manually fixed to restore the service











Byzantine Faults

Don’t bring a system to a full-stop

Maybe be caused by malicious or flaky peers

How do we know who to speak with?

Keep track of every peer?

Keep track of every peer?

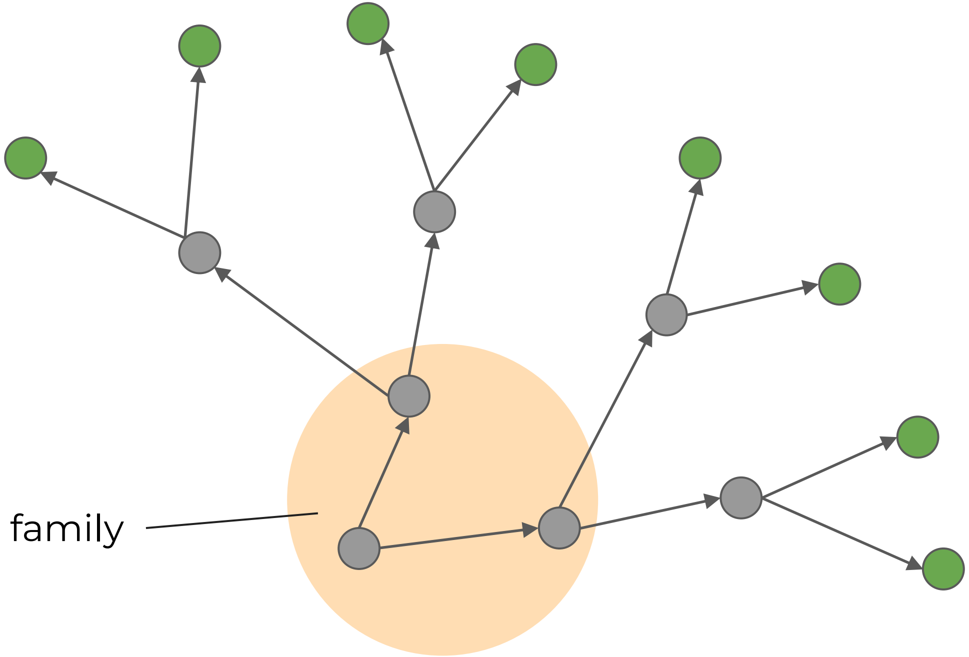
$$O(horrible) = O(n)$$

Not scalable!

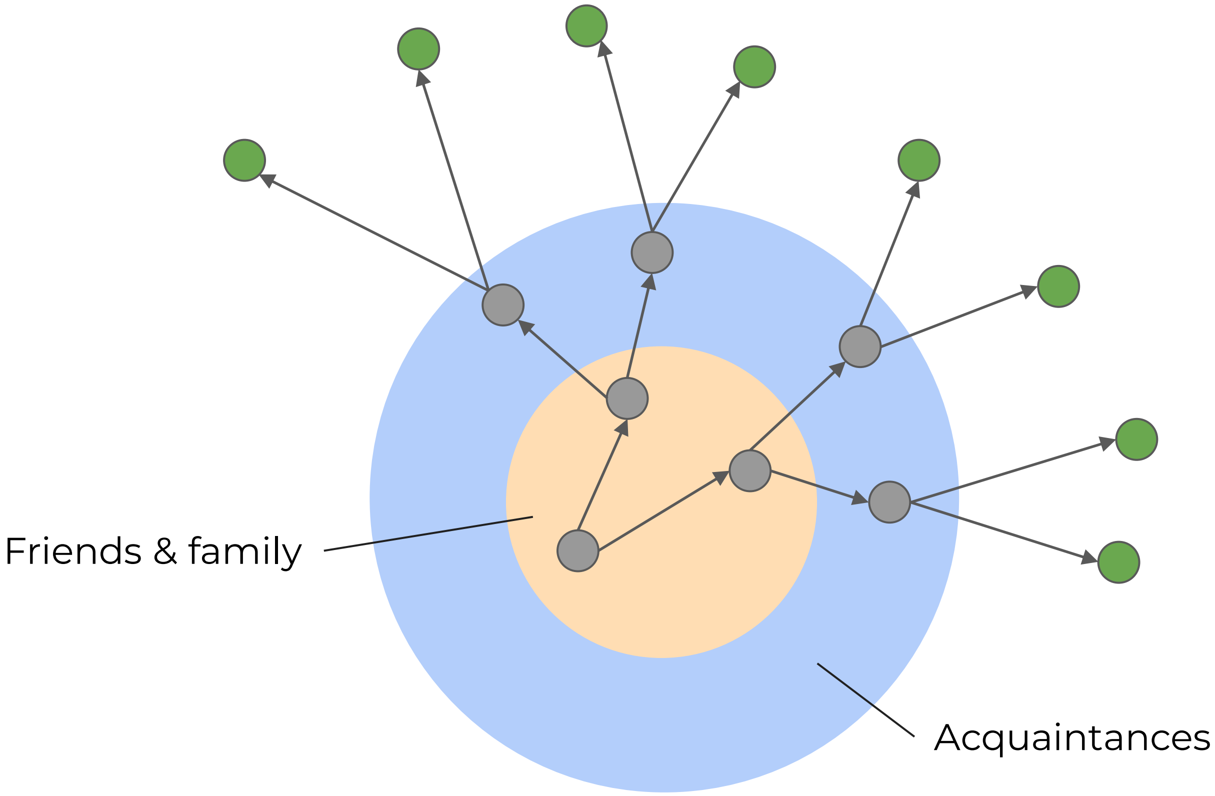
HyParView

A membership protocol for reliable gossip-based broadcast (Joao et al.)

Hybrid **Par**tial **View**



Friends & family

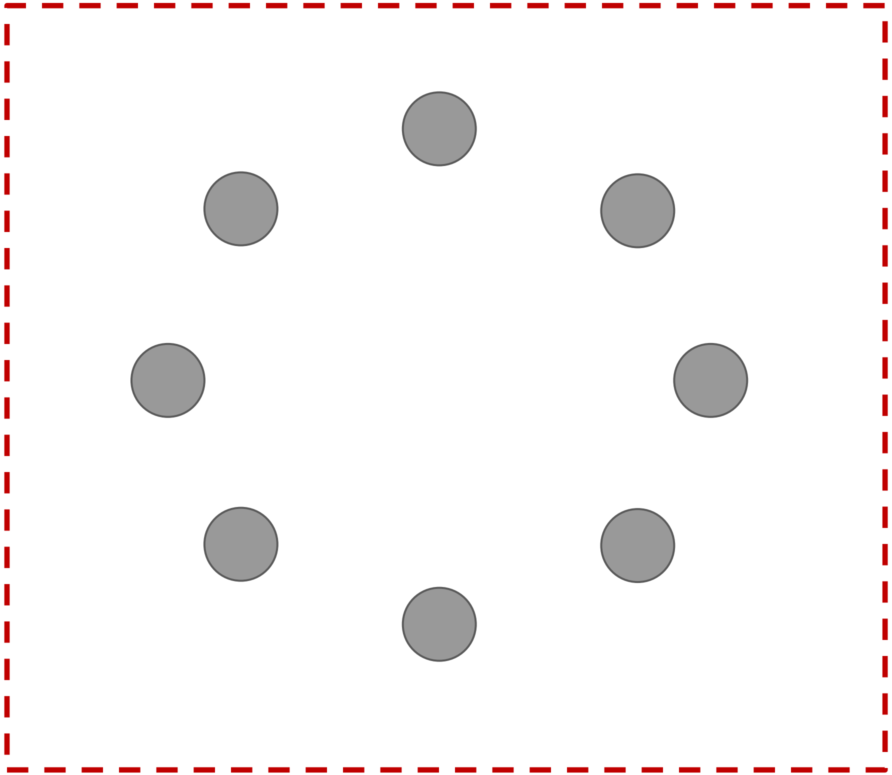

Friends & family
Acquaintances

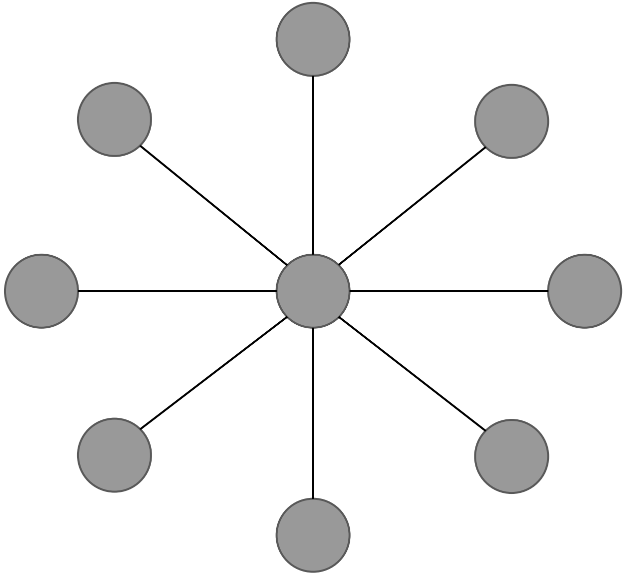
$$O(\log n)$$

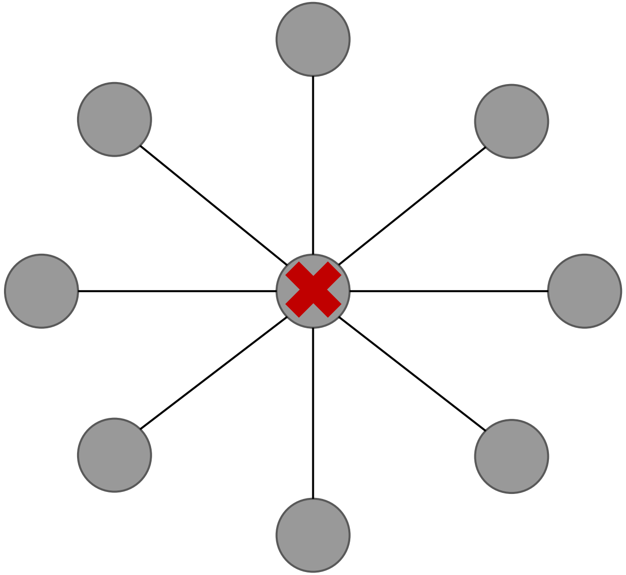
Offers strong resilience to failures of peers
(as high as 90%)

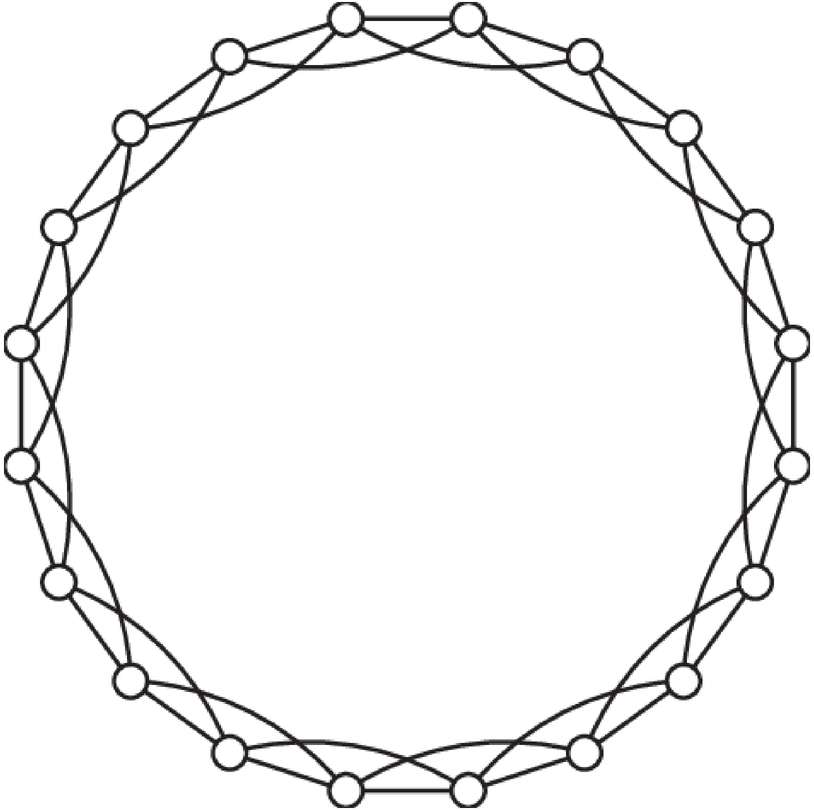
Takes advantage of TCP

Overlay / Topology









Source

Applications

Cluster Membership (Cassandra)
Failure Detection (HashiCorp Serf)
Distributed training of ML models
Compute aggregates
Disseminate metadata (CockroachDB)

Resources

Simulator

Convergence Simulator

Gossip Dissemination – Martin Fowler

Introduction to Gossip – Felix Lopez

HyParView

Amazon S3 Outage (Web Archive)

HyParView: F#

HyParView: Rust

The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations?

Q&A

tomff.com/talks/gossip.pdf