GOSSIP CONCEPT

Based on random exchange of information

Uses fixed size messages, at a fixed rate. This bounds the load gossip can impose on a system, which is valuable in settings like monitoring

But the fixed size and randomness also introduce some issues
GOSSIP IS SLOW TO SPREAD BUT REACHES EVERYONE

Average load is constant, independent of size of the system.

Total network load linear in system size.

Information spreads in \( \log(N) \) time, yet that limit on work per process remains in effect!
GOSSIP CONTINUES TO WORK IF PACKETS ARE DROPPED

Gossip is sent over unreliable messages. This is good because overloaded routers are designed to drop packets as a way to signal congestion.

This makes gossip extremely robust and tolerant of failures or network problems. Information can flow down an exponential number of paths.

But several factors limit gossip to probabilistic robustness... a major one is that because we limit message sizes, data tends to age out.
PRIORITIZATION

Refers to the need to decide which information to put in the message, because of the size limit on messages.

Normally, applications prioritize recent information.

Old data might never reach some participants! Thus, gossip is very robust but not perfectly reliable. In contrast, atomic multicast is totally reliable but might have to pause (or change a group view) if something fails.
MICA SYSTEM

Makes it easy to build layered gossip-based solutions in Java

Your code decides what to put in the gossip messages. Mica handles the implementation of the gossip exchanges

Inside the gossip messages you might find the actual data, or a summary (digest) of data if some items are large.
This was a popular idea for a while, but UDP multicast storms became a concern.

Today we more often combine gossip with unreliable multicast that tunnels over TCP. This idea was proposed in “pbcast” by Guerraoui and Kermarrec. It doesn’t use UDP multicast, yet enables a Bimodal Multicast behavior.

**Expander graph**: a term for a routing graph that will reach every process within log(N) hops. Important if each gossiping node only knows a few neighbors.
We discussed

- Kelips, a gossip-based key-value store. More useful in WAN settings.
- Astrolabe, a gossip-based data mining system
- Bimodal Multicast, for rapid and robust data sharing
- S3 uses gossip to track available space on storage servers
- Many management systems use gossip to track slowly changing state
In this way of thinking we focus on how data spreads in a system rather than thinking about the actual messages it sends.

Because gossip sends at such a slow and steady pace, but information needs to spread exponentially quickly, the information-space perspective actually can tell us something the protocol itself “causes” but might not explicitly “reveal” – a simulation or an analysis of data spread is often needed to understand how a given protocol will spread information.
**INFORMATION SPACE PERSPECTIVE**

Bad Astrolabe aggregation graph: diameter $O(n)$


Official (correct) Astrolabe version: diameter $O(\log(n))$

![Diagram of an official (correct) Astrolabe version with nodes G to O and connections G-F, E-D, C-B, I-J, K-L, M-N, O-P.]