CS514: Intermediate Course in Operating Systems

Professor Ken Birman
Vivek Vishnumurthy: TA

Perspectives on Computing Systems and Networks

- CS314: Hardware and architecture
- CS414: Operating Systems
- CS513: Security for operating systems and apps
- CS514: Emphasis on “middleware”: networks, distributed computing, technologies for building reliable applications over the middleware
- CS519: Networks, aimed at builders and users
- CS614: A survey of current research frontiers in the operating systems and middleware space
- CS619: A reading course on research in networks

Styles of Course

- CS514 tries to be practical in emphasis:
  - We look at the tools used in real products and real systems
  - The focus is on technology one could build / buy
  - But not specific products
- Our emphasis:
  - What’s out there?
  - How does it work?
  - What are its limits?
  - Can we find ways to hack around those limits?

Recent Trends

- The internet boom is maturing
  - We understand how to build big data centers and have a new architecture, Web Services, to let computers talk directly to computers using XML and other Web standards
  - There are more and more small devices, notably web-compatible cell phones
- Object orientation and components have emerged as prevailing structural option
  - CORBA, J2EE, .NET
- Widespread use of transactions for reliability and atomicity

Understanding Trends

- Basically two options
  - Study the fundamentals
  - Then apply to specific tools
- Or
  - Study specific tools
  - Extract fundamental insights from examples

Understanding Trends

- Basically two options
  - Study the fundamentals
  - Then apply to specific tools
- Or
  - Study specific tools
  - Extract fundamental insights from examples

Ken’s bias

- I work on reliable, secure distributed computing
  - Air traffic control systems, stock exchanges, electric power grid
  - Military “Information Grid” systems
  - Modern data centers
- To me, the question is:

  How can we build systems that do what we need them to do, reliably, accurately, and securely?

Butler Lampson’s Insight

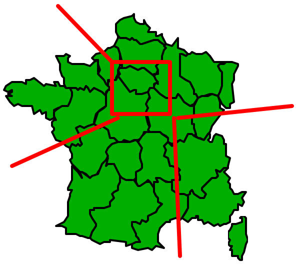
- Why computer scientists didn’t invent the web
  - CS researchers would have wanted it to “work”
  - The web doesn’t really work
  - But it doesn’t really need to!
- Gives some reason to suspect that Ken’s bias isn’t widely shared!

Example: Air Traffic Control using Web technologies

- Assume a “private” network
- Web browser could easily show planes, natural for controller interactions
- What “properties” would the system need?
  - Clearly need to know that trajectory and flight data is current and consistent
  - We expect it to give sensible advice on routing options (e.g. not propose dangerous routes)
  - Continuous availability is vital: zero downtime
    - Expect a soft form of real-time responsiveness
  - Security and privacy also required (post 9/11!)

ATC systems divide country up

France



More details on ATC

- Each sector has a control center
- Centers may have few or many (50) controllers
  - In USA, controller works alone
  - In France, a “controller” is a team of 3-5 people
- Data comes from a radar system that broadcasts updates every 10 seconds
- Database keeps other flight data
- Controllers each “own” smaller sub-sectors

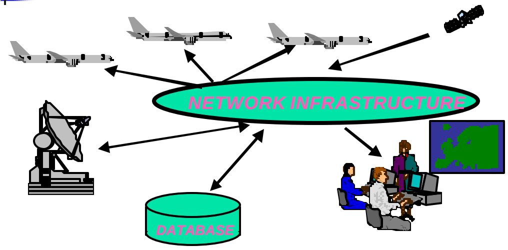
Issues with old systems

- Overloaded computers that often crash
  - Attempt to build a replacement system failed, expensively, back in 1994
- Getting slow as volume of air traffic rises
- Inconsistent displays a problem: phantom planes, missing planes, stale information
- Some major outages recently (and some near-miss stories associated with them)
  - TCAS saved the day: collision avoidance system of last resort… and it works….

Concept of IBM's 1994 system

- Replace video terminals with workstations
- Build a highly available real-time system guaranteeing no more than 3 seconds downtime per year
- Offer much better user interface to ATC controllers, with intelligent course recommendations and warnings about future course changes that will be needed

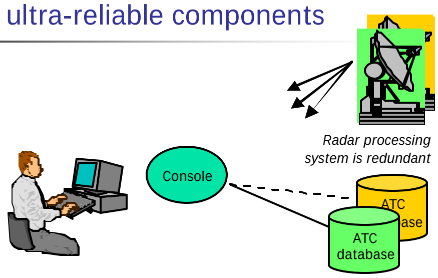
ATC Architecture



So… how to build it?

- In fact IBM project was just one of two at the time; the French had one too
  - IBM approach was based on lock-step replication
    - Replace every major component of the system with a fault-tolerant component set
    - Replicate entire programs ("state machine" approach)
  - French approach used replication selectively
    - As needed, replicate specific data items.
    - Program "hosts" a data replica but isn't itself replicated

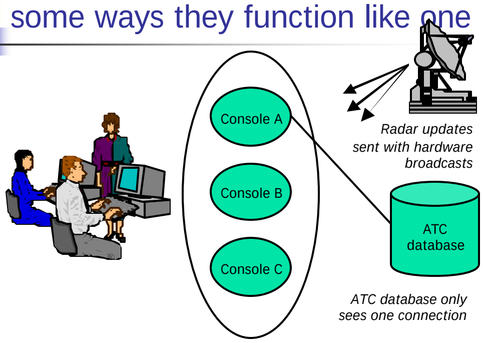
IBM: Independent consoles… backed by ultra-reliable components



Radar processing system is redundant

ATC database is really a high-availability cluster

France: Multiple consoles… but in some ways they function like one

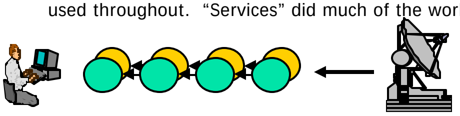


Radar updates sent with hardware broadcasts

ATC database only sees one connection

Different emphasis

- IBM imagined pipelines of processing with replication used throughout. "Services" did much of the work.



- French imagined selectively replicated data, for example "list of planes currently in sector A.17"
  - E.g. controller interface programs could maintain replicas of certain data structures or variables with system-wide value
  - Programs did computing on their own helped by databases

Other technologies used

- Both used standard off-the-shelf workstations (easier to maintain, upgrade, manage)
  - IBM proposed their own software for fault-tolerance and consistent system implementation
  - French used Isis software developed at Cornell
- Both developed fancy graphical user interface much like the Web, pop-up menus for control decisions, etc.

IBM Project Was a Fiasco!!

- IBM was unable to implement their fault-tolerant software architecture! Problem was much harder than they expected.
  - Even a non-distributed interface turned out to be very hard, major delays, scaled back goals
  - And performance of the replication scheme turned out to be terrible for reasons they didn't anticipate
- The French project was a success and never even missed a deadline… In use today.

Where did IBM go wrong?

- Their software "worked" correctly
  - The replication mechanism wasn't flawed, although it was much slower than expected
- But somehow it didn't fit into a comfortable development methodology
  - Developers need to find a good match between their goals and the tools they use
  - IBM never reached this point
- The French approach matched a more standard way of developing applications

ATC problem lingers in USA…

- "Free flight" is the next step
  - Planes use GPS receivers to track own location accurately
  - Combine radar and a shared database to see each other
  - Each pilot makes own routing decisions
  - ATC controllers only act in emergencies
- Already in limited use for long-distance flights

Free Flight (cont)

- Now each plane is like an ATC workstation
- Each pilot must make decisions consistent with those of other pilots
  - … but if FAA's project failed in 1994, why should free flight succeed in 2010?
  - Something is wrong with the distributed systems infrastructure if we can't build such things!
- In CS514, learn to look at technical choices and steer away from high-risk options

Impact of technology trends

- Web Services architecture should make it much easier to build distributed systems
  - Higher productivity because languages like Java and C# and environments like J2EE and .NET offer powerful help to developers
- The easy development route inspires many kinds of projects, some rather "sensitive"
  - But the "strong" requirements are an issue
    - Web Services aren't aimed at such concerns

Examples of mission-critical applications

- Banking, stock markets, stock brokerages
- Heath care, hospital automation
- Control of power plants, electric grid
- Telecommunications infrastructure
- Electronic commerce and electronic cash on the Web (very important emerging area)
- Corporate “information” base: a company’s memory of decisions, technologies, strategy
- Military command, control, intelligence systems

We depend on distributed systems!

- If these critical systems don’t work
  - When we need them
  - Correctly
  - Fast enough
  - Securely and privately
- … then revenue, health and safety, and national security may be at risk!

Critical Needs of Critical Applications

- Fault-tolerance: many flavors
  - Availability: System is continuously “up”
  - Recoverability: Can restart failed components
- Consistency:
  - Actions at different locations are consistent with each other.
  - Sometimes use term “single system image”
- Automated self-management
- Security, privacy, etc….:
  - Vital, but not our topic in this course

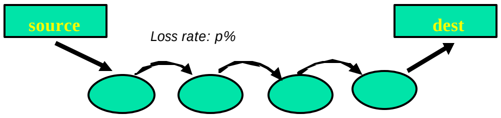
So what makes it hard?

- ATC example illustrated a core issue
- Existing platforms
  - Lack automated management features
  - Handle errors in ad-hoc, inconsistent ways
  - Offer one form of fault-tolerance mechanism (transactions), and it isn’t compatible with high availability
- Developers often forced to step outside of the box… and might stumble.
  - But why don’t platforms standardize such things?

End-to-End argument

- Commonly cited as a justification for *not* tackling reliability in “low levels” of a platform
- Originally posed in the Internet:
  - Suppose an IP packet will take n hops to its destination, and can be lost with probability p on each hop
  - Now, say that we want to transfer a file of k records that each fit in one IP (or UDP) packet
  - Should we use a retransmission protocol running “end-to-end” or n TCP protocols in a chain?

End-to-End argument



Probability of successful transit: $(1-p)^n$,
Expected packets lost: $k-k*(1-p)^n$

Saltzer et. al. analysis

- If p is very small, then even with many hops most packets will get through
  - The overhead of using TCP protocols in the links will slow things down and won't often benefit us
  - And we'll need an end-to-end recovery mechanism "no matter what" since routers can fail, too.
- Conclusion: let the end-to-end mechanism worry about reliability

Generalized End-to-End view?

- Low-level mechanisms should focus on speed, not reliability
- The application should worry about "properties" it needs
- OK to violate the E2E philosophy if E2E mechanism would be much slower

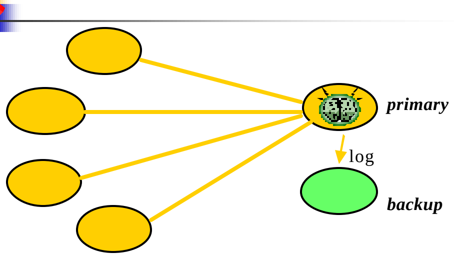
E2E is visible in J2EE and .NET

- If something fails, these technologies report timeouts
  - But they also report timeouts when nothing has failed
  - And when they report timeouts, they don't tell you what failed
  - And they don't offer much help to fix things up after the failure, either

Example: Server replication

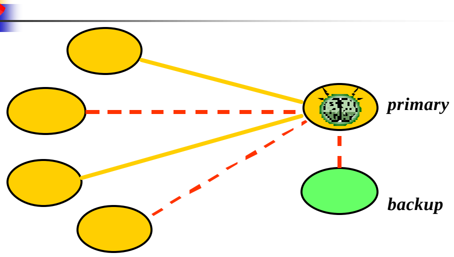
- Suppose that our ATC needs a highly available server.
- One option: "primary/backup"
  - We run two servers on separate platforms
  - The primary sends a log to the backup
  - If primary crashes, the backup soon catches up and can take over

Split brain Syndrome…



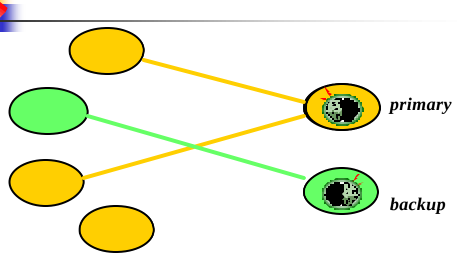
Clients initially connected to primary, which keeps backup up to date. Backup collects the log

Split brain Syndrome…



Transient problem causes some links to break but not all. Backup thinks it is now primary, primary thinks backup is down

Split brain Syndrome



Some clients still connected to primary, but one has switched to backup and one is completely disconnected from both

Implication?

- Air Traffic System with a split brain could malfunction disastrously!
  - For example, suppose the service is used to answer the question “is anyone flying in such-and-such a sector of the sky”
  - With the split-brain version, each half might say “nope”... in response to different queries!

Can we fix this problem?

- No, if we insist on an end-to-end solution
  - We’ll look at this issue later in the class
  - But the essential insight is that we need some form of “agreement” on which machines are up and which have crashed
  - Can’t implement “agreement” on a purely 1-to-1 (hence, end-to-end) basis.
    - Separate decisions can always lead to inconsistency
    - So we need a “membership service”... and this is fundamentally not an end-to-end concept!

Can we fix this problem?

- Yes, many options, once we accept this
  - Just use a single server and wait for it to restart
    - This common today, but too slow for ATC
  - Give backup a way to physically “kill” the primary, e.g. unplug it
    - If backup takes over... primary shuts down
  - Or require some form of “majority vote”
    - Ad mentioned, maintains agreement on system status
- Bottom line? You need to anticipate the issue... and to implement a solution.

CS514 project

- We’ll work with Web Services
  - .NET with ASP.NET in the language of your preference (C# is our favorite)
  - Or Java/J2EE
- We’ll extend the platform with features like replication for high availability, self-management, etc
- And we’ll use this in support of a mission critical application, mostly as a “demo”

You can work in small teams

- Either work alone, or form a team of 2 or 3 members
  - Teams should tackle a more ambitious problem and will also face some tough coordination challenges
  - Experience is like working in commercial settings...

Not much homework or exams

- In fact, probably *no* graded homework or graded exams
  - But we may assign thought problems to help people master key ideas
- Grades will be based on the project
  - Can be used as an MEng project if you like
  - In this case, also sign up for CS790 credits

Planned coverage of topics

- 1/27-2/3: Web Services: Concepts, Architecture, limitations
- 2/8: Trustworthy distributed computing systems
- 2/10-2/17: Global states and event ordering
  - **Logical clocks**
  - **Vector clocks.**
  - **Consistent cuts and global property detection.**
- 2/22-2/24: Transactions and multiphase commit
- 3/1-3/3: Rollback-recovery and message-logging protocols
- 3/8: Agreement protocols
- 3/10-5/7: Group-based programming abstractions
  - **Virtual synchrony, Primary-backup approach**
  - **Failure detection, roles for quorums**
  - **State machine approach**
- 4/12-4/21: Gossip and epidemic algorithms.
- 4/26-5/5: Time services, time-critical computing and protocols

Textbook and readings

- Ken's textbook is coming out from Springer Verlag in March
  - We're proving an online galley copy
  - It still has some typos; we'll point out the ones we know of… you should point out mistakes or problems if you see any
- Additional readings: Web page has references and links