Epidemic Algorithms and Emergent Shape

Ken Birman

On Gossip and Shape

- Why is gossip interesting?
  - Powerful convergence properties?
    - Especially in support of epidemics
  - Mathematical elegance?
    - But only if the system model cooperates
  - New forms of consistency?
    - But here, connection to randomness stands out as a particularly important challenge

On Gossip and Shape

- Convergence around a materialized "graph" or "network topology" illustrates several of these points
  - Contrasts convergence with logical determinism of traditional protocols
  - Opens the door to interesting analysis
  - But poses deeper questions about biased gossip and randomness

Value of convergence

- Many gossip/epidemic protocols converge exponentially quickly
  - Giving rise to "probability 1.0" outcomes
  - Even model simplifications (such as idealized network) are washed away!
  - A rarity: a theory that manages to predict what we see in practice!

Convergence

- I'll use the term to refer to protocols that approach a desired outcome exponentially quickly
- Implies that new information mixes (travels) with at most log(N) delay

Consistency

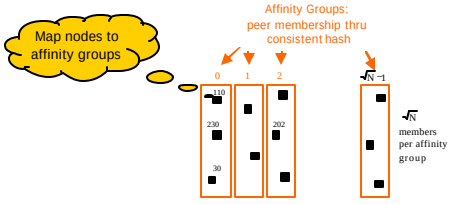
- A term to capture the idea that if A and B could compare their states, no contradiction is evident
  - In systems with "logical" consistency, we say things like "A's history is a closed prefix of B's history under causality"
  - With probabilistic systems we seek exponentially decreasing probability (as time elapses) that A knows "x" but B doesn't
- Gossip systems are usually probabilistic

Convergent consistency

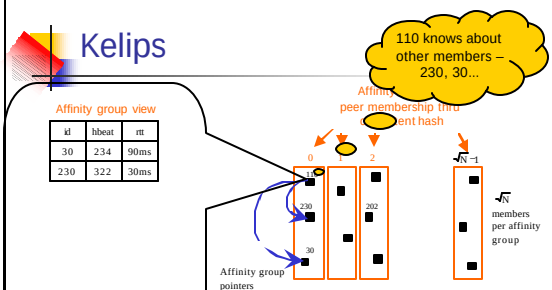
- To illustrate our point, contrast Cornell's Kelips system with MIT's Chord
  - Chord: The McDonald's of DHTs
  - Kelips: DHT by Birman, Gupta, Linga.
  - Prakash Linga is extending Kelips to support multi-dimensional indexing, range queries, self-rebalancing
- Kelips is convergent. Chord isn't

Kelips

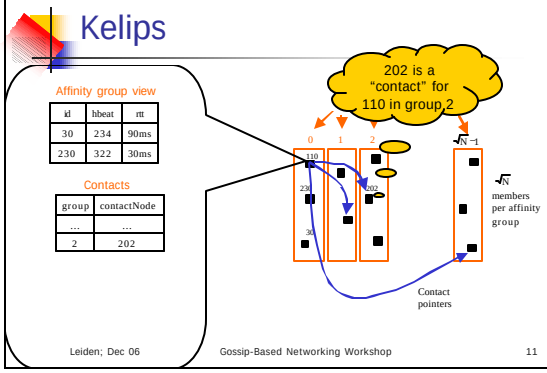
Kelips



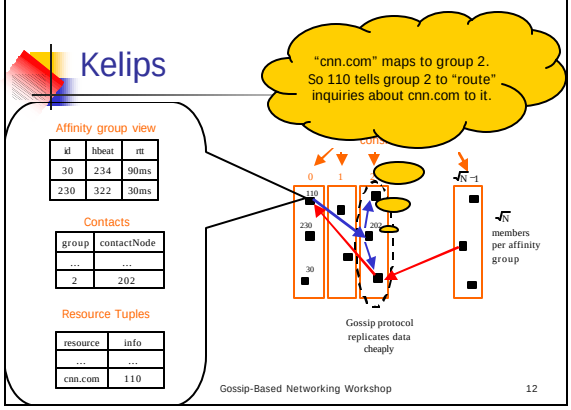
Kelips



Kelips



Kelips



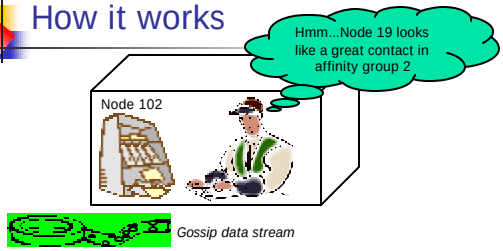
How it works

- Kelips is *entirely* gossip based!
  - Gossip about membership
  - Gossip to replicate and repair data
  - Gossip about "last heard from" time used to discard failed nodes
- Gossip "channel" uses fixed bandwidth
  - ... fixed rate, packets of limited size

How it works

- Basically...
  - A stream of gossip data passes by each node, containing information on various kinds of replicated data
  - Node "sips" from the stream, for example exchanging a questionable contact in some group for a better one
    - Based on RTT, "last heard from" time, etc

How it works



- Heuristic: periodically ping contacts to check liveness, RTT... swap so-so ones for better ones.

Convergent consistency

- Exponential wave of infection overwhelms disruptions
  - Within logarithmic time, reconverges
  - Data structure *emerges* from gossip exchange of data.
  - Any connectivity at all suffices....

... subject to a small caveat

- To bound the load, Kelips
  - Gossips at a constant rate
  - Limits the size of packets
  - ...Kelips has limited incoming "info rate"
- Behavior when the limit is continuously exceeded is not well understood.

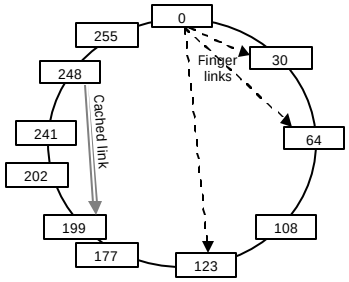
What about Chord?

- Chord is a "true" data structure mapped into the network
  - Ring of nodes (hashed id's)
  - Superimposed binary lookup trees
  - Other cached "hints" for fast lookups
- Chord is *not* convergently consistent

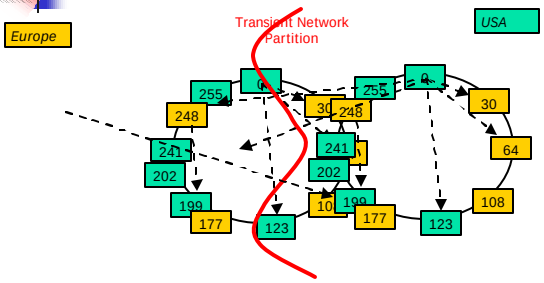
... so, who cares?

- Chord lookups can fail... and it suffers from high overheads when nodes churn
  - Loads surge just when things are already disrupted... quite often, because of loads
  - And can't predict how long Chord might remain disrupted once it gets that way
- Worst case scenario: *Chord can become inconsistent and stay that way*

Chord picture

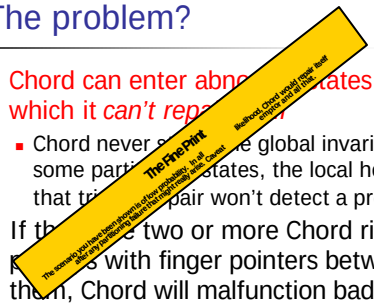


Chord picture



The problem?

- Chord can enter abnormal states in which it *can't repair itself*
  - Chord never s[illegible] global invariant... in some parti[illegible]states, the local heuristics that tr[illegible]pair won't detect a problem
- If th[illegible] two or more Chord rings, [illegible]s with finger pointers between them, Chord will malfunction badly!



So... can Chord be fixed?

- Epichord doesn't have this problem
  - Uses gossip to share membership data
  - If the rings have any contact with each other, they will heal
- Similarly, Kelips would heal itself rapidly after partition
- Gossip is a remedy for what ails Chord!

Insight?

- Perhaps large systems shouldn't try to "implement" conceptually centralized data structures!
- Instead seek emergent shape using decentralized algorithms

Emergent shape

- We know a lot about a related question
  - Given a connected graph, cost function
  - Nodes have bounded degree
  - Use a gossip protocol to swap links until some
- Another
  - Given a gossip overlay, improve it by selecting "better" links (usually, lower RTT)

Example: The "Anthill" framework of Alberto Montresor, Ozalp Babaoglu, Hein Meling and Francesco Russo

Problem description

- Given a description of a data structure (for example, a *balanced* tree)
  - ... design a gossip protocol such that the system will rapidly converge towards that structure even if disrupted
  - Do it with bounced per-node message rates, sizes (network load less important)
- Use aggregation to test tree quality?

Connection to self-stabilization

- Self-stabilization theory
  - Describe a system and a desired property
  - Assume a failure in which code remains correct but node states are corrupted
  - Proof obligation: property reestablished within bounded time
- *Kelips is self-stabilizing. Chord isn't.*

Let's look at a second example

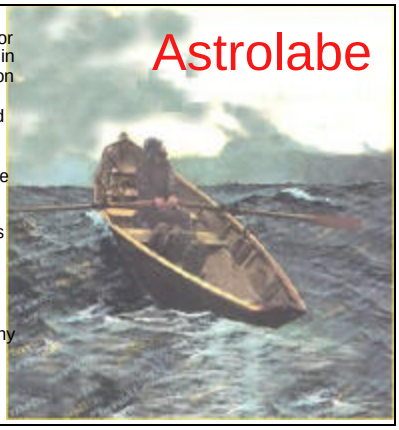
- Astrolabe system uses a different emergent data structure – a tree
- Nodes are given an initial location – each knows its "leaf domain"
- Inner nodes are elected using gossip and aggregation

Astrolabe

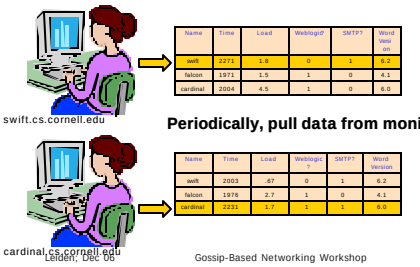
- Intended as help for applications adrift in a sea of information
- Structure emerges from a randomized gossip protocol
- This approach is robust and scalable even under stress that cripples traditional systems

Developed at RNS, Cornell

- By Robbert van Renesse, with many others helping...
- Today used extensively within Amazon.com



Astrolabe is a flexible monitoring overlay



Name	Time	Load	Weblogic?	SMTP?	Word Version
swift	2271	1.8	0	1	6.2
falcon	1971	1.5	1	0	4.1
cardinal	2004	4.5	1	0	6.0

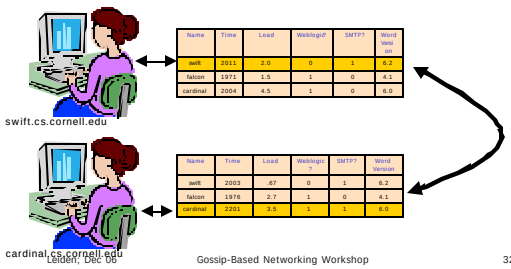
Periodically, pull data from monitored systems

Name	Time	Load	Weblogic?	SMTP?	Word Version
swift	2003	.67	0	1	6.2
falcon	1976	2.7	1	0	4.1
cardinal	2231	1.7	1	1	6.0

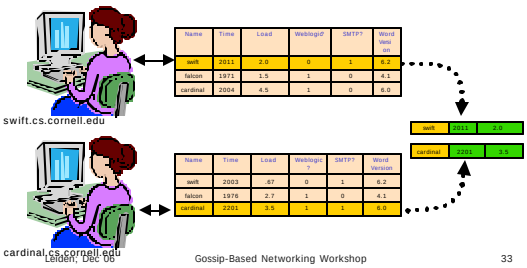
Astrolabe in a single domain

- Each node owns a single tuple, like the management information base (MIB)
- Nodes discover one-another through a simple broadcast scheme (“anyone out there?”) and gossip about membership
  - Nodes also keep replicas of one-another’s rows
  - Periodically (uniformly at random) merge your state with some else…

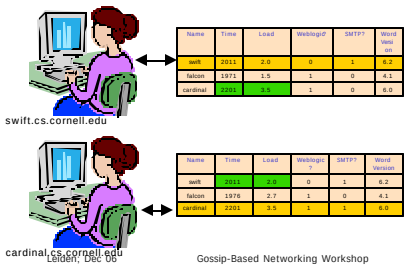
State Merge: Core of Astrolabe epidemic



State Merge: Core of Astrolabe epidemic



State Merge: Core of Astrolabe epidemic



Observations

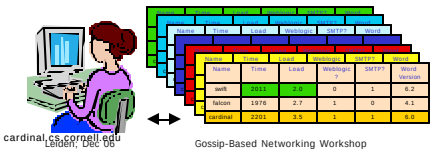
- Merge protocol has constant cost
  - One message sent, received (on avg) per unit time.
  - The data changes slowly, so no need to run it quickly – we usually run it every five seconds or so
  - Information spreads in O(log N) time
- But this assumes bounded region size
  - In Astrolabe, we limit them to 50-100 rows

Big systems…

- A big system could have *many* regions
  - Looks like a pile of spreadsheets
  - A node only replicates data from its neighbors within its own region

Scaling up... and up...

- With a stack of domains, we don't want every system to "see" every domain
  - Cost would be huge
- So instead, we'll see a summary



Astrolabe builds a hierarchy using a P2P protocol that "assembles the puzzle" without any servers

Dynamically changing query output is visible system-wide

SQL query "summarizes" data

Large scale: "fake" regions

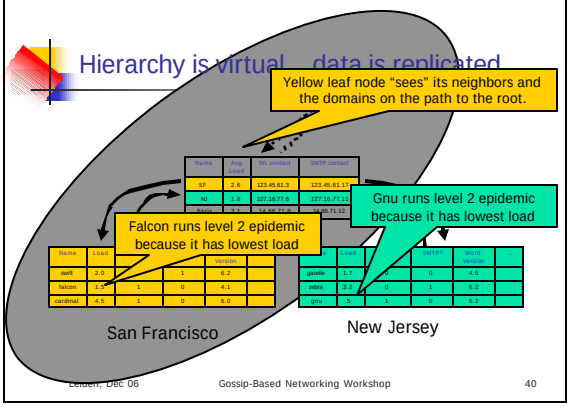
- These are
  - Computed by queries that summarize a whole region as a single row
  - Gossiped in a read-only manner within a leaf region
- But who runs the gossip?
  - Each region elects "k" members to run gossip at the next level up.
  - Can play with selection criteria and "k"

Hierarchy is virtual... data is replicated

Yellow leaf node "sees" its neighbors and the domains on the path to the root.

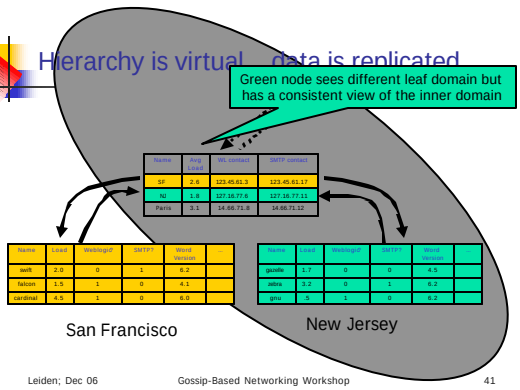
Falcon runs level 2 epidemic because it has lowest load

Gnu runs level 2 epidemic because it has lowest load



Hierarchy is virtual... data is replicated

Green node sees different leaf domain but has a consistent view of the inner domain



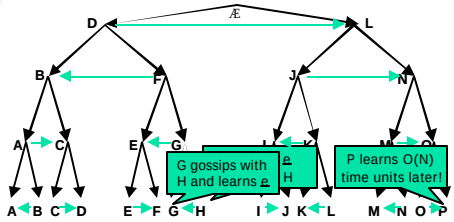
Worst case load?

- A small number of nodes end up participating in $O(\log_{fanout} N)$ epidemics
  - Here the fanout is something like 50
  - In each epidemic, a message is sent and received roughly every 5 seconds
- We limit message size so even during periods of turbulence, no message can become huge.
  - Instead, data would just propagate slowly
  - Robbert has recently been working on this case

Emergent shapes

- Kelips: Nodes start with a-priori assignment to affinity groups, end up with a superimposed pointer structure
- Astrolabe: Nodes start with a-priori leaf domain assignments, build the tree
- *What other kinds of data structures can be achieved with emergent protocols?*

Van Renesse's dreadful aggregation tree



What went wrong?

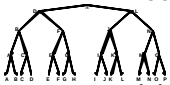
- In Robbert's horrendous tree, each node has equal "work to do" but the information-space diameter is larger!
- Astrolabe benefits from "instant" knowledge because the epidemic at each level is run by someone elected from the level below

Insight: Two kinds of shape

- We've focused on the aggregation tree
- But in fact should also think about the information flow tree

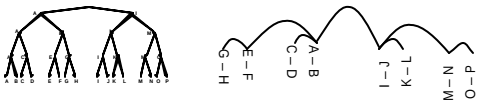
Information space perspective

- Bad aggregation graph: diameter O(n)



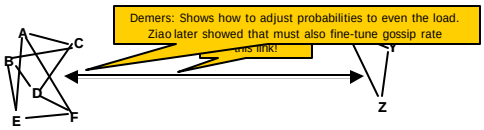
H – G – E – F – B – A – C – D – L – K – I – J – N – M – O – P

- Astrolabe version: diameter O(log(n))



Gossip and bias

- Often useful to "bias" gossip, particularly if some links are fast and others are very slow



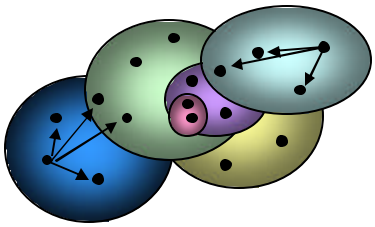
How does bias impact information-flow graph

- Earlier, all links were the same
- Now, some links carry
  - Less information
  - And may have longer delays
- Open question: Model bias in information flow graphs and explore implications

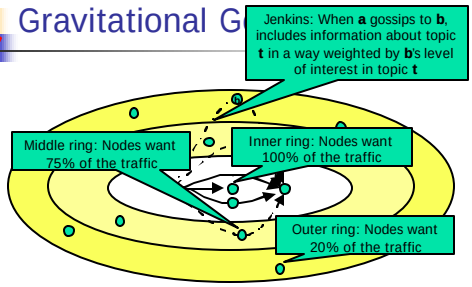
Gossip and bias

- Biased systems adjust gossip probabilities to accomplish some goal
  - Kate Jenkins: "Gravitational gossip" (ICDCS '01) illustrates how far this can be carried
    - A world of multicast groups in which processes subscribe to x % of the traffic in each group
    - Kate showed how to set probabilities from a set of such subscriptions… resulting protocol was intuitively similar to a simulation of a gravitational well…

Gravitational Gossip



Gravitational G



Questions about bias

- When does the biasing of gossip target selection break analytic results?
  - Example: Alves and Hopcroft show that with fanout too small, gossip epidemics can die out, logically partitioning a system
- Question: Can we relate the question to flooding on an an expander graph?

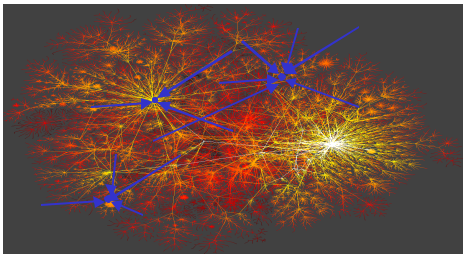
… more questions

- Notice that Astrolabe forces participants to agree on what the aggregation hierarchy should contain
  - In effect, we need to "share interest" in the aggregation hierarchy
- This allows us to bound the size of messages (expected constant) and the rate (expected constant per epidemic)

The question

- Could we design a gossip-based system for "self-centered" state monitoring?
- Each node poses a query, Astrolabe style, on the state of the system
  - We dynamically construct an overlay for each of these queries
  - The "system" is the union of these overlays

Self-centered monitoring



Self-centered queries...

- Offhand, looks like a bad idea
  - If everyone has an independent query
  - And everyone is iid in all the obvious ways
  - Than everyone must invest work proportional to the number of nodes monitored by each query
- In particular if queries touch O(n) nodes, global workload is $O(n^2)$

Aggregation

- ... but in practice, it seems unlikely that queries would look this way
- More plausible is something Zipf-like
  - A few queries look at broad state of system
  - Most look at relatively few nodes
  - And a small set of aggregates might be shared by the majority of queries
- Assuming this is so, can one build a scalable gossip overlay / monitoring infrastructure?

Questions about shape

- Can a system learn its own shape?
  - Obviously we can do this by gossiping the full connectivity graph
  - But are there ways to gossip constant amounts of information at a constant rate and still learn a reasonable approximation to the topology of the system?
- Related topic: "sketches" in databases

... yet another idea

- Today, "structural" gossip protocols usually
  - Put nodes into some random initial graph
  - Nodes know where they would "like" to be
- Biological systems:
  - Huge collections of nodes (cells) know roles
  - Then they optimize (against something: what?) to better play those roles
  - Create gossip systems for very large numbers of nodes that behave like biological systems?

Emergent Shape: Topics

- Consistency models
- Necessary conditions for convergent consistency
- Role of randomness
- Implications of bias
- Emergent structures
- Self-centered aggregation
- Bandwidth-limited systems; “sketches”
- Using aggregation to fine-tune a shape with constant costs