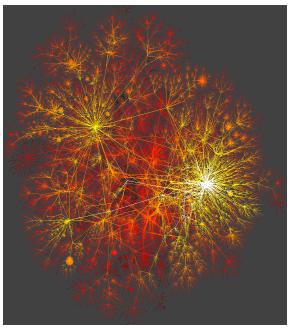
CS514: Intermediate Course in Computer Systems

Lecture 31: April 7, 2003
Astrolabe

The Internet



Source: Burch and Cheswick

Massive scale.
Constant flux

Demand for more "autonomic", intelligent behavior

CS514

- Human beings constantly adapt as their environment changes
  - You bike up hill… start to breath hard and sweat. At the top, cool off and catch your breath
  - It gets cold so you put on a sweater
- But computer systems tend to be rigid and easily disabled by minor events

Typical examples

CS514

- IBM finds that many Web Services systems are perceived as unreliable
  - End-user gets erratic response time
  - Client could be directed to the wrong server site
- But the Web Sphere software isn't at fault!
  - Usually these problems arise from other systems to which WS is connected
  - A snarl of spaghetti sits behind the front end

A tale of woe

CS514

- Human operators lack tools to see state of system
  - Can't easily figure out what may be going wrong
- In fact operators cause as much as 70% of all Web-related downtime!
- And they directly trigger 35% of crashes

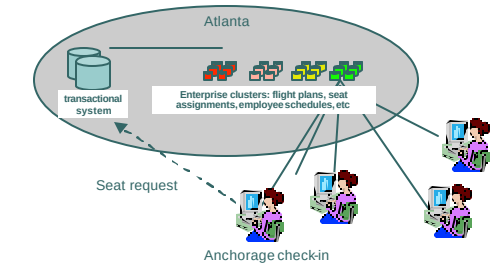
Sample tale of woe

CS514

- Delta Airlines maintains the "Delta Nerve Center" in Atlanta
  - It has an old transactional mainframe for most operations
  - Connected to this are a dozen newer relational database applications on clusters
  - These talk to ~250,000 client systems using a publish-subscribe system

Delta Architecture

CS514



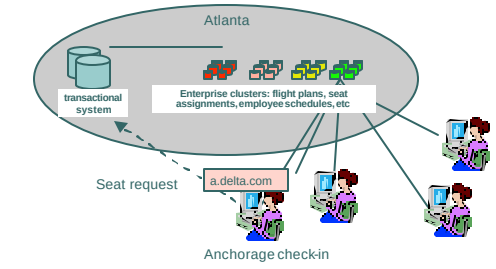
Anchorage goes down

CS514

- Problem starts in Atlanta
  - System operator needs to move a key subsystem from computer A to B
  - But the DNS is slow to propagate the change (maybe 48 to 72 hours)
- Anchorage still trying to talk to A
  - So a technician fixes the problem by typing in the IP address of B
  - Gets hired by United with a hefty raise

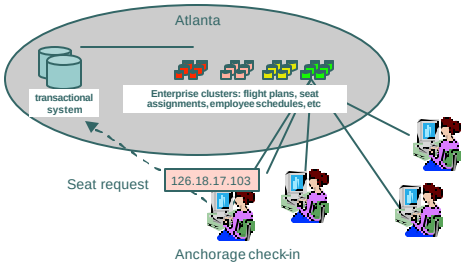
Delta Architecture

CS514



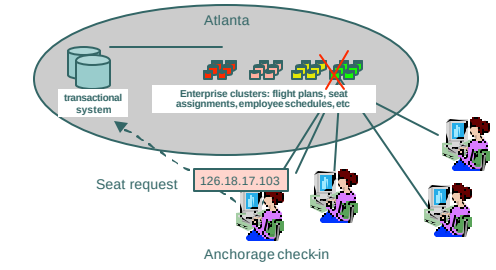
Delta Architecture

CS514



Delta Architecture

CS514



Six months later…

CS514

- Time goes by and now we need to move that service again Anchorage crashes again…
  - But this time nobody can figure out why!
  - Hunting for references to the service or even to B won't help
  - Need to realize that the actual IP address of B is wired into the application now
  - Nobody remembers what the technician did or why he did it!

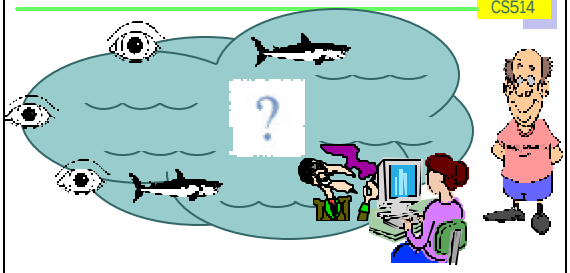
Homeland Defense and Military Data Mining

CS514

- Issue is even worse for a new generation of "smart" data mining applications
- Their role is to look for something on behalf of the good guys
  - Look for images of tall thin guys with long white beards on horseback

The Dilemma of the Intelligent Application

CS514



Intelligent application is separated from sensors by the network It lives in a sea of shifting state, changing conditions and threats…

Two options

CS514

- We could ship all the data to the analyst's workstation
  - E.g. ship every image, in real-time
  - If N machines gather I images per second, and we have B bytes per image, the load on the center system grows with N*I*B. With A analysts the load on the network grows as N*I*B*A
- Not very practical.

Two options

CS514

- Or, we could ship the work to the remote sensor systems
  - They do all the work "out there", so each just searches the images it is capturing
  - Load is thus quite low
- But how could we build such a system?

Sentient Distributed Systems

CS514

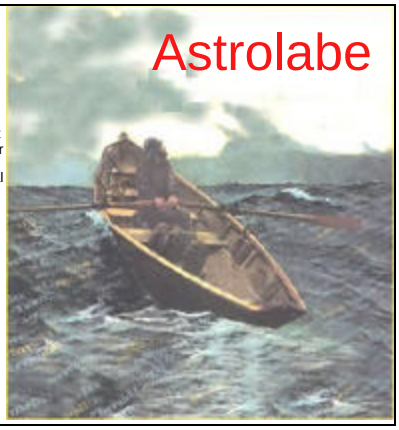
- Can we build a new generation of smart middleware in support of these new intelligent systems?
  - Middleware that *perceives* the state of the network
  - It can *represent this knowledge* in a form smart applications can exploit
  - Although built from large numbers of rather dumb components the *emergent behavior is intelligent.* These applications are more robust, more secure, more responsive than any individual component
  - When something unexpected occurs, they can *diagnose the problem* and trigger a *coordinated distributed response*
  - They *repair themselves* after damage

Astrolabe

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

Developed at Cornell

- By Robbert van Renesse, with many others helping…
- Just an example of the kind of solutions we need
- Astrolabe is a form of knowledge representation for sentient networks



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

CS514

Dynamically changing query output is visible system-wide

SQL query "summarizes" data

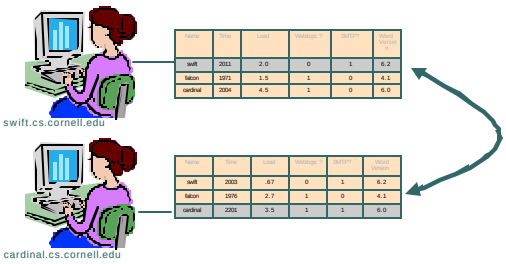
Astrolabe in a single domain

CS514

- 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…

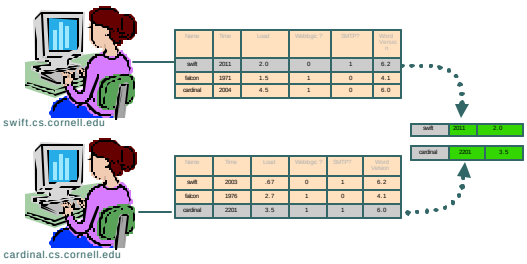
State Merge: Core of Astrolabe epidemic

CS514



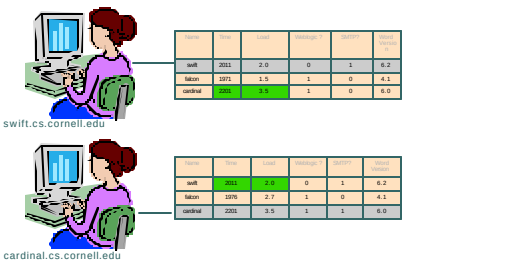
State Merge: Core of Astrolabe epidemic

CS514



State Merge: Core of Astrolabe epidemic

CS514



Observations

CS514

- 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 system will have many regions

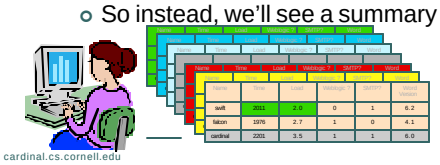
CS514

- Astrolabe usually configured by a manager who places each node in some region, but we are also playing with ways to discover structure automatically
  - For example could use Paul's ID Maps or the new CMU GNP concept
- 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…

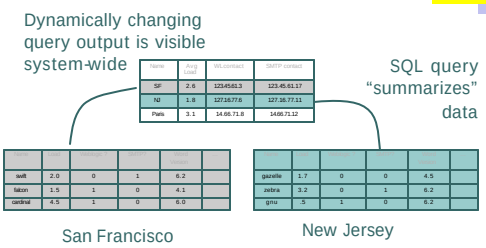
CS514

- 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

CS514



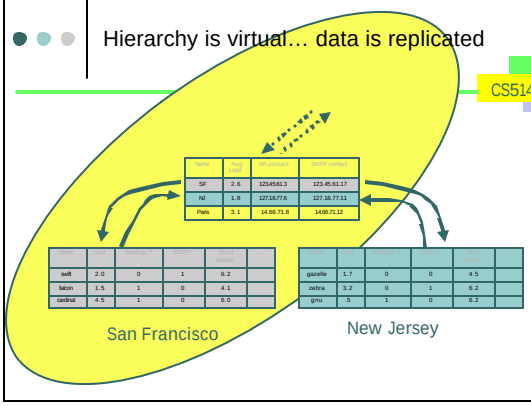
Large scale: "fake" regions

CS514

- 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"

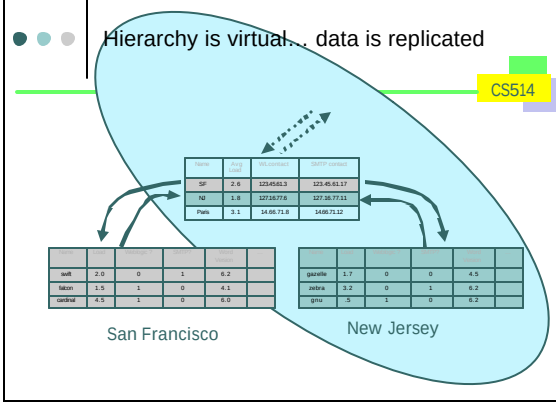
Hierarchy is virtual… data is replicated

CS514



Hierarchy is virtual… data is replicated

CS514



Worst case load?

CS514

- 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
  - Haven't really looked hard at this case

What makes Astrolabe a good fit

CS514

- Notice how hierarchical database abstraction "emerges" without ever being physically represented on any single machine
  - Moreover, this abstraction is very robust
  - It scales well… localized disruptions won't disrupt the system state… consistent in eyes of varied beholders
  - Yet individual participant runs a nearly trivial peer-to-peer protocol
- Supports distributed data aggregation, data mining. Adaptive and self-repairing…

Data Mining

CS514

- In client-server systems we usually
  - Collect the data at the server
  - Send queries to it
- With Astrolabe
  - Send query (and CFC) to edge nodes
  - They pull in desired data and query it
  - User sees a sort of materialized result

Pros and Cons

CS514

- Pros:
  - As you look "up" the hierarchy the answer you see can differ for different users ("where can I get some gas?")
  - Parallelism makes search fast
- Cons:
  - Need to have agreement on what to put into the aggregations
  - Everyone sees the same hierarchy

Other such abstractions

CS514

- Scalable probabilistically reliable multicast based on P2P (peer-to-peer) epidemics
- Some of the work on P2P indexing structures and file storage
- Stoica's new ideas for a P2P Internet Indirection Infrastrastructure

Solutions that share properties

CS514

- Scalable
- Robust against localized disruption
- Have emergent behavior we can reason about, exploit in the application layer
- Think of the way a hive of insects organizes itself or reacts to stimuli. There are many similarities

Revisit our goals

CS514

- Are these potential components for sentient systems?
  - Middleware that *perceives* the state of the network
  - It *represent this knowledge* in a form smart applications can exploit
  - Although built from large numbers of rather dumb components the *emergent behavior is intelligent.* These applications are more robust, more secure, more responsive than any individual component
  - When something unexpected occurs, they can *diagnose the problem* and trigger a *coordinated distributed response*
  - They *repair themselves* after damage
- We seem to have the basis from which to work!

Brings us full circle

CS514

- Our goal should be a new form of very stable "sentient middleware"
- Have we accomplished this goal?
  - Probabilistically reliable, scalable primitives
  - They solve many problems
  - Gaining much attention now from industry, academic research community
- Fundamental issue is skepticism about peer-to-peer as a computing model

Conclusions?

CS514

- We're at the verge of a breakthrough – networks that behave like sentient infrastructure on behalf of smart applications
- Could open the door to big advances
- But industry has yet to deploy these ideas and may think about them for a long time before doing so