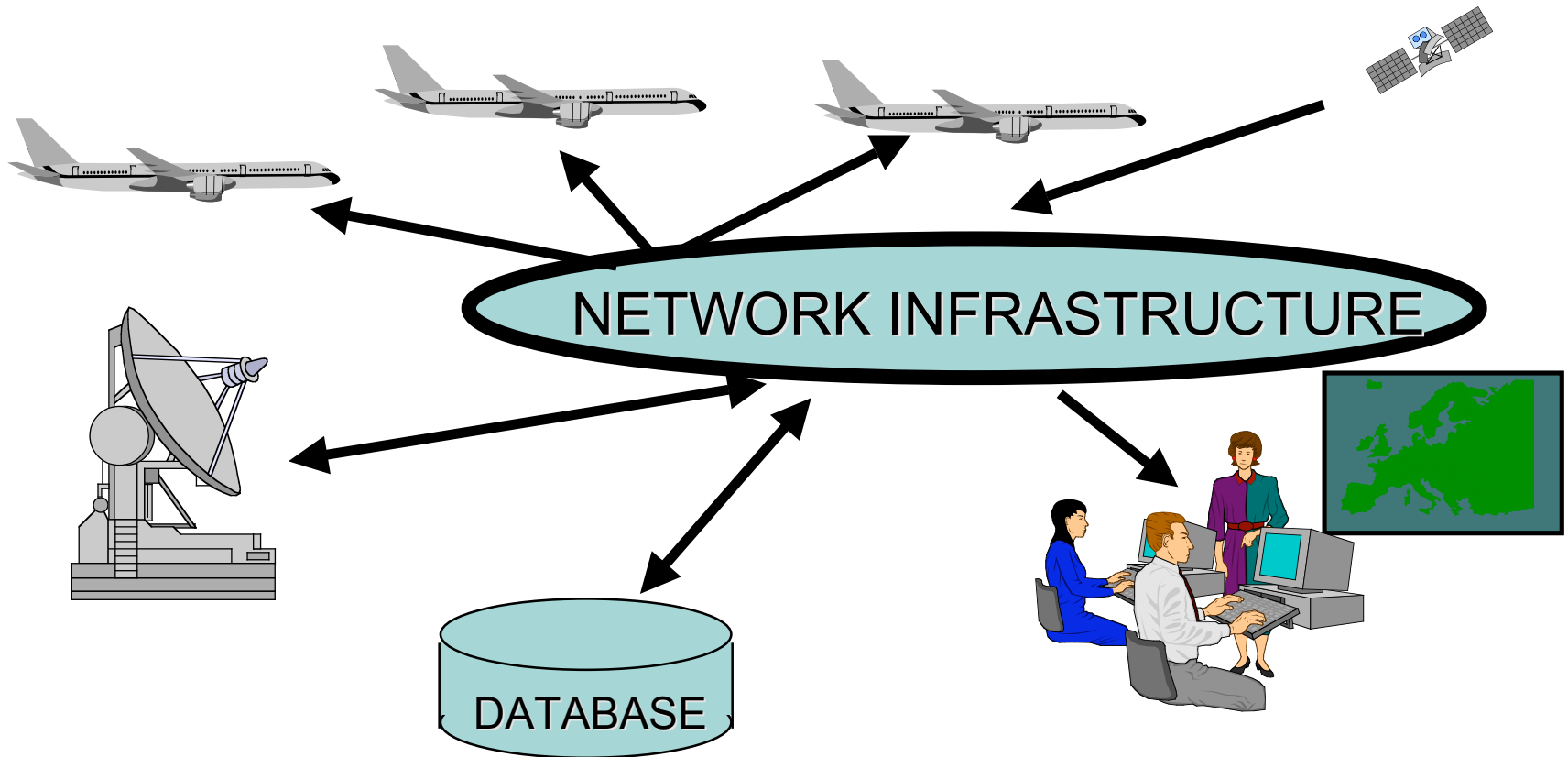


CS603: Distributed Systems

Lecture 2: Client-Server Architecture, RPC, Corba

ATC Architecture



HOW WOULD YOU START BUILDING SUCH A SYSTEM?

Outline

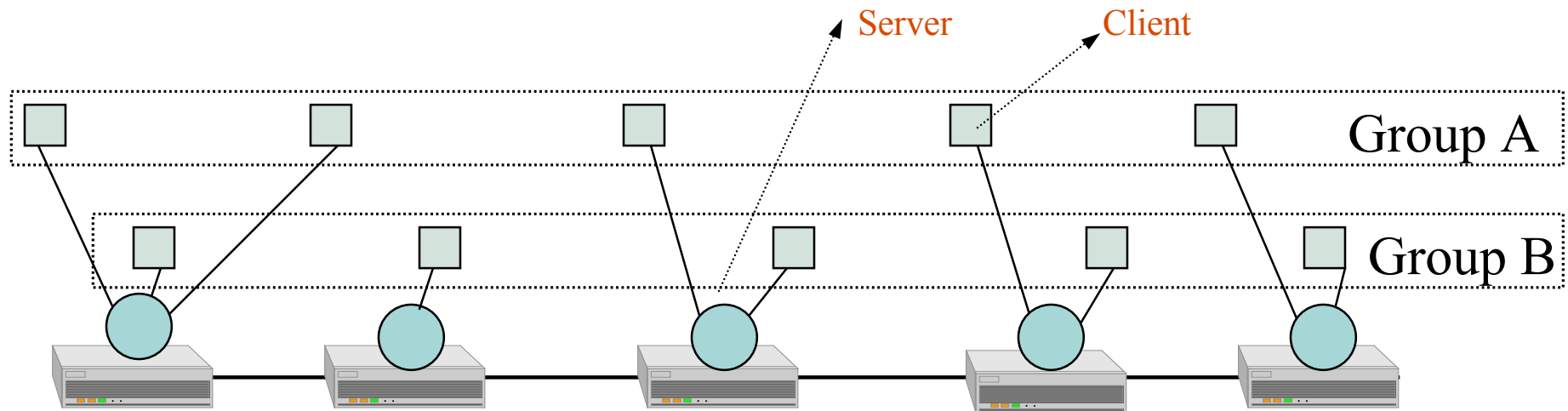
- Technologies/Protocols used in Designing Distributed Systems
 - Client/Server
 - RPC
 - Corba
 - J2EE
 - .NET
 - Web Services



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

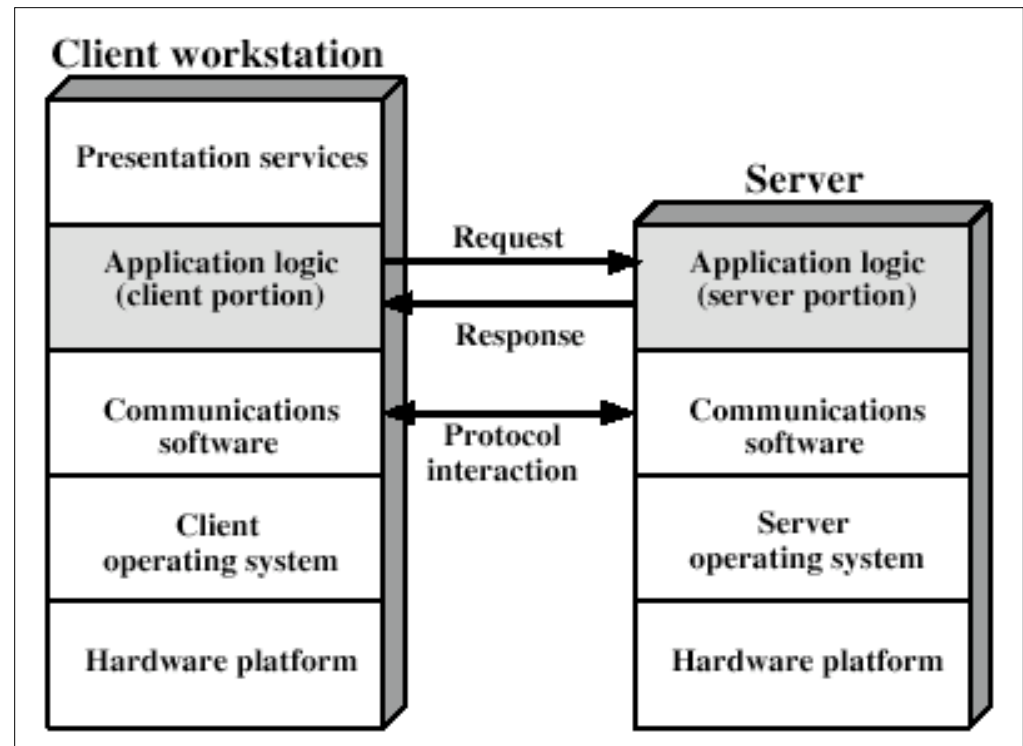


- Reliable and ordered message delivery
- Group membership service (list with connected members)

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

Client/Server General Architecture

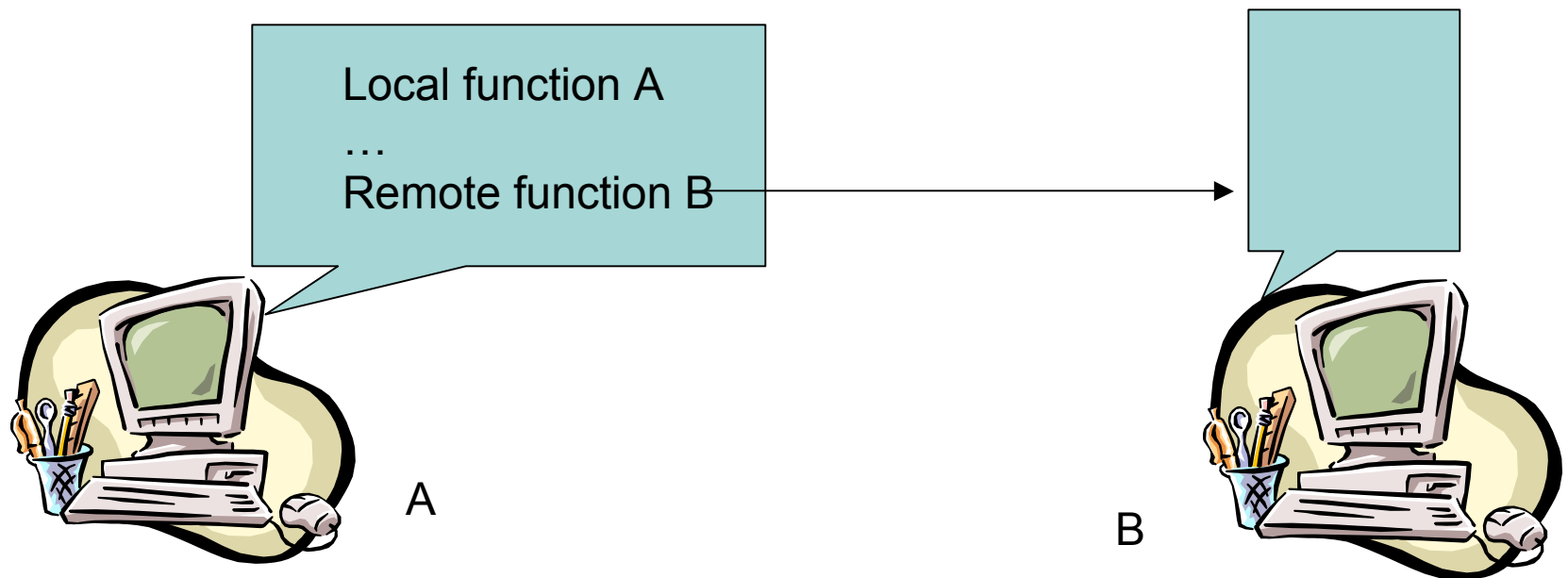
- Client must 'bind' to a server
- Standard services run on well-known ports
- Clients discover services (directory of servers providing a desired service)



Styles of Client/Server

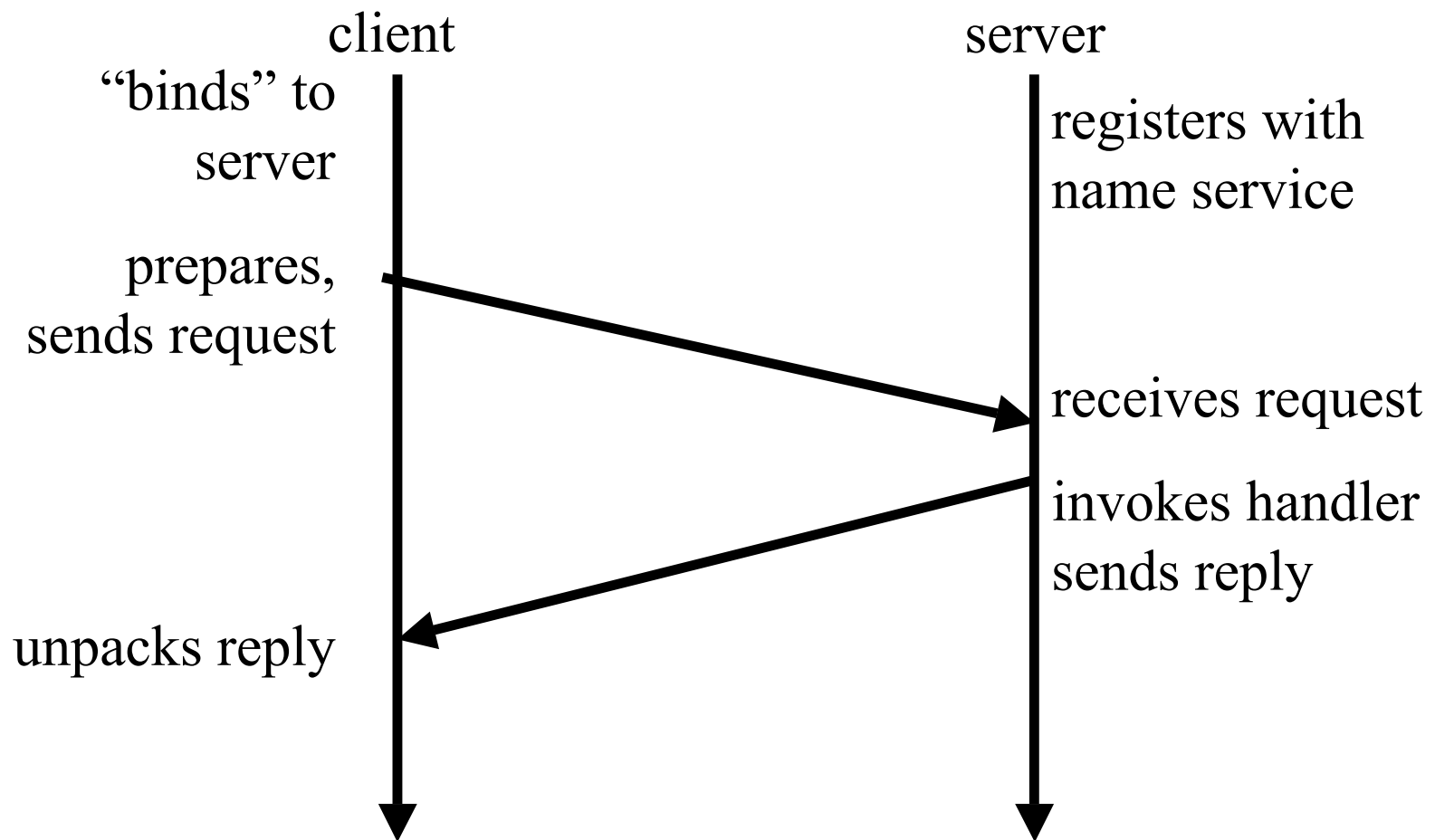
- **Stateless**: server does not keep any information between requests. There may be a shared state in the form of cache, but the correct function does not require the shared state to be accurate.
- **Stateful**: server remembers information between requests. Client may take local actions based on accuracy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

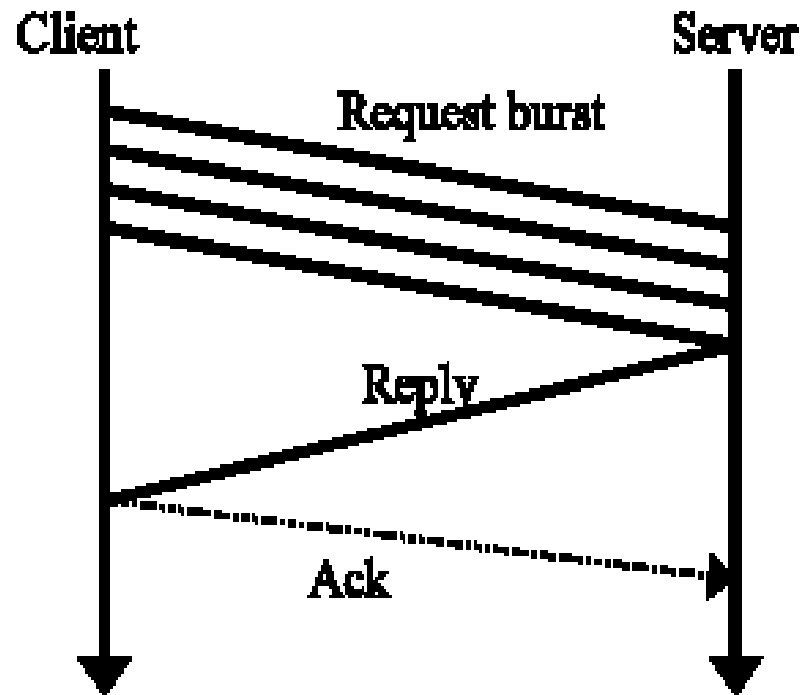


Figure 4.4. RPC using a burst protocol; here the reply is sent soon enough so that an acknowledgement to the burst is not needed.

- 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

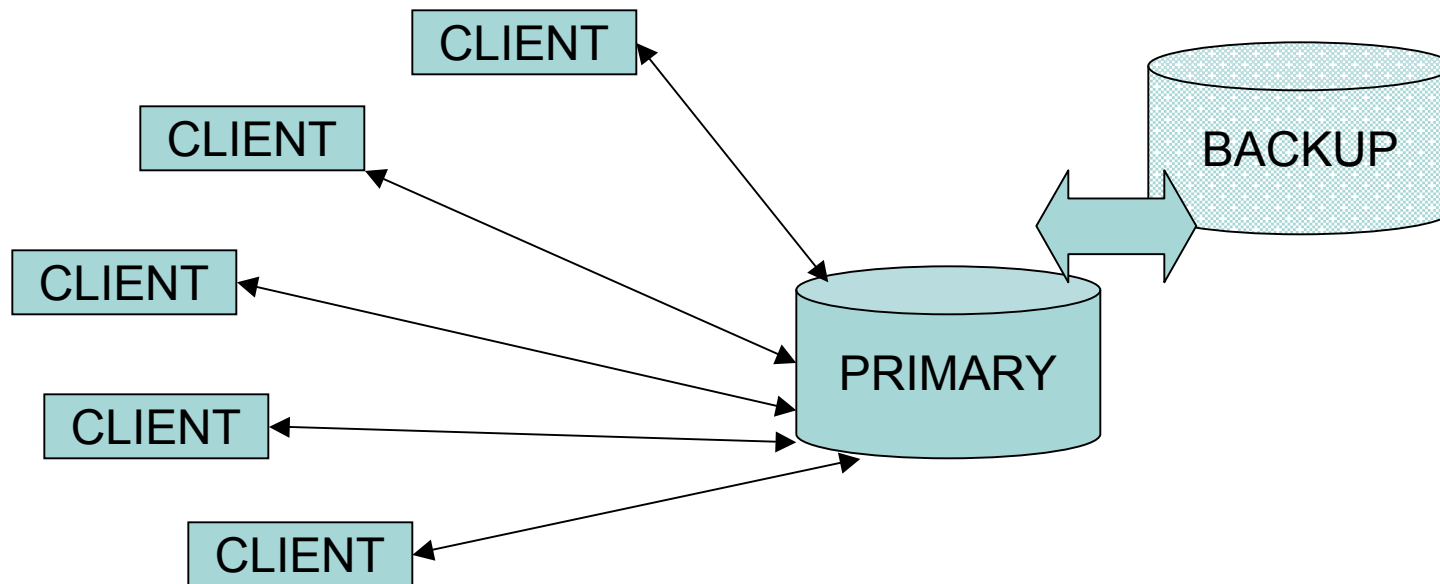
What does a failed request mean?

- 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?????

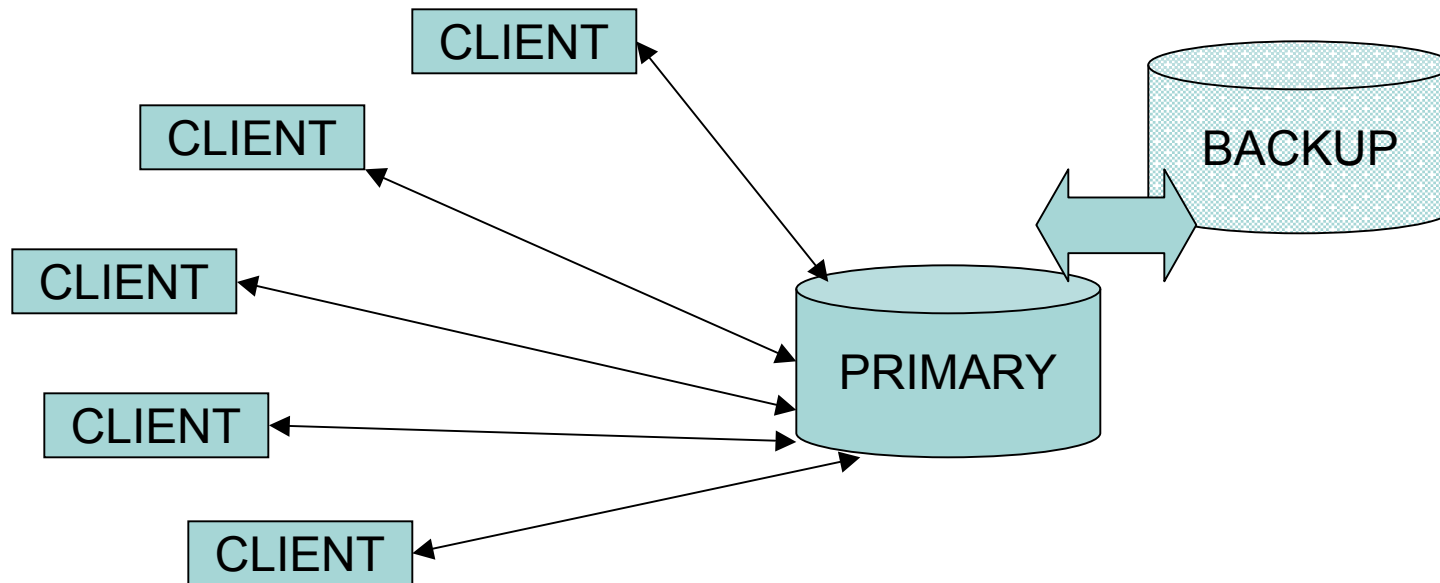
Example: Server Replication

- 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

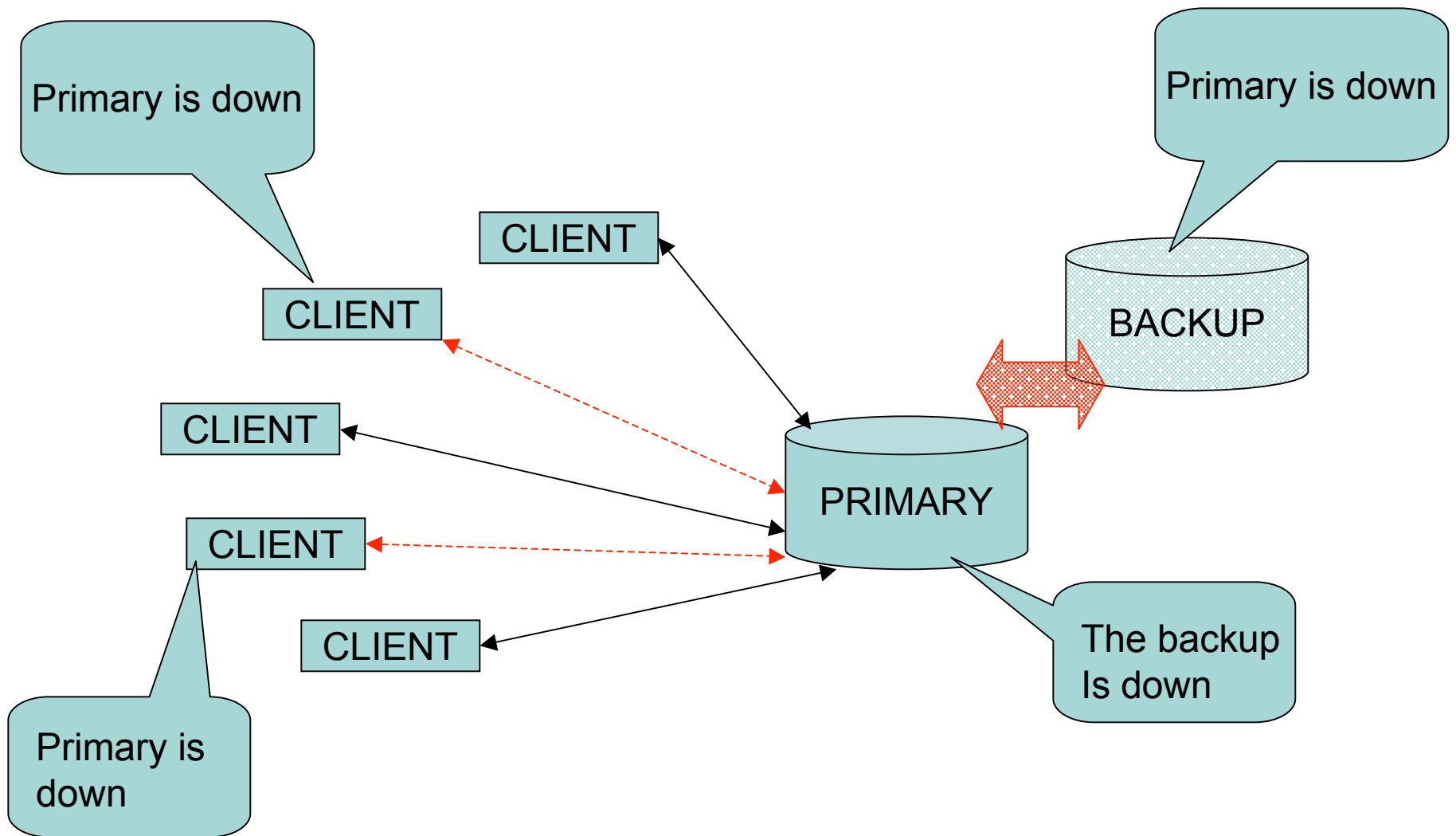


Normal case

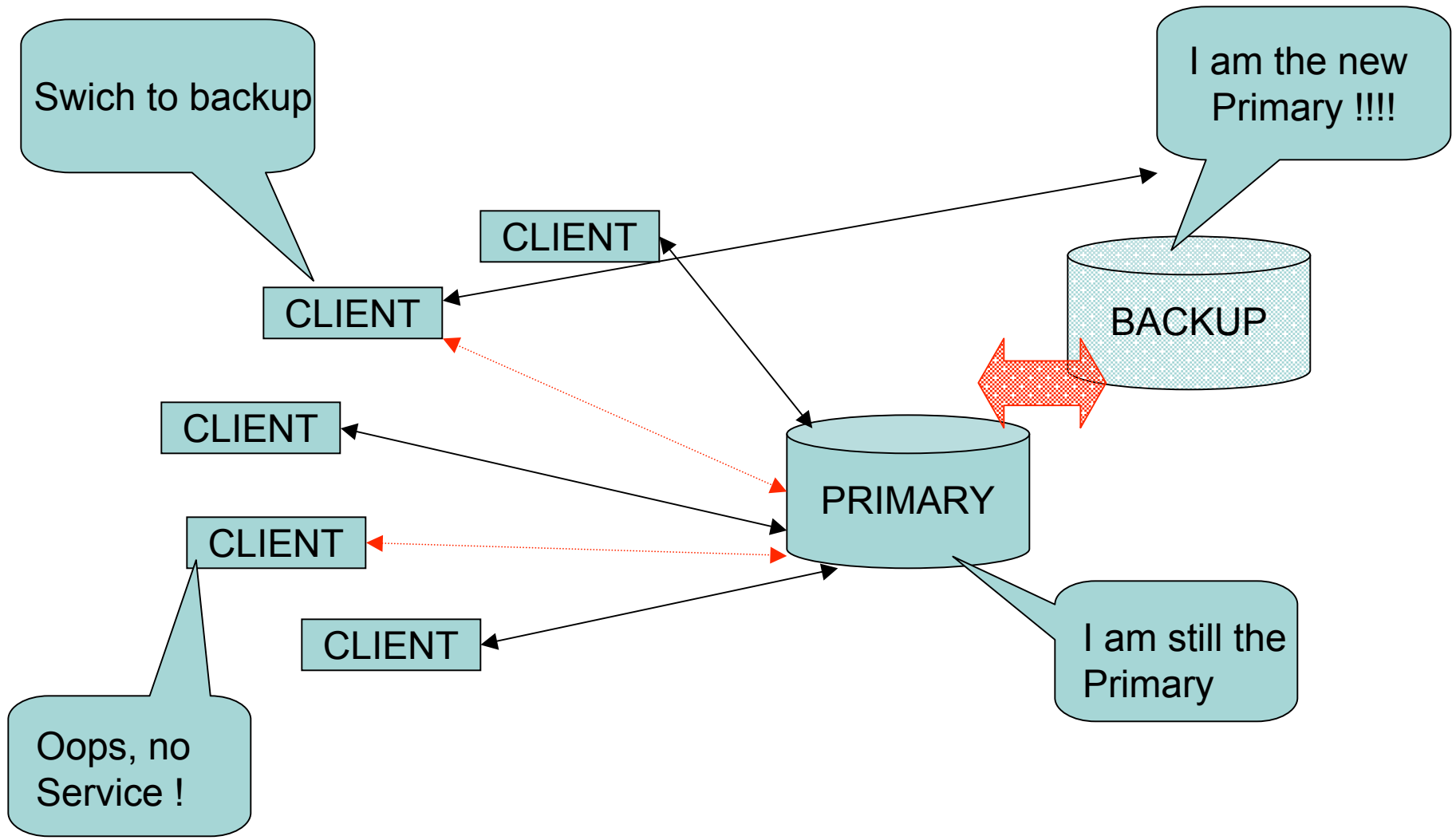
- Everybody connected to the backup, no problems



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?

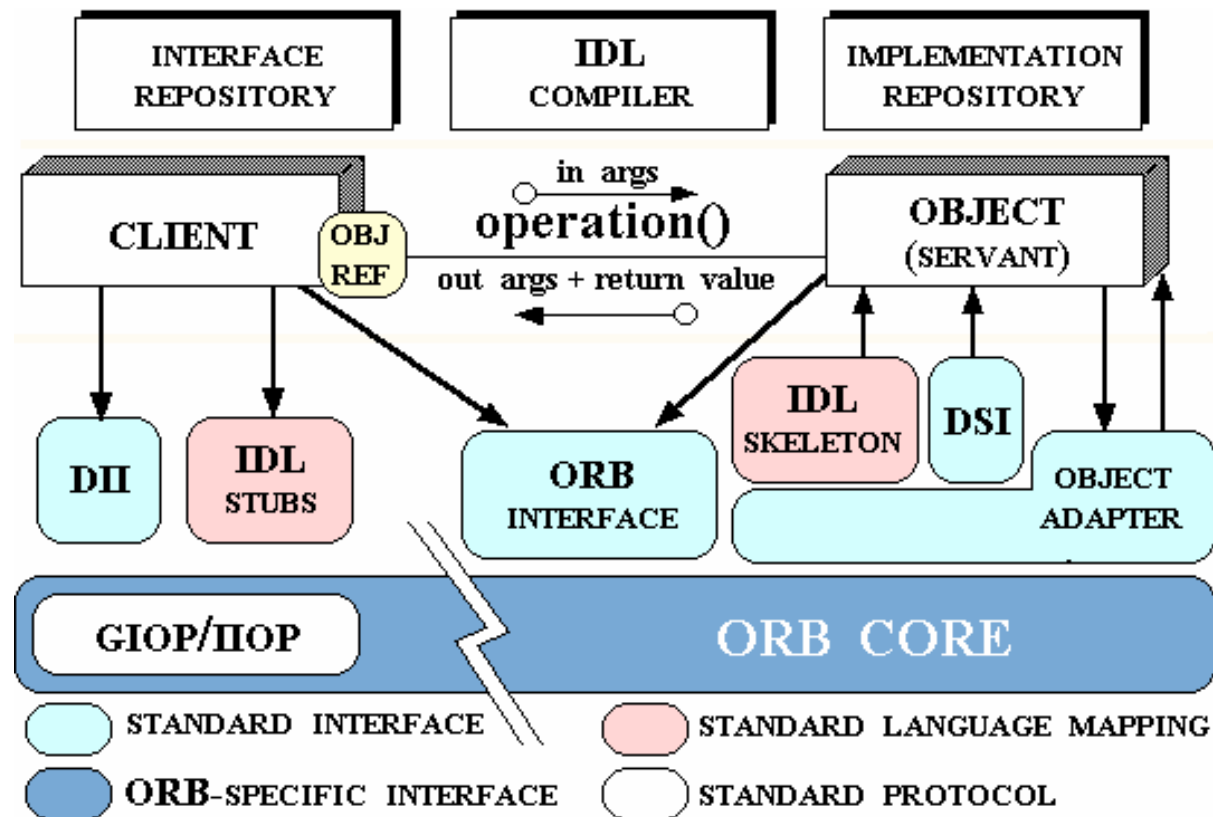


CORBA

- GOAL: Facilitate the development of distributed object-oriented applications.
- Common Object Request Broker Architecture (CORBA): provides a **standard** for open distributed object computing
- Examples of Services:
 - Object registration, location, and activation;
 - request demultiplexing;
 - error-handling;
 - parameter marshalling and demarshalling
- **SPECIFICATION vs IMPLEMENTATION:** many CORBA implementations, some open source, some proprietary

Architecture

- Objects provide different services.
- Object interface must be known (public) - provides signature for each object method.
- A client makes a request to an object for a service.
- Client doesn't need to know where the object is, or anything about how the object is implemented!
- Remote methods: static invocation or dynamic invocation



Picture from TAO Project webpage: <http://www.cs.wustl.edu/~schmidt/corba-overview.html>

Interface Definition Language

- IDL: declarative language used to describe object interfaces
- IDL is language neutral - there are mappings for many object oriented languages (C++, Smalltalk, Java).

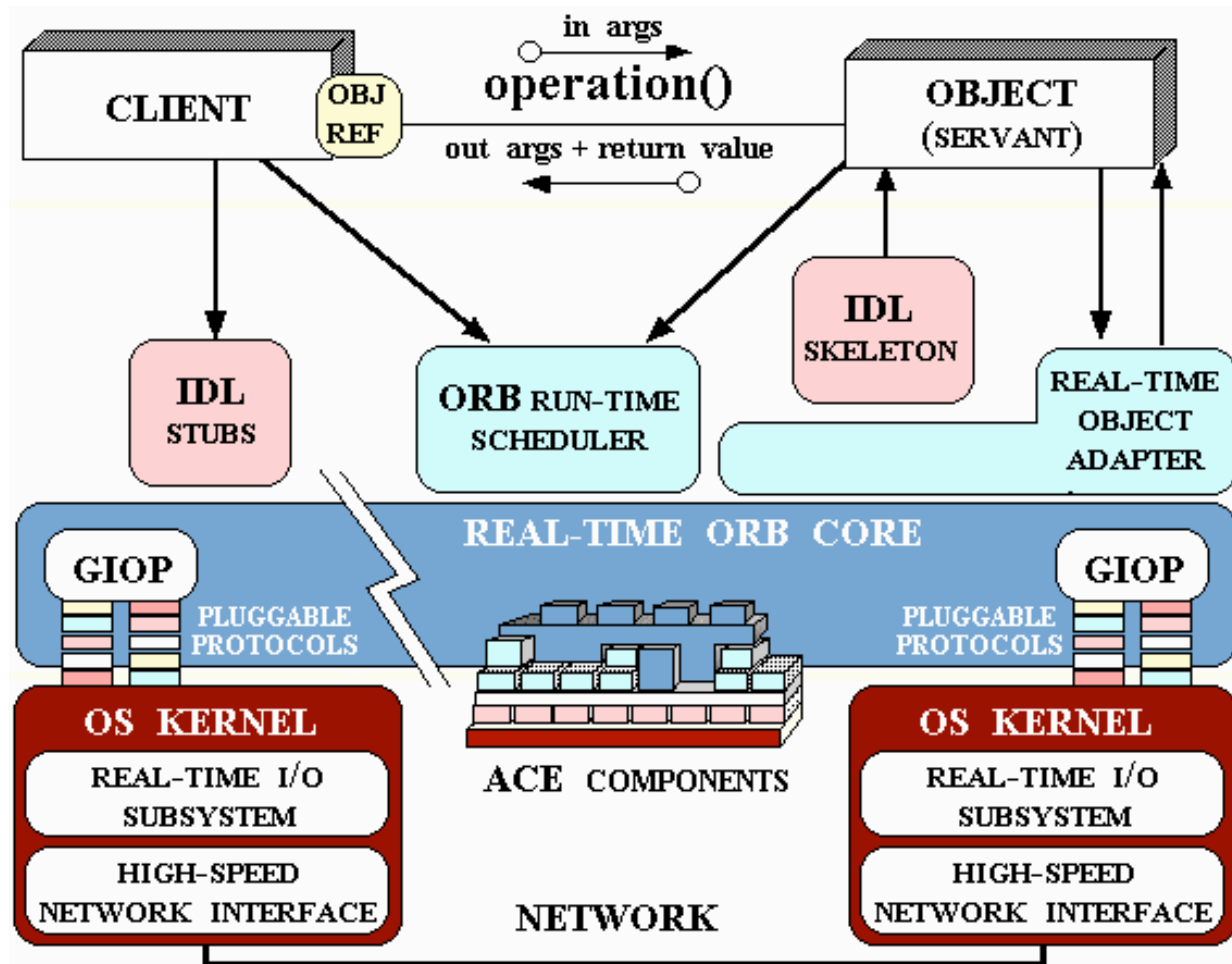
Object Request Broker

- The ORB acts as the middleman in all remote method invocations.
- The ORB finds a server that can handle a method invocation, passes the request to the server, receives the response and forwards it to the client.
- The functions handled by an ORB are actually *implemented* in both client and server.

Reliability Concepts in CORBA

- Recovering from failures: when invoking a remote server
- Transactional architecture:
 - Provides support for applications to perform operations on persistent objects
 - Does not affect other concurrently and independent operations
 - Persistent objects remains intact even if a failure happens during the operation

TAO: Real-Time Implementation of CORBA



Picture from TAO Project webpage: <http://www.cs.wustl.edu/~schmidt/TAO-intro.html>

Next week

- Continue with technologies, overview of Web services
- Will start looking into details of overcoming failures in a distributed system (chapter 14)
- Start the project, make sure your account is active and you understand the project requirements