CS514: Intermediate Course in Computer Systems

Lecture 2: August 1, 2002

Introduction to Object Oriented Systems and Architectures

Overview of Lecture

CS514

Introduction to the object layer

- Networks used to access objects (RMI, RPC).
- Message Oriented Middleware (MOM) Systems
- Web Services Architecture (WSDL, UDDI, XML)
- Rules for how to ask an object to do something
  - Marshalling: *Extensible Markup Language* – XML
  - Encoding requests: *Simple Object Access Protocol* – SOAP (in retrospect, should have been named *Services Oriented Architecture Protocol*).
- Behavior of interfaces: Idempotent and non-idempotent operations.
- Reliability models: best effort, at most once, at least once, exactly once, transactional.

Some terminology

CS514

- A program is the code you type in
- A process is what you get when you run it
- A message is used to communicate between processes. Arbitrary size.
- A packet is a fragment of a message that might travel on the wire. Variable size but limited, usually to 1400 bytes or less.
- A protocol is an algorithm by which processes cooperate to do something using message exchanges.

Some terminology

CS514

- An object could be any of these things – a program, a data file, a message, etc.
- Object-oriented languages and systems try to standardize some aspects of dealing with objects
- They also tend to be event oriented in the sense that the object
  - Registers methods to handle incoming events
  - Events could be windows (“frames” in Microsoft)-related, such as “redraw”, “mouse click”, etc
  - Or they could be incoming messages
  - They could even be exception notifications
- Many are multi-threaded and hence concurrent

More terminology

CS514

- A network is the infrastructure that links the computers, workstations, terminals, servers, etc.
  - It consists of routers
  - They are connected by communication links
- A network application is one that fetches needed data from servers over the network
- A distributed system is a more complex application designed to run on a network. Such a system has multiple processes that cooperate to do something.

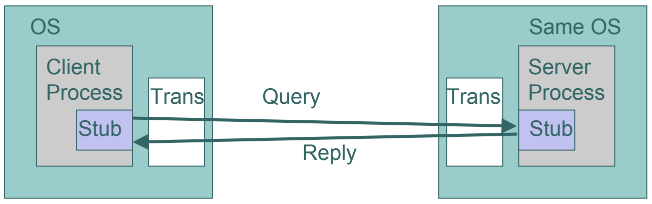
Remote Procedure Call

CS514

- Procedure call semantics, but to another process (on another machine)
- Simple query/reply, typically synchronous (blocking)
- Originally “hard-wired” into programs
- Evolved to include marshalling, naming service, process management

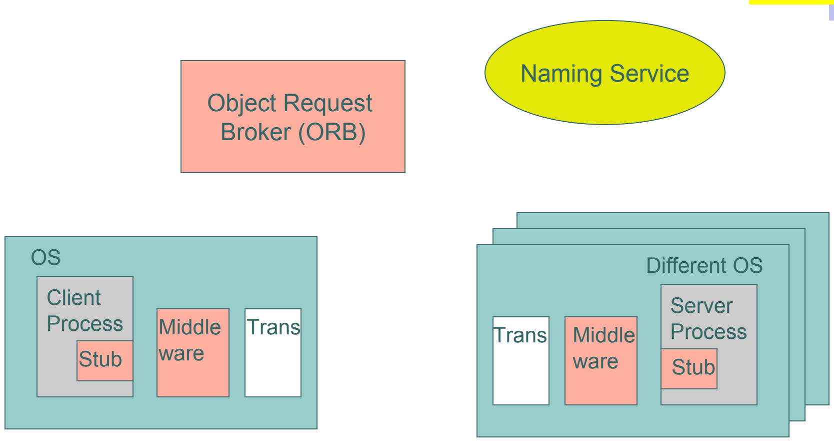
Simple RPC

CS514



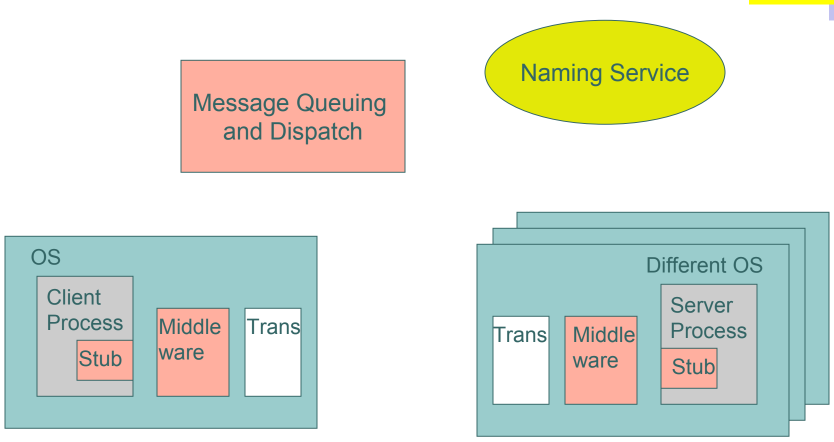
Full-featured RPC

CS514



Message Oriented Middleware (MOM)

CS514



RPC versus MOM

CS514

- RPC = sync, MOM = async
- MOM isolates client from server behavior
  - Server doesn't have to be up
  - Client doesn't need to know anything about server
- MOM may have better throughput
  - But at the expense of latency between query and reply

Web sites and services

CS514

- A web site is a document service. Using HTTP browser requests files, which can be generated on the fly, “pushes” data to database, checks to see if a cached object has changed.

Web sites and services

CS514

- "Web services are Internet-based modular applications that perform a specific business task and conform to a specific technical format. A Web service can be anything from a restaurant review service to a real-time travel advisory to an entire airline ticket reservation process. The modular technical format ensures these self-contained business services (from the same or different companies) will mix and match easily to create a complete business process. Businesses can dynamically publish, discover and aggregate a range of Web services via the Internet; in this way, they can more easily and dynamically create innovative products, business processes, and value chains. Web services can be delivered to any customer device (cell phone, PDA, computer, etc.) and can be created or transformed from existing applications." -- IBM
- Web services *generalize* the idea of a Web site.

Web sites and services

CS514

- "Web services are Internet-based modular applications that perform a specific business task and conform to a specific technical format.
-

Web sites and services

CS514

- "Web services are Internet-based modular applications that perform a specific business task and conform to a specific technical format. A Web service can be anything from a restaurant review service to a real-time travel advisory to an entire airline ticket reservation process.
-

Web sites and services

CS514

- "Web services are Internet-based modular applications that perform a specific business task and conform to a specific technical format. A Web service can be anything from a restaurant review service to a real-time travel advisory to an entire airline ticket reservation process. The modular technical format ensures these self-contained business services (from the same or different companies) will mix and match easily to create a complete business process.
-

Web sites and services

CS514

- "Web services are Internet-based modular applications that perform a specific business task and conform to a specific technical format. A Web service can be anything from a restaurant review service to a real-time travel advisory to an entire airline ticket reservation process. The modular technical format ensures these self-contained business services (from the same or different companies) will mix and match easily to create a complete business process.
  Businesses can dynamically publish, discover and aggregate a range of Web services via the Internet; in this way, they can more easily and dynamically create innovative products, business processes, and value chains. Web services can be delivered to any customer device (cell phone, PDA, computer, etc.) and can be created or transformed from existing applications." -- IBM
- Web services *generalize* the idea of a Web site.

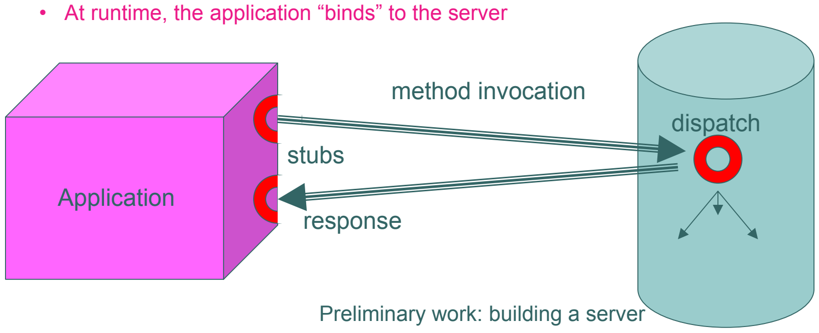
Objects “vs.” Web Services?

CS514

- Two sets of terminology but ultimately, the same ideas
- A Web Service is just an object accessible over a network
  - The rules for how that object should be accessed are fixed in the standards
  - But pretty much any object can be a Web Service
- This is the key insight behind Microsoft .NET and Sun’s J2EE: each architecture tries to make it as easy to build a Web Service as it is to define a new object

Preliminary work: building a client application

- Application imports the server’s interface definition
- Compiler links the application with a “stub”
- At runtime, the application “binds” to the server



Preliminary work: building a server

- The server *exports* its interfaces
- Compiler creates a receiver-side “dispatch” routine to interpret incoming requests
- At runtime, the server registers itself in a namespace
- Then it loops waiting for incoming requests

Why object orientation?

CS514

- Until mid 1980's, we were very program-oriented
  - Operating system was an environment for managing files and running programs
  - Programs had ways to create network sockets – communication endpoints
    - IP address plus a port number
    - Few standards on how these were assigned
    - Application defined own communication rules
  - File systems and databases offered special libraries so that remote clients could gain access

1980's: Distributed Computing Environments

CS514

- In mid 1980's pressure grew to provide better support for client-server computing
  - Server could be a file system, database or some other application, such as "network information service", system password service, etc.
  - Client could be anything
- DCEs proposed to standardize model

DCEs quickly “failed”

- Harder to use DCEs than people expected!
- Main issue was that
  - Programs were written in any way you liked
  - But DCE wanted standards!
- Further issues were lack of standards for thread packages, use of weakly typed languages, confusion about the best model for supporting communication in operating systems.

But object orientation didn’t

- Push emerged to define object-oriented programming models
- The UNIX community went with CORBA: Common Object Request Broker Architecture.
- The Microsoft world went with DCOM: Distributed Common Object Model, and Active-X

New languages triggered breakthrough!

CS514

- When Sun promoted Java (recently renamed by Microsoft as C#), a barrier fell
- These languages were object oriented yet looked like C++
- Better match between language and object model fueled huge advances in terms of runtime environment

The picture today?

CS514

- Two dominant models
  - J2EE: Java Enterprise Edition, Version 2.0. (Jini flamed out…)
  - .NET
- CORBA hasn't yet faded away
- Web Services will unify everything under a single, extremely "verbose" standard – slow but universal.

How did we get here?

CS514

- DCE was too early, too complex
- CORBA is the right idea, though quite heavyweight
  - (OSI baggage like ASN.1)
- COM/DCOM is Microsoft
- Java was just right for web app
  - J2EE grew out of this wave
- .NET is Microsoft
- Web Services --- still lots of hype, we'll see…

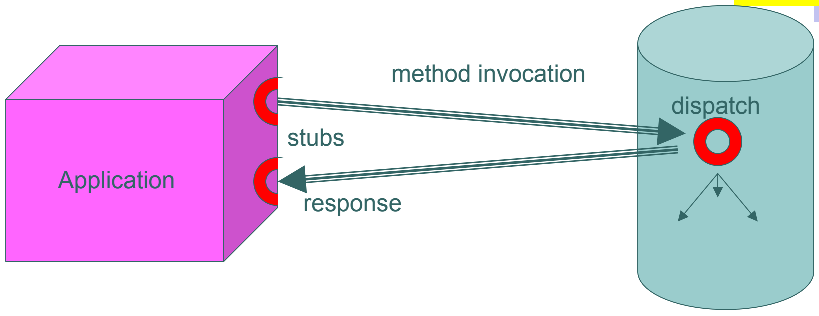
Key goals of object oriented systems, of all "flavors"

CS514

- Location transparency
  - Programmer shouldn't need to code in a very different way when invoking an object whether it is
    - In local address space
    - In a shared library or DLL
    - On a remote machine
- Object reuse
- Fault containment

What are Web Services?

CS514



Goal is just to support client-server model using Web standards

What are Web Services?

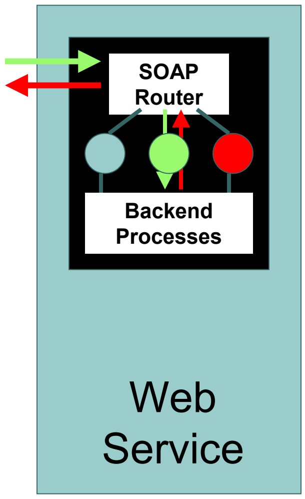
CS514

- “Web Services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP.”

What are Web Services?

CS514

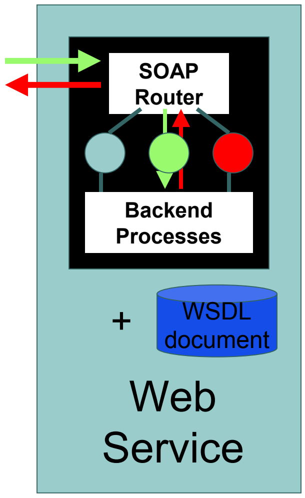
- "Web Services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."
- Today, SOAP is the primary standard, but one could imagine other future standards. SOAP provides rules for encoding the request the client wishes to issue and its arguments, so that the service can perform the desired operations.



What are Web Services?

CS514

- "Web Services are software components described via WSDL which are capable of being accessed via standard network protocols such as SOAP over HTTP."
- WSDL documents can be used to drive object assembly, code generation, and other tools to assist in application development. For now, SOAP plus WSDL are the core of the Web Services architecture. Other elements such as UDDI are "add ons" for specific uses.



Role of UDDI?

CS514

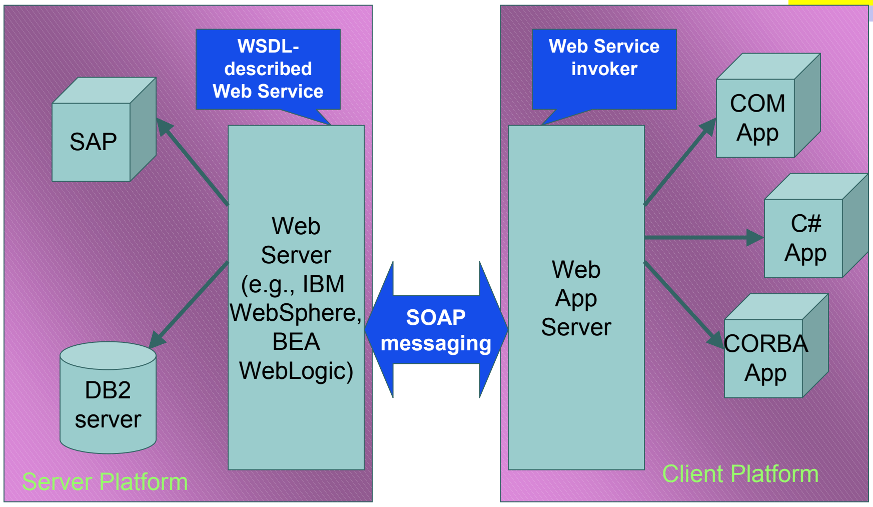
- UDDI is a form of name space
- The Web Services system can publish various attributes using UDDI
- Clients employ UDDI to find the service and obtain its WSDL documentation
- There are also other such standards – security, for example, and “inspection”

Web Services are often Front Ends

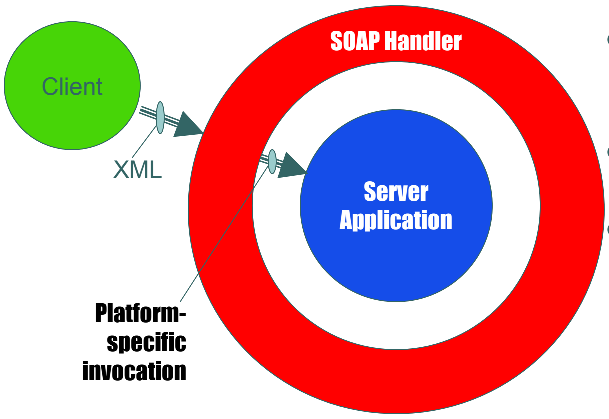
CS514

- Usually, the Web Service system sits in front of a set of corporate database systems, file systems, and other servers
- On the client side, the term “application server” is often used
- Approach offers a great deal of platform independence

Web Services are often Front Ends



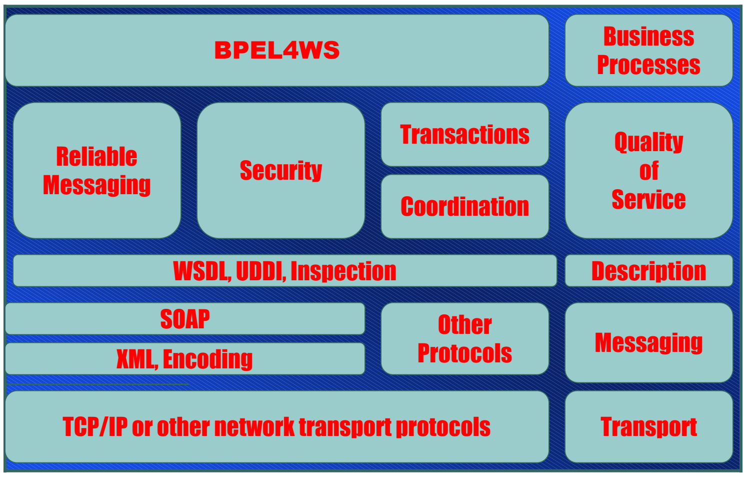
Web Services as a "wrapper"



- The traditional application never went away
- But now we can talk to it through SOAP
- Overhead is often considerable. So this will demand a round of upgrading to new machines!

The Web Services "stack"

CS514



Issues?

CS514

- ☹ Building "good" Web Servers
  - Not really our topic in CS514
  - Quite a challenge! Performance issues are very difficult, and tools are still primitive
- Tolerating failures in various parts of the architecture
- Issues of naming, mobility, network quality of service
- Scalability challenges

Where are the issues?

CS514

- In the past, two “components” of a system could interact with any protocol we chose to implement
- Now, with Web Services, the intent is that all components will interact using the WS_XXXX standards
- Is this an issue?

The problem with transports

CS514

- The Web Services stack does allow for new transport protocols, but
  - What are the reliability properties of the existing architecture?
  - To what extent are problematic properties tied to the transport layer?
  - Could such issues be tackled purely by changing the transport, or might they demand “other” changes?

Example: High availability

CS514

- Suppose that a Web Service runs on multiple machines
  - WS_0 on machine alpha.mit.edu
  - WS_1 on machine beta.mit.edu
- A client system wants to treat this Web Service as a “single” abstract service
- What should be the rules for fail-over?

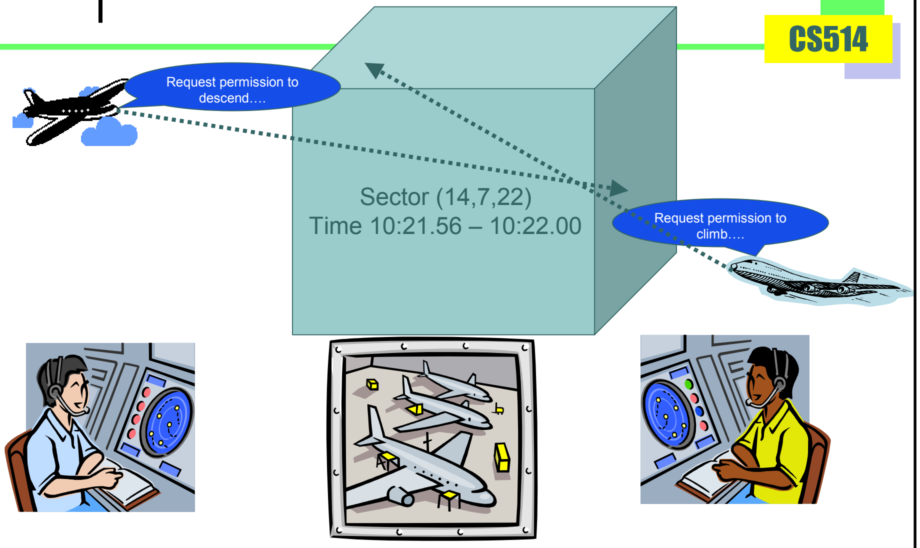
Recall the split brain problem

CS514

- Suppose our Web Service is a front end to a database of air traffic control data.
  - Requests could “reserve sector x,y,z from time t_a to t_b”
  - If the server grants such a request, the sector should be “safe” for the controller to direct a plane into it
  - Can also “cancel reservation res_id”

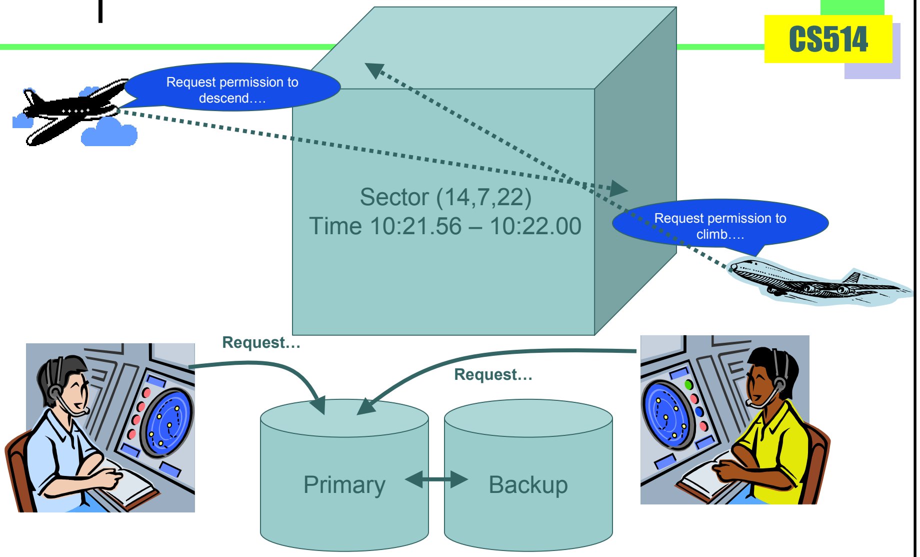
Troublesome scenario

CS514


Request permission to descend….
Request permission to climb….
Sector (14,7,22)
Time 10:21.56 – 10:22.00

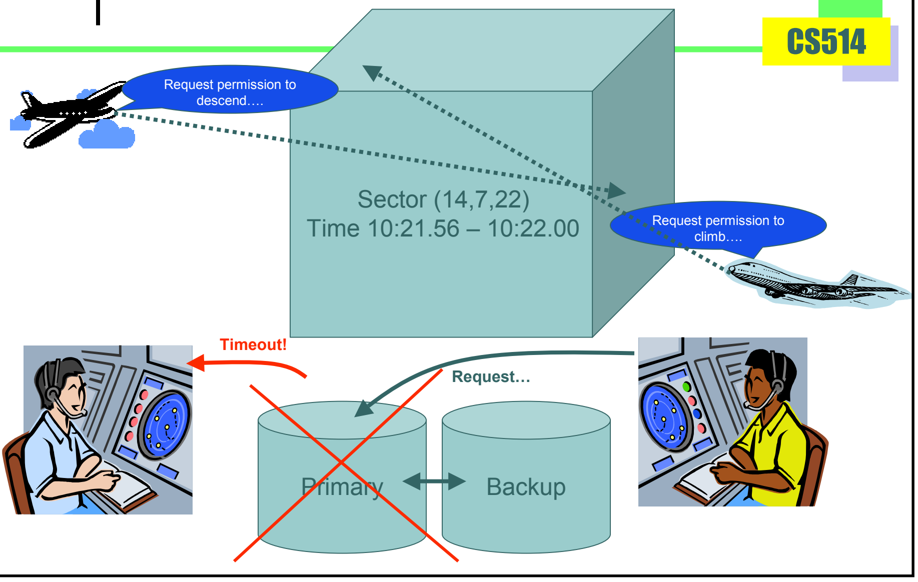
Troublesome scenario

CS514



Troublesome scenario

CS514



What does this controller know?

CS514

- The request was submitted
  - But we don't know if the server saw it
  - We don't know if it tried to respond
  - And we don't actually know if the server is even up or down!
- High availability is a *guarantee* that no matter what, the system can remain responsive with small delays
  - So we need a way to move forward, fast!

Some questions

CS514

- Suppose that the client application issues a reservation request and gets a timeout
- Can it safely disconnect, connect to the backup, and issue the same request?
- This question is actually *underspecified.* Without pinning down more details, we just can't answer it.

Things we would need to know

CS514

- How is the database itself replicated?
  - Is there a single database server, or did each WS front-end have its own server?
  - If the latter, how do the servers handle replication?
  - In particular, when WS_0 fails, is there a "cleanup delay" before WS_1 knows of all the actions that WS_0 might have taken?
- Is WS_0 even down? How do Web Services detect failures? If by timeout, when the client sees a timeout, does everyone else see one too?

A partial answer

CS514

- Web Services use timeout for failure detection
  - This happens in the "transport" layer
  - For example, TCP has a KEEPALIVE parameter
  - RMI (aka RPC) has built-in timeouts too
- The application can get some control over timeout parameters
- But the application can't "review" suspected failures… hence mistakes can happen

Classes of operations

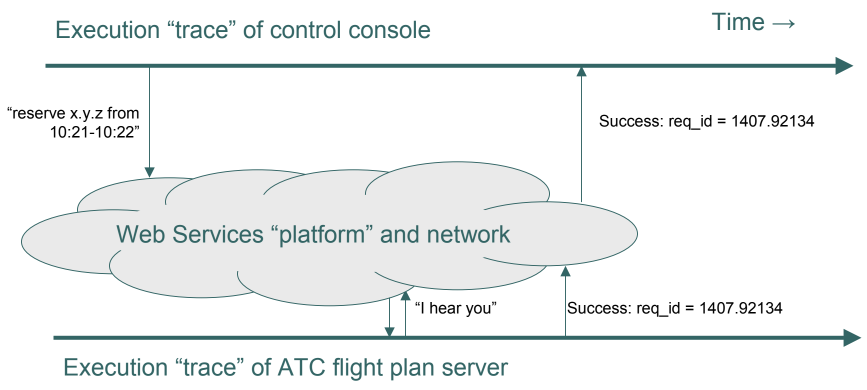
CS514

- Idempotent: Client can issue the operation as many times as it likes. Each time, effect is identical.
- At least once: The system promises to somehow ensure that the request will get done *one or more times.* But the client will need some sort of acknowledgement that the system "has" the request firmly in hand
- Exactly once: Similar to at least once, but the guarantee is exact.
- At most once: The request won't occur twice…

Knowledge of outcomes

CS514

- Consider a request done in stages:


Execution "trace" of control console
Time →
"reserve x.y.z from 10:21-10:22"
Success: req_id = 1407.92134
Web Services "platform" and network
"I hear you"
Success: req_id = 1407.92134
Execution "trace" of ATC flight plan server

What do we "know" at each stage?

CS514

- First the client system has the request but the knowledge is contained in the application. Clearly the WS system can't promise anything yet
  - Why? Well, the client could crash
  - So… the reservation hasn't happened yet but is "pending"

What do we "know" at each stage?

- Next, the client has invoked the WS platform *on its own machine*
- Now what do we know?
  - Is the WS platform a "networked application" or a "distributed system"?
  - In fact it is something of a black box!
  - The "I hear you" ack is part of the way that TCP works. Some RPC protocols also have an ack. But *not an element of the WS specification* – the application can't see it

What do we "know" at each stage?

- Eventually, the WS platform has sent the request (encoded in XML, using HTTP to get it there, over TCP) to the remote WS platform on the server
  - And eventually, it can tell that the request made it to the destination machine
  - But the server could still crash
  - And the database could crash
  - And nobody "logs" these requests
- So, do we "know" anything new?

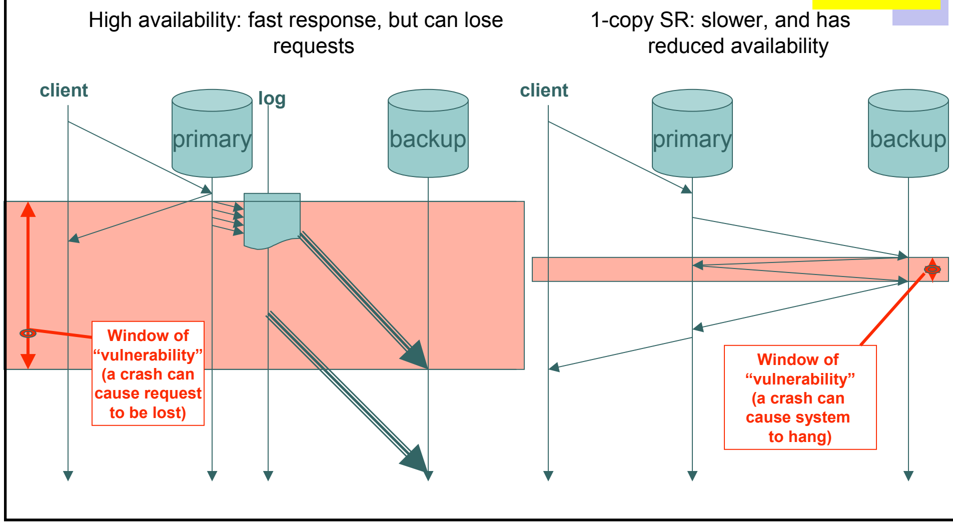
What do we "know" at each stage?

CS514

- Now the WS platform decodes the request and the database subsystem gets the operation and performs it
- How does it get replicated?
  - *High availability* database products update immediately, then spool an update record into a log to be delivered later at the backup
  - Thus a delay can occur and a crash could prevent the update from getting through
  - So-called "2-phase commit" can avoid this, but can compromise availability.

Database Replication Dilemma

CS514


High availability: fast response, but can lose requests
client
primary
log
backup
Window of "vulnerability" (a crash can cause request to be lost)
1-copy SR: slower, and has reduced availability
client
primary
backup
Window of "vulnerability" (a crash can cause system to hang)

Why is high availability so hard?

CS514

- This relates to the database durability guarantee
- We'll look at it in more detail later, but
  - To replicate data, database may need to run a 2-phase commit
  - This protocol can hang if a crash happens at the wrong step –there are no non-blocking commit protocols
  - Thus a database must pick between higher performance/availability and stronger guarantees. They pick the former.

Reasons a backup could lag

CS514

- We might want to log many records before sending a message.
  - This is because the cost of sending a message is largely independent of the size of the message!
  - Many messages will cost many times as much as a single bigger one!
- Also, the client is actually waiting and may "evaluate" the DB by response time

Other reasons?

CS514

- Backup itself may process updates in batches
- Backup may need time to switch to being in "primary" mode
  - After a crash a backup may need to clean up the database
  - In fact, after a crash, a backup might even need to wait for the primary to recover!
  - And in some situations, human intervention is ultimately required

Other options?

CS514

- Primary and backup could share a physical disk on which the database resides
  - Then backup runs directly from the real database
  - But if that disk crashes, there is no second copy
  - This is also much more expensive and hence a less common option.

Back to our ATC problem

CS514

So, what do we "know"?

- Until we get a "reserved" message back, we know nothing at all
- Once we do get that message, we may still not be sure that, if the primary were to crash, the backup would "know" about the reservation
- And even if it eventually will find out, there may be a time delay before it does, and we have no idea how long this will take
- Even worse (as we'll see later in the course), there can be cases where a backup would get "stuck" when using the lock-step style of replication!

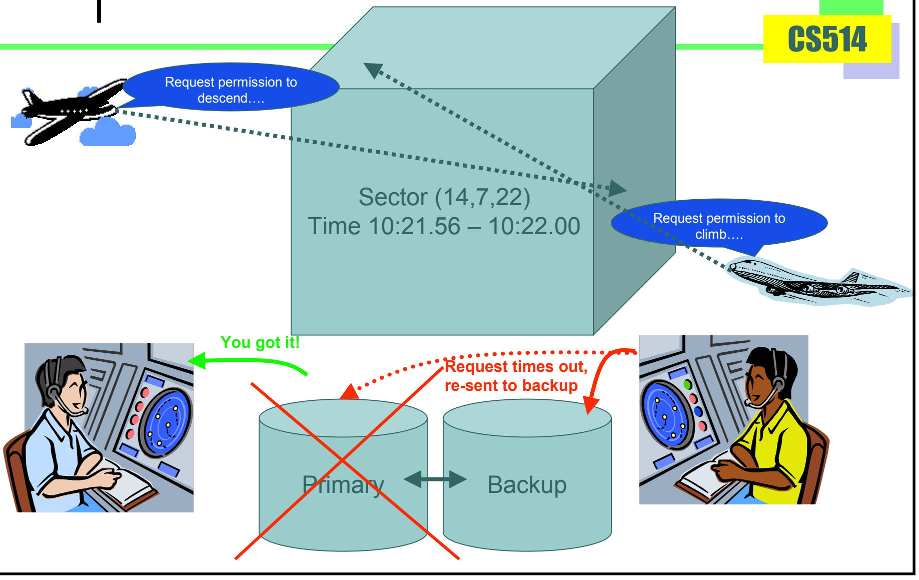
A reservation is idempotent

CS514

- We could keep trying until successful
  - This is a case of "at least once" semantics
- But would that be enough?
  - Yes, if we are only worried about a single controller
  - But no if multiple controllers depend on the system to resolve conflicts!

Troublesome scenario

CS514


Request permission to descend….
Sector (14,7,22)
Time 10:21.56 – 10:22.00
Request permission to climb….
You got it!
Request times out, re-sent to backup
Primary
Backup

Scenario?

CS514

- Controller “a” asks WS_0 to reserve x,y,z at time t and gets an ack.
- WS_0 is seen as having crashed by controller “b”. So that computer fails over and asks WS_1 to do a conflicting request (involving this same reserved sector of the sky)
- Without more application-specific mechanisms, we just can’t be confident that WS_1 will tell “b” that this request is not safe!

Highlights a flaw in the WS architecture

CS514

- There has been little attention to robustness at all levels
  - Internet itself (this dates back to its origins)
  - Database (this reflects commercial considerations – "judged by speed")
  - WS protocol (it basically mimics Web browser transactions)
- Issue is particularly evident if we want high availability, although also seen in other (slightly more complex) scenarios.

Can a system "do better"?

CS514

- Databases first encountered this same issue in non-distributed settings
- Issue was as follows:
  - Application accesses a shared database
  - A crash occurs
  - Did its operations get "saved"?
  - Also, what if several are doing this at once. Could there be interference?

Transactions

CS514

- Database model was proposed by Jim Gray and others to address this issue
- Application must
  - *begin* the transaction
  - Issue operations: *read* or *update*
  - End with *commit* or *abort*
- Database offers various guarantees

Transaction

CS514

- Atomicity: Although the client issues multiple operations, the database treats them as a single operation. Thus nobody can ever see a partially executed database state
- Consistency: Although the database permits a high level of concurrency, it looks to the client as if one transaction occurred at a time, in some sequential order. “Serializability”
- Isolation: Transactions running concurrently can’t interfere with one-another. (Follows from above?)
- Durability: If the database successfully commits a transaction, it won’t forget the transaction even if a crash and recovery occur.

WS_Transaction

- Web Services are offering this same transactional interface to users
  - So a client can talk to a database through a Web Service interface
- But Web Services lack a fail-over standard
  - So, we can't solve our problem using transactions
  - Also, if the client's "commit" request times out, WS_transaction lacks any guaranteed way to figure out if the service committed or aborted its request.

Summary

- Object architectures have become standard
- Web Services have become the standard interoperability mechanism for such architectures (despite high overheads)
  - But the mechanism has many potential consistency problems
  - WS_transaction can only address some of them – those relating to interaction with a single non-replicated database or a database replicated, but not able to guarantee high availability