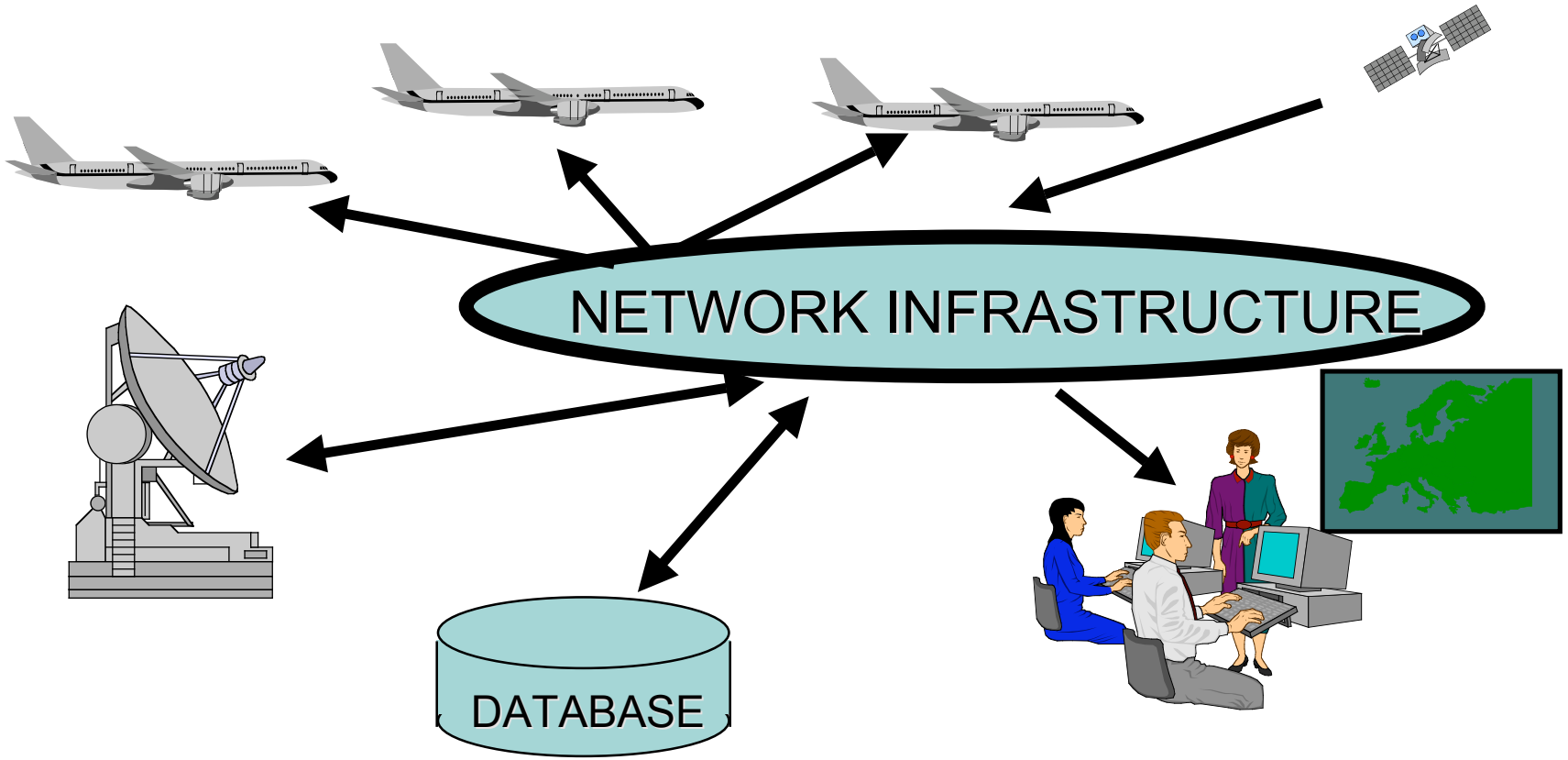
CS603: Distributed Systems

Lecture 2: Client-Server Architecture, RPC, Corba

PURDUE UNIVERSITY

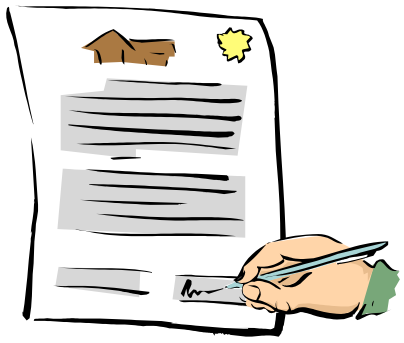
ATC Architecture



HOW WOULD YOU START BUILDING SUCH A SYSTEM?

Outline

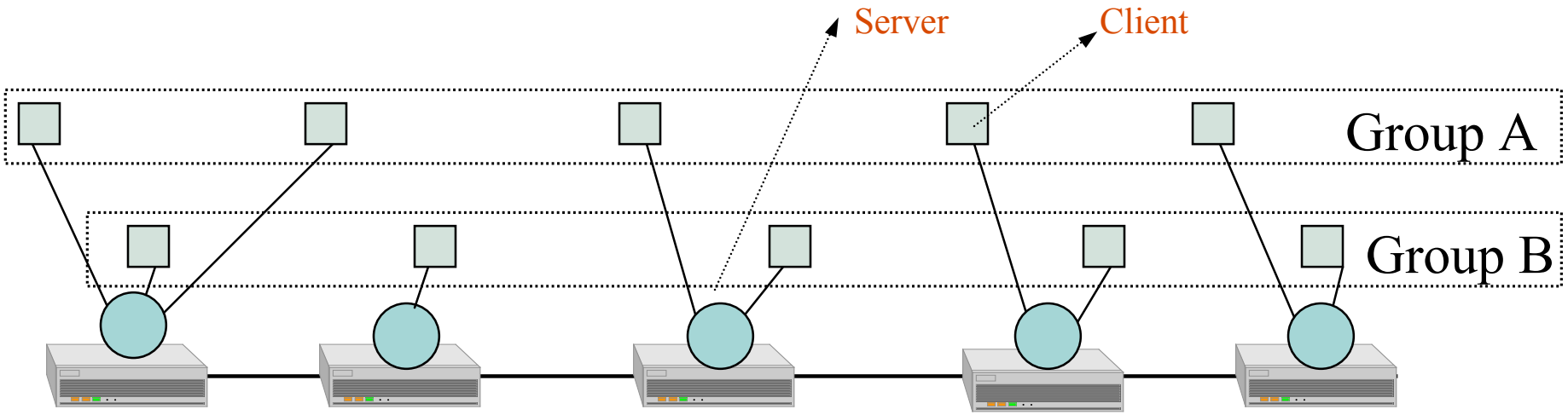
- Technologies/Protocols used in Designing Distributed Systems
  - Client/Server
  - RPC
  - Corba
  - J2EE
  - .NET
  - Web Services



Client/Server Architecture

- Functionality is partitioned in a set of services provided by a set of servers
- Clients (applications) interact with each through the servers
- Examples:
  - File servers
  - Database servers
  - Network name servers
  - Network time servers
  - Mail servers
  - Web servers

Example: Group Communication Systems

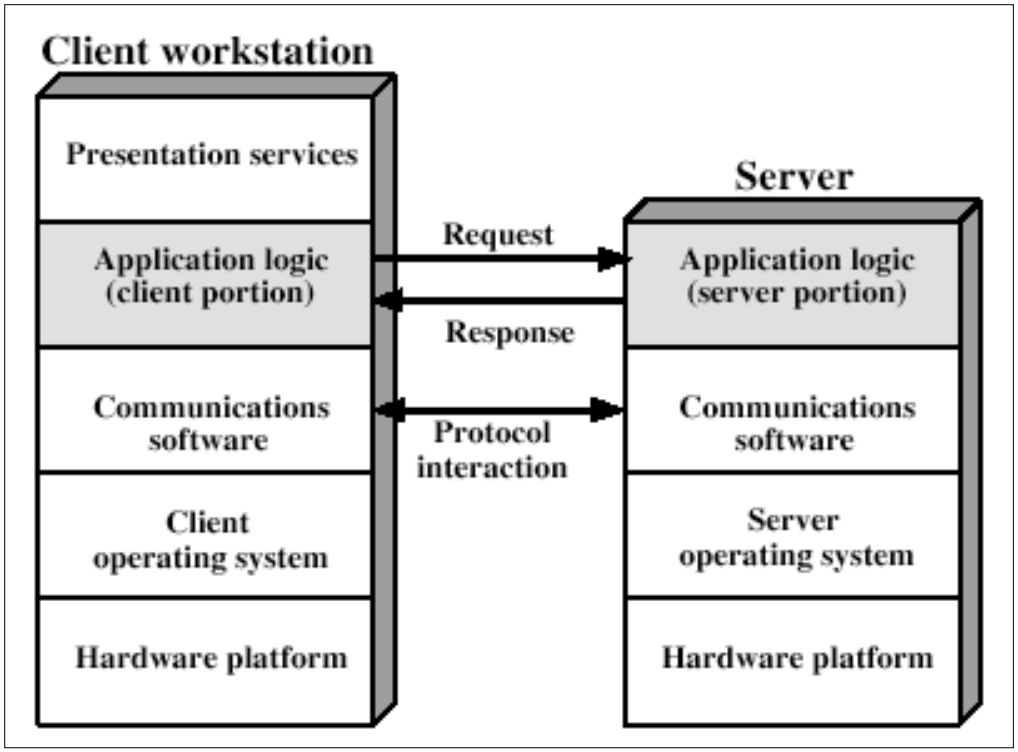


- Reliable and ordered message delivery
- Group membership service (list with connected members)

Clients do not connect with each other, they communicate using the GCS servers

Client/Server General Architecture

- Client must ‘bind’ to a server
- Standard services run on well-known ports
- Clients discover services (directory of servers providing a desired service)

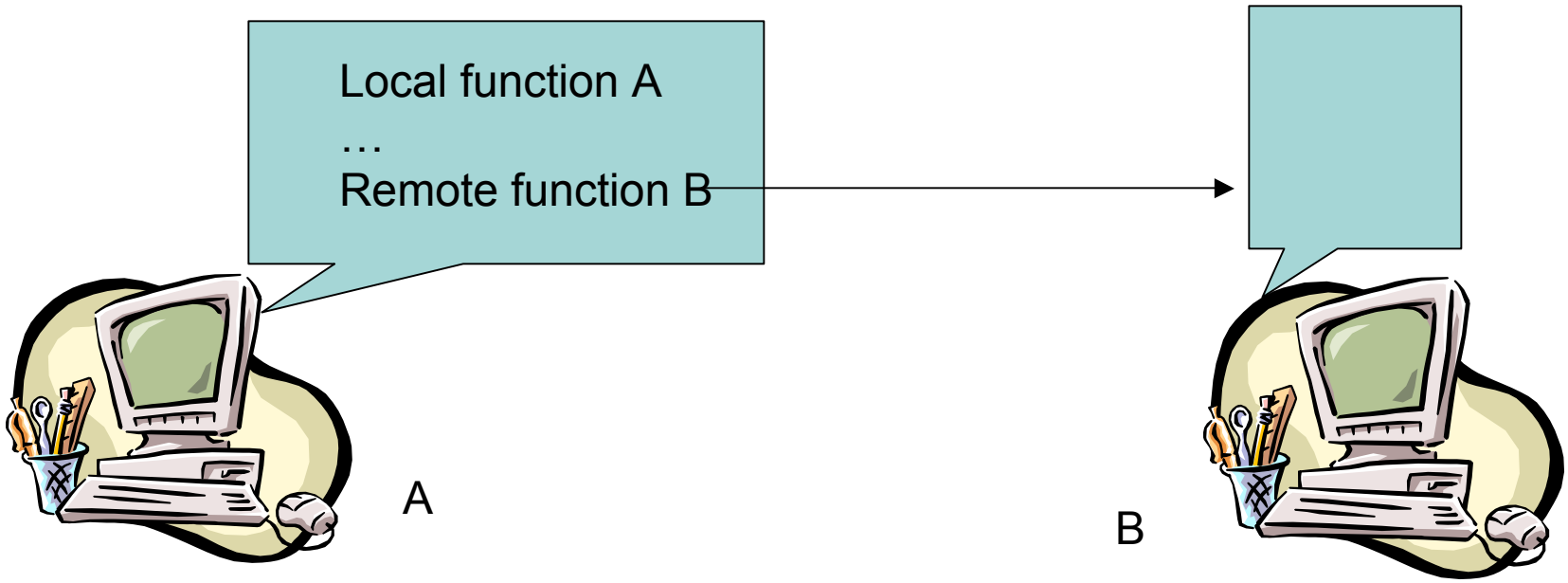


Styles of Client/Server

- Stateless: server does not keep any information between requests. There may be a shared state in the form of cache, but the correct function does not require the shared state to be accurate.
- Stateful: server remembers information between requests. Client may take local actions based on accuracy of information.

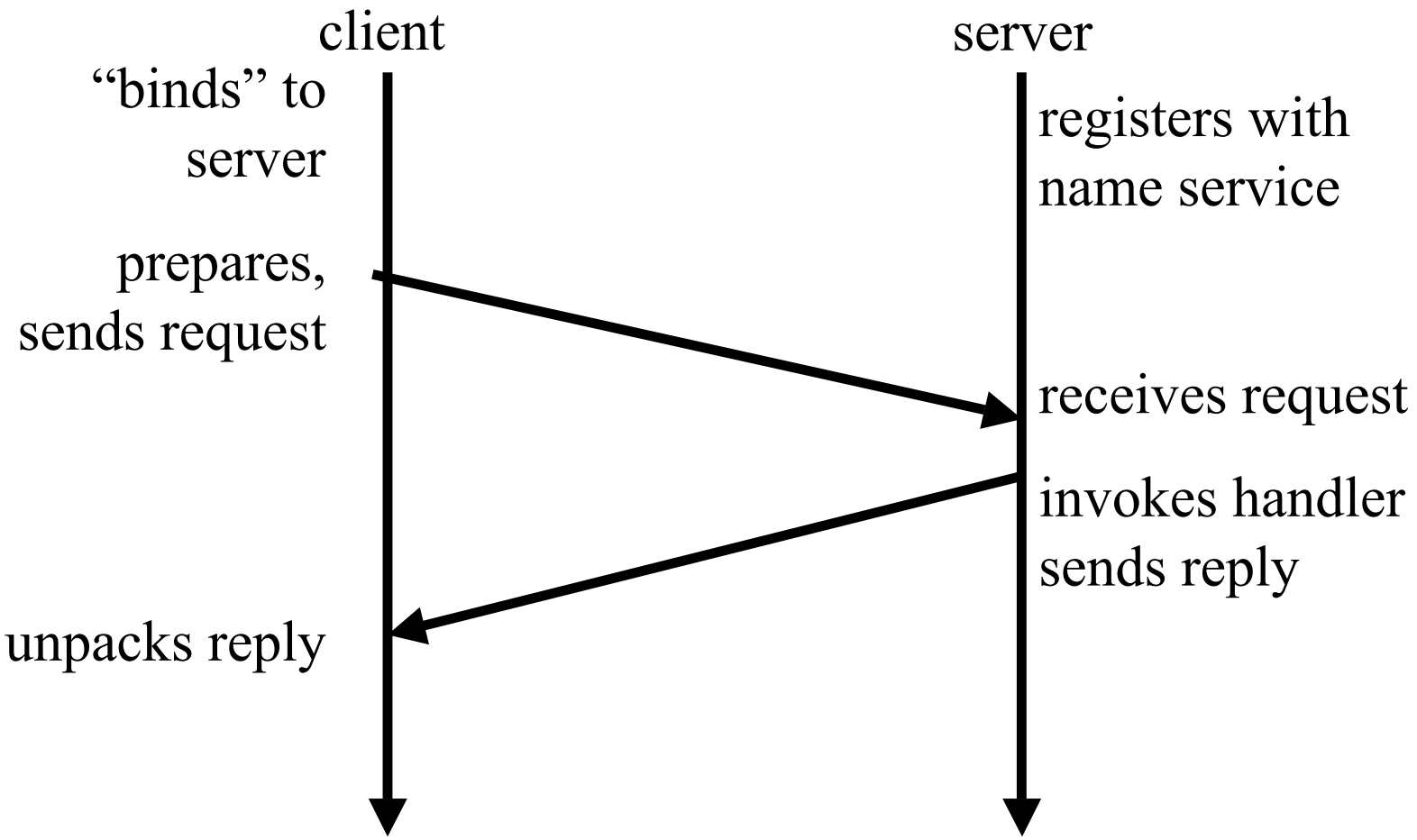
- Can you think about examples in each case?

Remote Procedure Call (RPC)



Provides support for programs to call a procedure on a remote machine "just" as you would on the local machine.(Birrell and Nelson 1985)

The basic RPC protocol



RPC

- Provides a portable, high-level programming interface, hides details as byte-ordering, alignment
- The remote procedure interface defines all communication.
- Servers can be found with the help of portmapper:
  - Server publish port, **name** and **version** with the portmapper daemon
  - Client contacts the portmapper and asks where it can find the remote procedure, using **name** and **version** ids. The portmapper daemon returns the address and client and server communicate directly.

RPC: What can go wrong?

- Network failure, client failure, server failure
- Runs on UPD, reliability is achieved using acknowledgments
  - If timeout with no ack, resend packet.
  - **May result in replayed requests.**
- Detect duplicates: a sequence number and timestamp embedded to enable detection of duplicates.

RPC Optimization

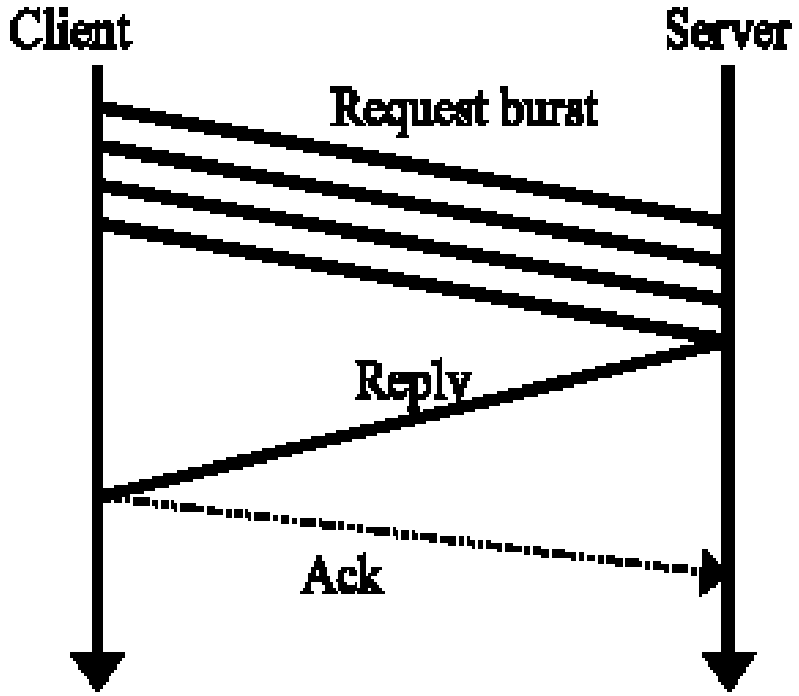


Figure 4.4. RPC using a burst protocol; here the reply is sent soon enough so that an acknowledgement to the burst is not needed.

- Delay sending acks, so that imminent reply itself acts as an ack.
- Don't send acks after each packet.
- Send ack only at the end of transmission of entire RPC request.
- NACK sent when missing sequence number detected

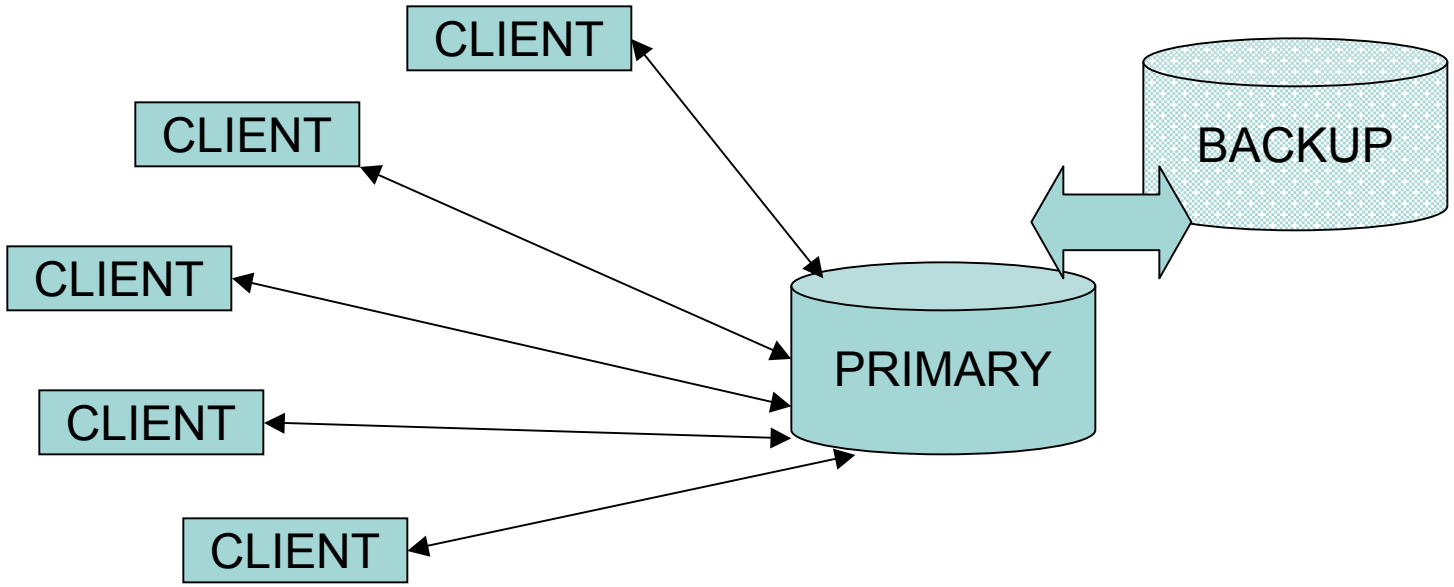
What does a failed request mean?

- Client is waiting for acknowledgment that does not come
- What does this mean: Network failure (also long delay) and/or machine failure!
- How long should the client wait?

IF THE MACHINE FAILED, DID THE SERVER PROCESS THE REQUEST?????

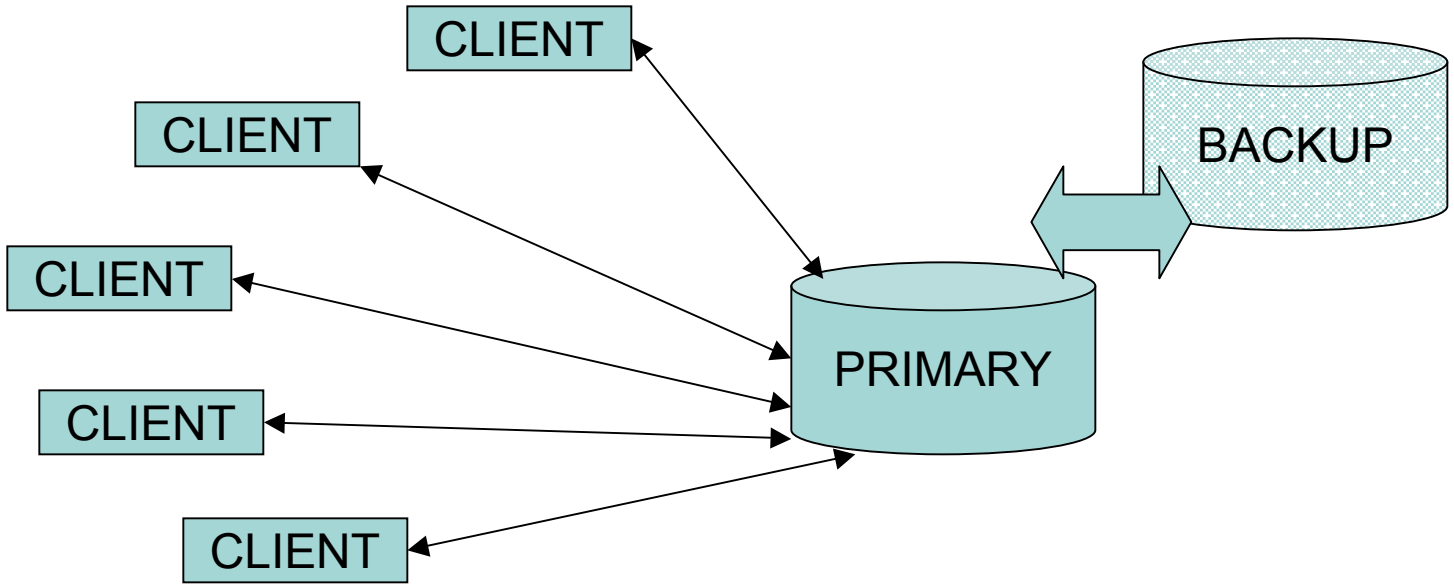
Example: Server Replication

- Provide a highly available service using two servers: a primary and a backup
- The primary logs everything to the backup
- If primary crashes, the backup soon catches up and can take over

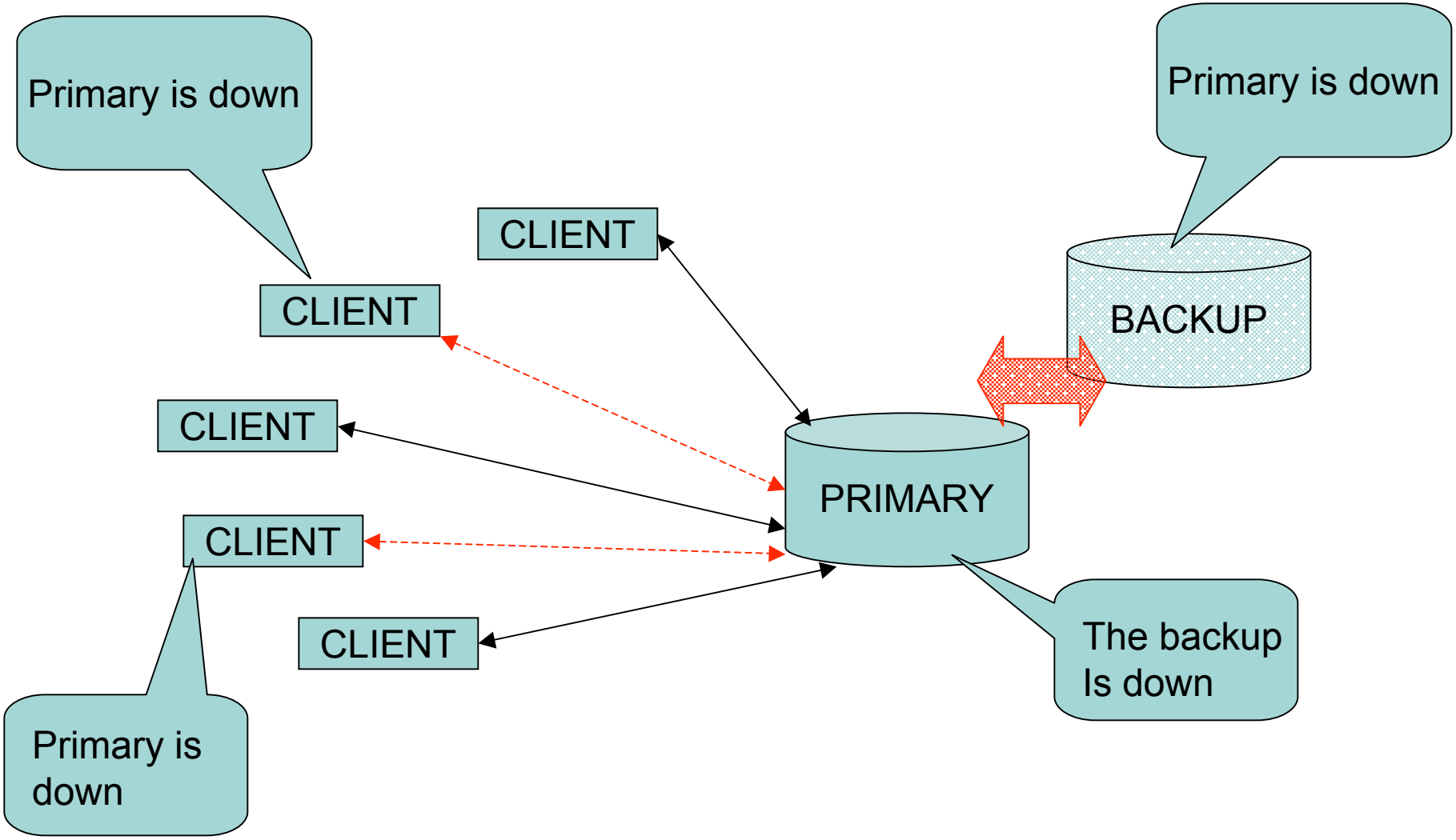


Normal case

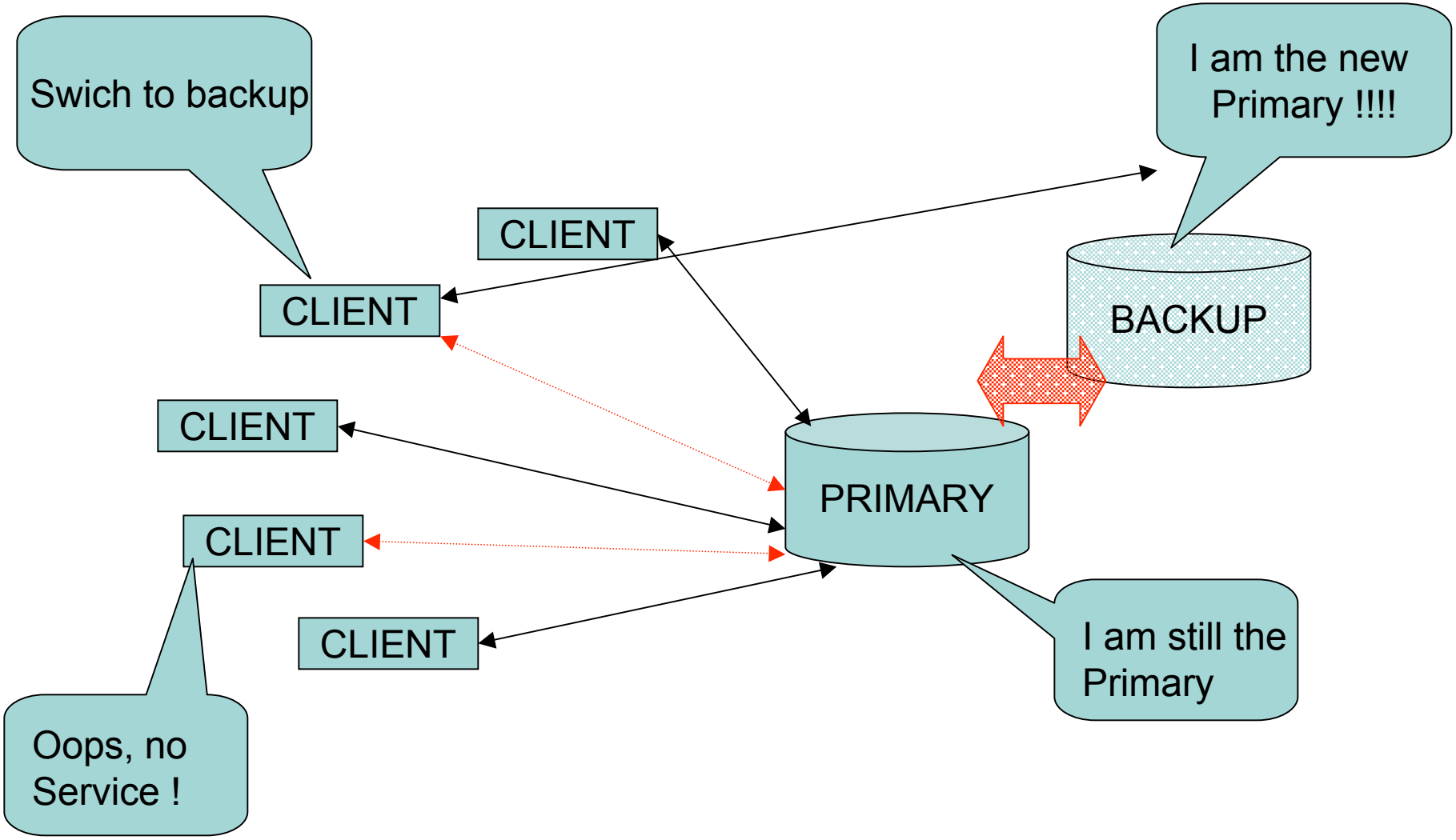
- Everybody connected to the backup, no problems



Things go wrong…



Things go **very** wrong…



SO WHAT?

Remember the ATC System: What if the system is used to answer the question “is anyone flying in such-and-such a sector of the sky”?????

How to fix it???

Intuitively all participants should agree on who is alive and who is not, should agree on who is the primary

We will see later more details about this

RPC on TCP

- TCP gives reliable communication when both ends and the network connecting them are up.
- RPC protocol itself does not need to employ timeouts and retransmission: less complex implementation.
- Broken connections reported by TCP mean the same thing they did earlier (without TCP): Client still doesn't know whether the server processed its request.

RPC Semantics

- "Exactly Once"
  - Each request handled exactly once.
  - Impossible to satisfy, in the face of failures.
  - ➢ Can't tell whether timeout was because of node failure or communication failure
- "At most Once"
  - Each request handled at most once.
  - Can be satisfied, assuming synchronized clocks, and using timestamps.
- "At least Once"
  - If client is active indefinitely, the request is eventually processed (maybe more than once)

- RPC: remote procedure call
- Lots of applications are object-oriented

How to provide support for distributed object oriented applications?

What are Web Services?

- Software components that allows applications (**different programming languages and different platforms**) to exchange data over computer networks
- Communication is via **SOAP** (uses HTTP)
- Web services can be described in a standard way **WSDL** (uses XML) language

Benefits

- Software components that allows applications (**different programming languages and different platforms**) to exchange data over computer networks

Portability, vendor, platform independence

Benefits

- Communication is via **SOAP** (uses HTTP)

Use of HTTP ensures that web services can work through many common firewall security measures without requiring changes to their filtering rules.

Web Services: Details

- Service must be published
- Client must
  - Discover the service
  - Bind to the server
  - Pack the request (marshaling) and send the SOAP request
- Server must
  - Unpack the request (demarshaling), handles it, computes result.
  - Sends answer back in the reverse direction: from the server to the SOAP router back to the client.
- Communication goes through the SOAP router

SOAP

- a cleansing agent made from the salts of vegetable or animal fats
  OOOPs! Wrong definition :)

- **Simple Object Access Protocol SOAP** : lightweight XML-based protocol for exchange of information in a decentralized, distributed environment:
  - an envelope that defines a framework for describing what is in a message and how to process it
  - a set of encoding rules for expressing instances of application-defined datatypes
  - a convention for representing remote procedure calls and responses

Where Web Services Fail Short

- Allow the data center to control decisions the client makes
- Provide assistance in implementing naming and discovery in scalable cluster-style services
  - How to load balance? How to replicate data? What precisely happens if a node crashes or one is launched while the service is up?
  - Help with dynamics. For example, best server for a given client can be a function of load but also affinity, recent tasks, etc

How Web Services Deal with Failures

- Failures of
  - naming service
  - backend servers
  - clients
- As other technologies, Web services suffers from:
  - can not distinguish between crash failures and transient failures (crash vs. latency)
  - when the service reports an error client does not know details about what happened

Web Service Applications

- **Grid Computing: Distributed computing**
- Involves coordinating and sharing computing, application, data, storage, or network resources across dynamic and geographically dispersed organizations.
- Research challenges: scalability, security, system management
- Computing power and storage increase, network remains the bottleneck

Future? Autonomic Computing

- Targets large-scale computer systems
- Computers must learn to manage themselves, in accordance with high-level guidance from humans.
- Self-monitoring, self-configuration, self-optimization, self-healing, and/or self-protection.
- Specific self-managing components, such as server, client, database, storage, or network elements. Emphasis should be placed on interactions with other components, or techniques or lessons that may generalize to other components.
- http://www.autonomic-conference.org/

Next …

- We will look at fundamental concepts in distributed systems:
  - Time
  - Consistency
  - Detecting failures
  - Membership