Cristina Nita-Rotaru

CS505: Distributed Systems

Introduction. Syllabus. Class Policy.

Outline

- Emergency policies
- Cheating policies
- Class information
- Syllabus overview
- Examples of distributed systems
- Basic communication services

EMERGENCY PREPAREDNESS – A MESSAGE FROM PURDUE

To report an emergency, call 911. To obtain updates regarding an ongoing emergency, sign up for Purdue Alert text messages, view www.purdue.edu/ea.

There are nearly 300 Emergency Telephones outdoors across campus and in parking garages that connect directly to the PUPD. If you feel threatened or need help, push the button and you will be connected immediately.

If we hear a fire alarm during class we will immediately suspend class, evacuate the building, and proceed outdoors. Do not use the elevator.

If we are notified during class of a Shelter in Place requirement for a tornado warning, we will suspend class and shelter in [the basement].

If we are notified during class of a Shelter in Place requirement for a hazardous materials release, or a civil disturbance, including a shooting or other use of weapons, we will suspend class and shelter in the classroom, shutting the door and turning off the lights.

Please review the Emergency Preparedness website for additional information.
http://www.purdue.edu/ehps/emergency_preparedness/index.html

Policies for Cheating

- It is allowed/encouraged to discuss homework problems and projects
- Considered cheating:
  - if you look at another student's written or typed answers, or let another student look at your written or typed answers
  - give your code to another student, have somebody's else code in your possession; have very similar lines of code
- ZERO TOLERANCE: If caught cheating you received automatically a failing grade
- Use accounts protected by password, don't give your password to anyone

Course Information

- Meetings
  - MW 4:30-5:45pm REC 121
- Professor contact info:
  - Office: LWSN 2142J
  - Email: crisn@cs.purdue.edu
  - Office hours: by appointment
- Class webpage
- http://homes.cerias.purdue.edu/~crisn/courses/cs505_Fall_2014/

- We will use piazza for class communication

Grading Policy

Written homework	10%
Programming projects	35%
Midterm	20%
Final exam	30%
Class participation	5%

Written Assignments

- 4 written theoretical assignments
- Every student gets 5 extra days that he can use for these assignments. Email me with name and number of extra days used for an assignment. After using your 5 extra days, no late homework will be accepted.
- NO HANDWRITTEN ASSIGNMENTS, PDF ONLY.

Programming Projects

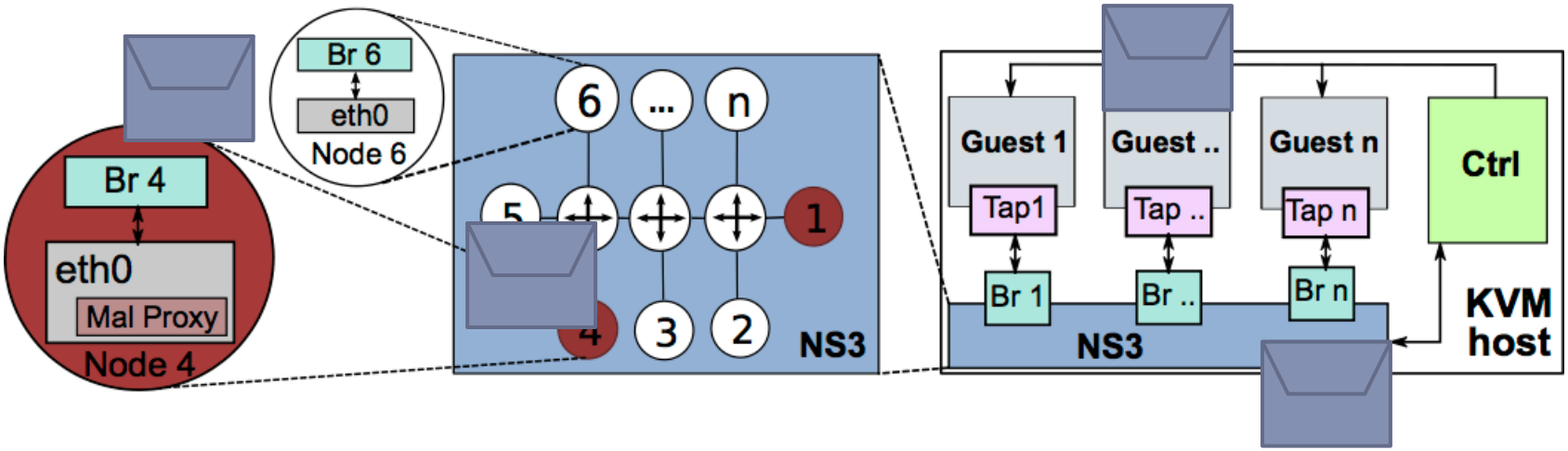
- 3 programming projects
- No extra days for programming projects
- Programming projects are individual
- Programming projects are assigned by me
- Programming language: C/C++
- Testing environment Turret

Turret

- Test unmodified implementations
- Use an environment as close as possible to the deployment environment
  - OS, libraries, etc.
- Reproducible results

- http://wave.cs.purdue.edu:8911/index.php

Turret Overview



Turret: A Platform for Automated Attack Finding in Unmodified Implementations of Distributed Systems H. Lee, J. Seibert, C. Killian, and C. Nita-Rotaru. ICDCS 2014.

Individual Meeting

- You are required to meet with me at least once per semester
- I will send doodle links with available time slots that you can sign up for, starting with next week
- You can always set up additional appointments by sending me an email first

Reference Material

- Textbooks
  - Ken Birman: Reliable Distributed Systems
- Recommended reading
  - Research papers that will be specified for each lecture

Your work this week

- No homework assigned this week
- Revisit C/C++ and socket programming
- Read all assigned reading
- Check out Turret website
- **Turret tutorial in class Wednesday Sept. 3 – Required to attend**

What is a Distributed System?

A distributed computing system is a set of computer programs executing on one ore more computers and coordinating actions by exchanging messages.

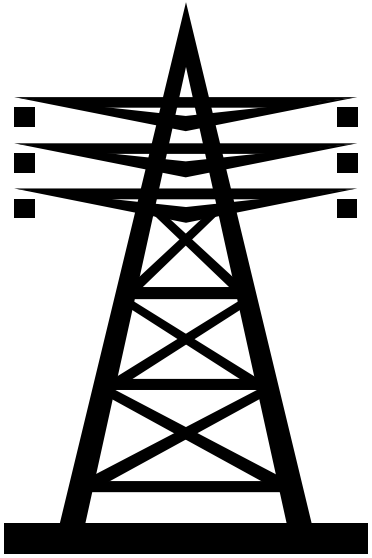
Or …

- A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable

- Attributed to Leslie Lamport

Examples of Distributed Systems

- Air Traffic Control
- Space Shuttle
- Banking Systems
- Grid Power Systems
- Modern Data Centers



Distributed Systems Requirements

- **Reliability**: provide continuous service
- **Availability**: ready to use
- **Safety**: systems do what they are supposed to do, avoiding catastrophic consequences
- **Security**: withstands passive/active attacks from outsiders or insiders

…not easy to achieve because

- Computers and networks fail in many (often unpredictable) ways
- Computers get compromised
- Real-time constraints
- Performance requirements
- Complexity

Why Do Computer Systems Fail?

- 1985, Fault-tolerant system (Tandem)
  - System administration (operator actions, system configuration and maintenance)
  - Software faults, environmental failures
  - Hardware failures (disks and communication controllers)
  - Power outages
- 2011, The Internet Age
  - Operator error (particularly configuration errors) is the leading cause of failures
  - Failures in custom-written front-end software
  - Not enough on-line testing

Why Do Computers Get Compromised?

- Software bugs
- Administration errors
- Lack of diversity, same vulnerability is exploited
- The explosion of the Internet facilitates the spread of malware

..how do computer system fail...

- Halting failures: no way to detect except by using timeouts
- Fail-stop failures: accurately detectable halting failures
- Send-omission failures
- Receive-omission failures
- Network failures
- Network partitioning failures
- Timing failures: temporal property of the system is violated
- Byzantine failures: arbitrary failures, include both benign and malicious failures

Cristina Nita-Rotaru

Example: Air Traffic Control

Prepared with slides courtesy of Prof. Ken Birman and used in a similar course at Cornell University

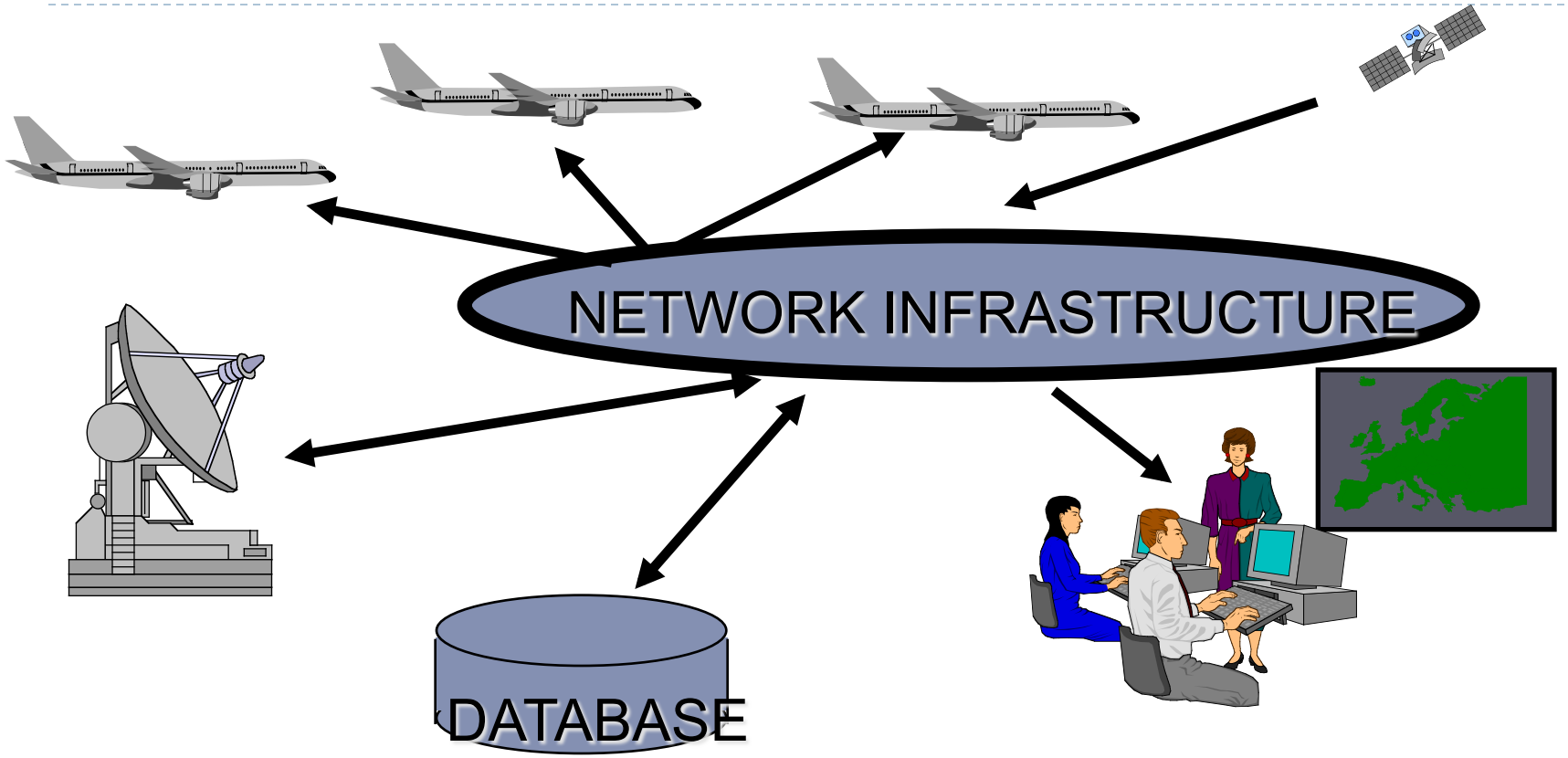
ATC and Its Role

- Assists planes in taking-off, landing and en route (during flying)
- Assigns trajectories making sure that planes fly at a safe distance
- Each ATC has a certain space assigned to it
- As planes move they enter the space controlled by different ATCs
- Planes are also equipped with a collision avoidance system TCAS

More Details on ATC

- Air space divided in sectors
- 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 "own" smaller sub-sectors
- Controllers make very quick decision(s) based on available data

ATC Architecture



THE SYSTEM MUST BE AVAILABLE ALL TIME and MAINTAIN CONSISTENCY OF THE INFORMATION

What Can Go Wrong?

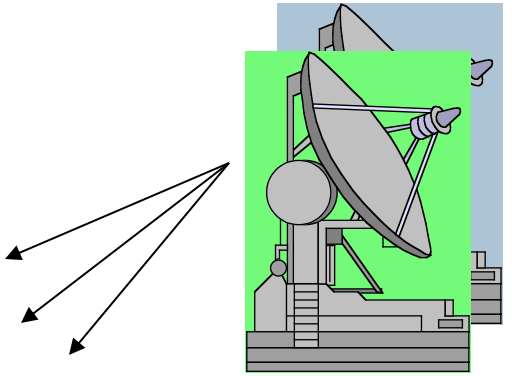
- Overloaded computers can often crash
- Systems may get slow as volume of air traffic rises
- Inconsistent displaying:
  - phantom planes
  - missing planes
  - stale information
- Scheduled maintenance going wrong
- Some major outages recently (and some near-miss stories associated with them)

Concept of IBM's 1994 System

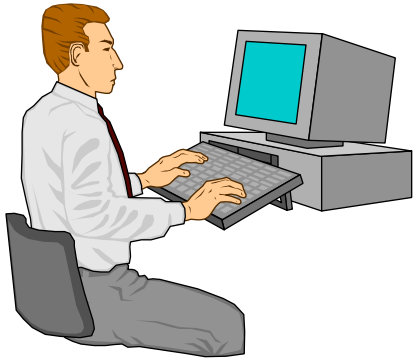
- Replace video terminals with workstations
- Build a highly available real-time system guaranteeing no more than 3 seconds downtime per year
- Offer much better user interface to ATC controllers, with intelligent course recommendations and warnings about future course changes that will be needed
- IBM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

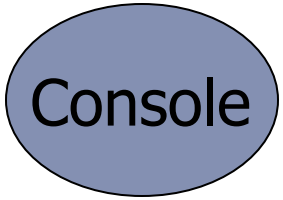
IBM ATC System Architecture

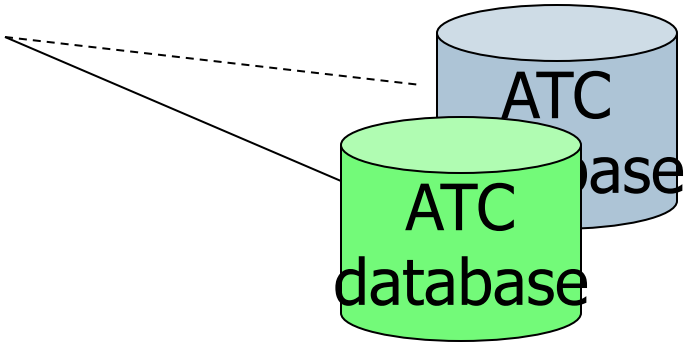
Independent consoles… backed by ultra-reliable components



Radar processing system is redundant







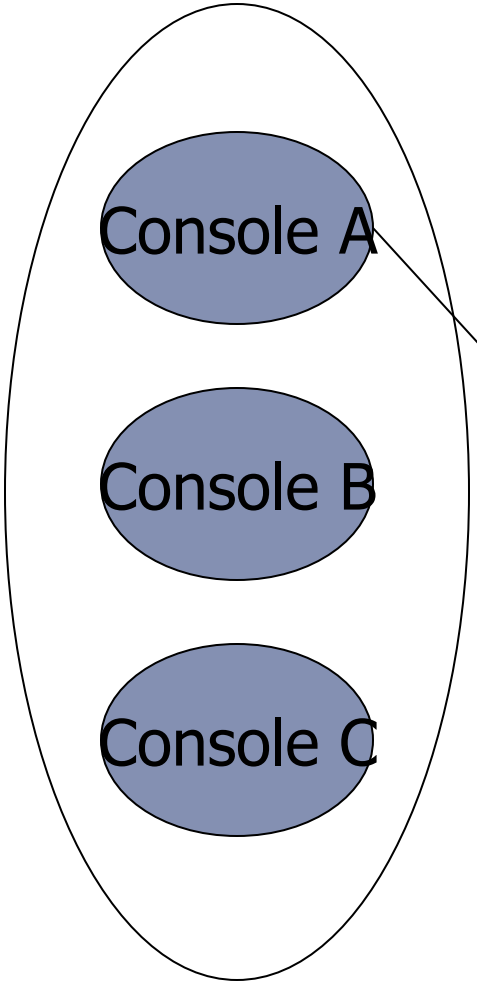
ATC database is really a high-availability cluster

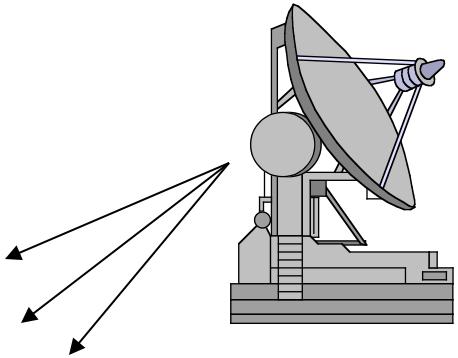
French ATC Project Concept

- French project used replication selectively
- Some specific and critical data was replicated, 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
  - Program "hosts" a data replica but isn't itself replicated

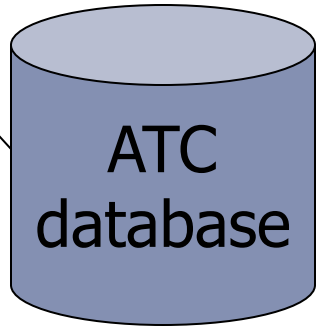
French ATC System Architecture

Multiple consoles… but in some ways they function like one





Radar updates sent with hardware broadcasts



ATC database only sees one connection

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

Cristina Nita-Rotaru

Example: Google File System

Application Characteristics

- Hundreds of clients must perform concurrent (atomic) appends with minimal synchronization
- Sustained bandwidth more important than latency
- Response time for individual read/write not important
- Non-traditional access patterns
- Files are very big, several gigs

Design Choices

- Fault-tolerance is a must; Constant monitoring, error detection, automatic recovery part of the design
- Designed for big files. I/O operations and block sizes have to be revisited.
- Non-traditional access patterns:
  - Most files are modified by appending rather than overwriting;
  - Large repositories that must be scanned (archival, streams, intermediate data)
  - Result: appending is the focus of optimization
- Co-design applications and file system; looser consistency

Interface Design

- Not standard API (such as POSIX)
- Supports file/directory hierarchy and the usual: {create, delete, open, close, read, write}
- Additionally:
  - snapshot: low cost file / directory tree copying
  - record append: concurrent appends, no locks!

Architecture Overview

- No files, base unit is chunk:
  - Fixed-part of a file, typically 64 MB
  - Global ID: 64 bit, unique "chunk handle", assigned by master server upon chunk creation
  - Read/Write: need chunk handle + byte range
- Servers:
  - Single master
  - Multiple backups (chunkservers)
- Multiple clients

Design Overview: Consistency Model

- File namespace modifications: atomic & handled only by master server
- Chunks:
  - "consistent" if all clients see same data, no matter which replica they ask
  - "defined" if it is consistent and known, i.e. some modification done w/o interruption
  - "undefined" if concurrent modifications are successful
  - "inconsistent" on any failed modification
  - Inconsistent regions may be padded or contain duplicates

Design Overview: Consistency Model

- File namespace modifications: atomic & handled only by master server
- Chunks:
  - "consistent" if all clients see same data, no matter which replica they ask
  - "defined" if it is consistent and known, i.e. some modification done w/o interruption
  - "undefined" if concurrent modifications are successful
  - "inconsistent" on any failed modification
  - Inconsistent regions may be padded or contain duplicates

New Google Storage System

- New requirements: highly interactive applications (for example email)
- Available all the time
- Wide-area network
- One of the biggest changes was using active replication – based on PAXOS, proven, optimal, fault-tolerant consensus algorithm with no requirement for a distinguished master

Cristina Nita-Rotaru

What is 505 about?

Course Overview (1)

- How and why computers systems fail. How to overcome failures in a distributed system. Failures models. The distributed commit problem.
- Dynamic membership. Replicating data with malicious failures. Impossibility of asynchronous consensus.
- Group communication systems, properties and dynamic group membership. Causal and total order.

Course Overview (2)

- Virtual Synchrony and extended virtual synchrony.
- Virtually synchronous algorithms and tools: replicated data, state transfer, load-balancing, primary-backup and coordinator-cohort fault tolerance.
- Architectures for group communication systems.
- Clock synchronization and synchronous systems.
- Transactional model and implementation of a transactional storage systems. Distributed transactions and multiphase commit.
- Weak consistency models. Weak and strong consistency in partitioned database systems. Linearizability.

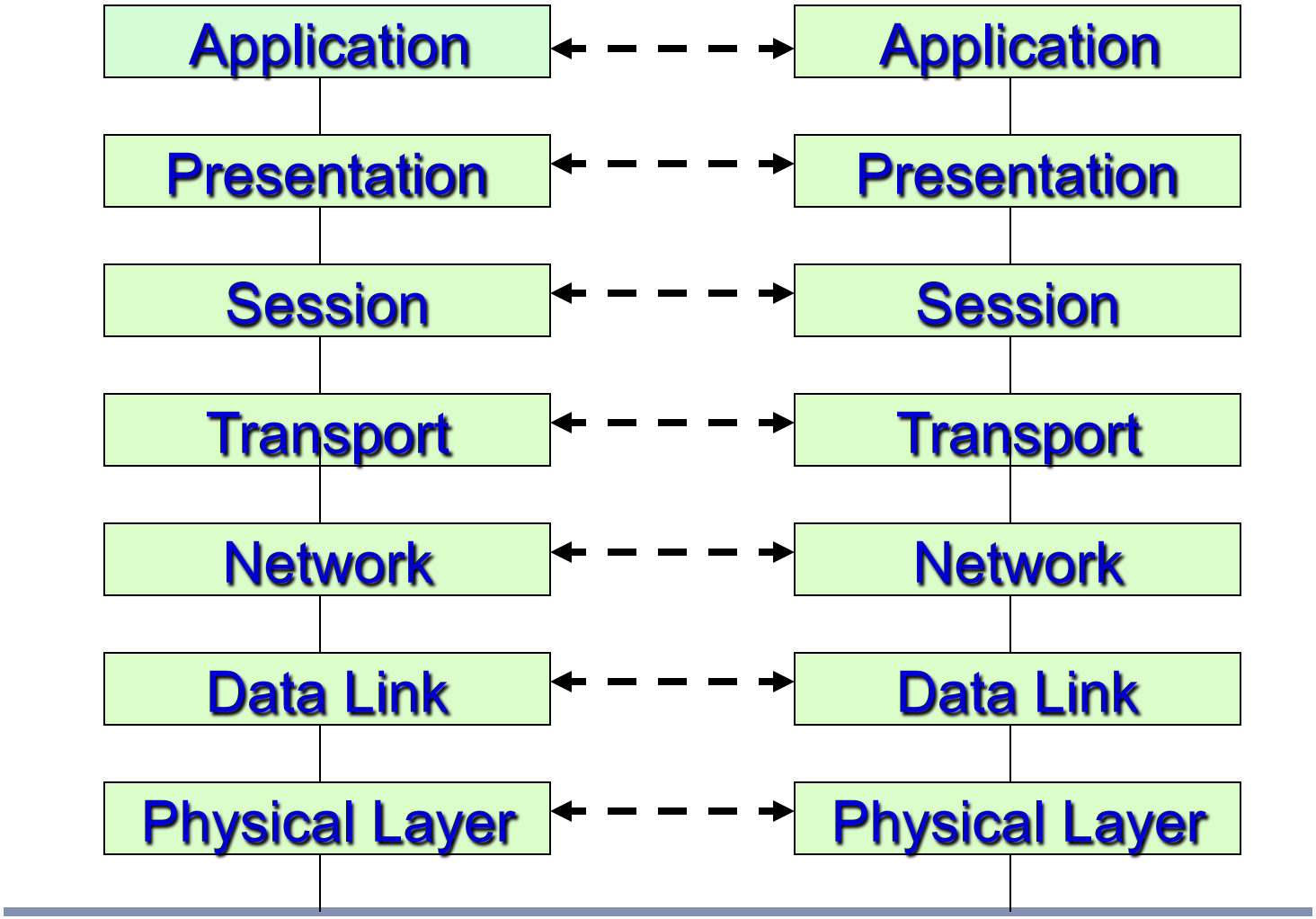
Course Overview (3)

- Peer-to-peer file sharing.
- Peer-to-peer distributed indexing.
- Building highly assured web services.
- Google storage system
- Haystack, Facebook picture storage system
- MapReduce
- Security issues in group communication systems.
- Security topics in overlay networks.
- Aspects of distributed algorithms in wireless networks

Cristina Nita-Rotaru

Basic communication services

OSI/ISO Model



Types of addresses

- Media Access Control (MAC) addresses in the network access layer
  - Associated with network interface card (NIC)
  - 48 bits or 64 bits
- IP addresses for the network layer
  - 32 bits for IPv4, and 128 bits for IPv6
  - E.g., 128.3.23.3
- IP addresses + ports for the transport layer
  - E.g., 128.3.23.3:80
- Domain names for the application/human layer
  - E.g., www.purdue.edu

Routing and translation of addresses

- Translation between IP addresses and MAC addresses
  - Address Resolution Protocol (ARP) for IPv4
  - Neighbor Discovery Protocol (NDP) for IPv6
- Routing with IP addresses
  - TCP, UDP, IP for routing packets, connections; IP communication between hosts, TCP and UDP between processes
  - Border Gateway Protocol for routing table updates
- Translation between IP addresses and domain names
  - Domain Name System (DNS)

NATs and their implications

- There are not enough IP addresses
- Solutions: IPv6 or ….Network Address Translation (NAT)
- NAT allows a single device, to act as an agent between the Internet (or "public network") and a local (or "private") network: only a single, unique IP address is required to represent an entire group of computers
- Computers can not communicate directly, STUN client-server protocol allows computers to discover each other behind a NAT (learn their public addresses), but requires presence of STUN server

Problems with NATs

- Break end-to-end control
- Hosts depend on same trusted point (the STUN server)
- Add complexity
- Prevent IP security deployment

Address Resolution Protocol (ARP)

- Interface between Link layer and Network Layer
- Allows hosts to query who owns an IP address on the same LAN
- Owner responds with hardware address
- Allows changes to link layer to be independent of IP addressing

Internet Protocol - IP

- IP is the current delivery protocol on the Internet
- Provides communication between hosts
- IP provides ‘best effort’, unreliable delivery of packets.
- There are two versions:
  - IPv4 is the current routing protocol on the Internet
  - IPv6, a newer version

Transport Protocols

- Provide communication between processes running on hosts
- The most common transport protocols are UDP and TCP.
- OS provides support for developing applications on top of UDP and TCP.

User Datagram Protocol - UDP

- Connectionless protocol for a user process:
  - No connection established
  - Unreliable transmission: no guarantee that the packets reach their destination
  - No guarantee on the order of packets
  - Error detection
- Runs on top of IP.

Transmission Control Protocol - TCP

- Connection oriented protocol for a user process:
  - Reliable, full-duplex channel: acknowledgements, retransmissions, timeouts, flow-control
  - The packets are delivered in the same order in which they were sent.
  - Flow Control: Max allowed window size
  - Congestion control:
    - Slow-start phase – exponential increase (until the slow-start threshold is hit)
    - Congestion Avoidance phase – additive increase
    - Multiplicative Decrease on timeout.

IP Multicast

- Provides support for group communication: send to multiple parties
- Groups are specified by reserved IP multicast addresses 224.0.0.0 to 239.255.255.255.
- Unreliable communication
- IGMP is used to dynamically register individual hosts in a multicast group on a particular LAN.
- Network cards recognize IP multicast addresses: hosts that did not subscribe to a particular group will not process those packets (unlike broadcast that is processed by all hosts in a network segment)
- Issues with IP multicast: can be used to cause DOS

Byte Order

- Different systems store multibyte values (for example int) in different ways.
  - HP, Motorola 68000, and SUN systems store multibyte values in Big Endian order: stores the high-order byte at the starting address
  - Intel 80x86 systems store them in Little Endian order: stores the low-order byte at the starting address.
- Why is this a problem for network applications? Data is interpreted differently on hosts with different architectures.

Buffering and Fragmentation

- Buffering: OS maintains a set of buffers used to temporary store incoming and outgoing messages
- The buffering space is limited
- Fragmentation: IP datagrams are fragmented, they can travel on different paths
- When processes send very fast, packets can be dropped by the OS without any notification
  - On sending: no OS memory can be obtained for one or several fragments
  - On receiving: one or several fragments did not make it to the destination, entire datagram is dropped

Why these protocols do not provide better support for distributed applications?

End to End Arguments in System Design.
Saltzer, Reed, Clark TOCS 1990.

What is all about?

- Analyzes what services should be provided at low levels and what should be provided by the application
  - Commonly cited as a justification for not addressing reliability at low levels and let application handle it
- Example: How to transfer a file: hop-by-hop or end-to-end?
- 1) Low-level mechanisms should focus on speed, not reliability
- 2) The application should worry about "properties" it needs

Cristina Nita-Rotaru

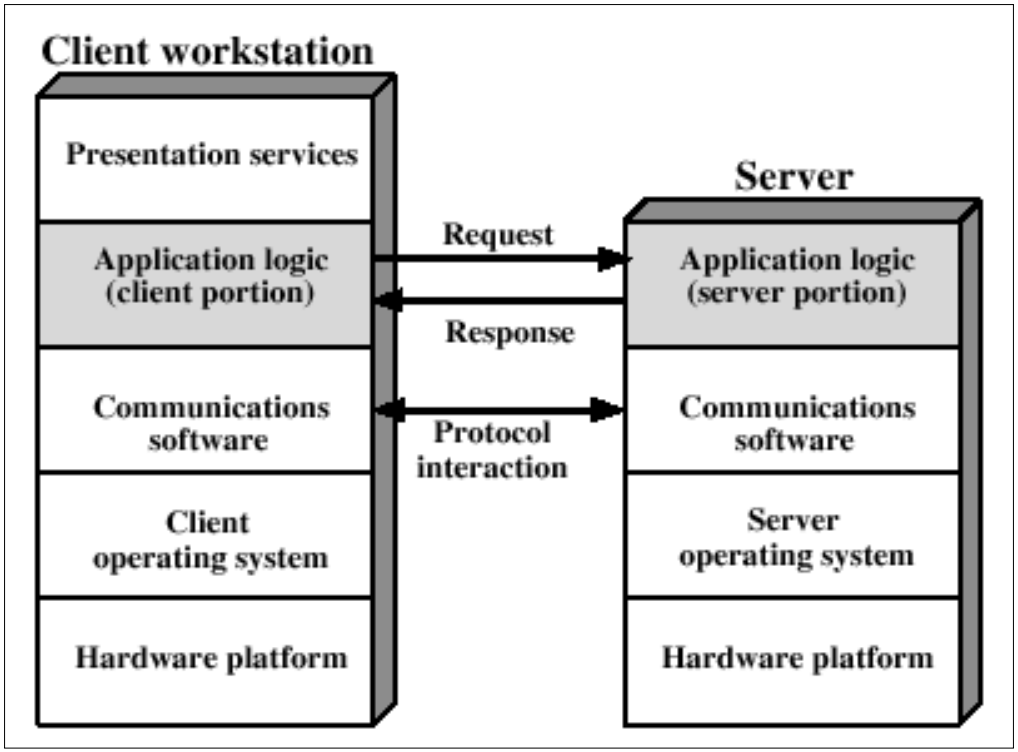
Basic communication services: RPC

Client/Server Architecture

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

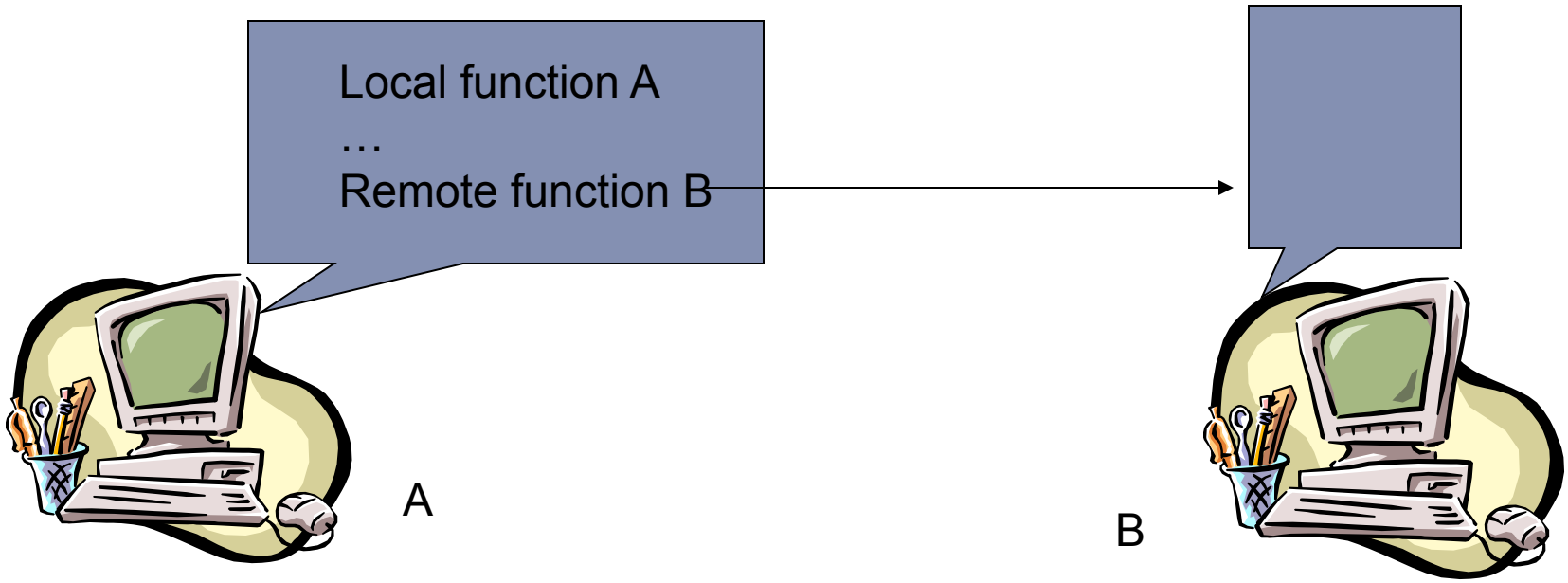


Types of Client/Server

- 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.

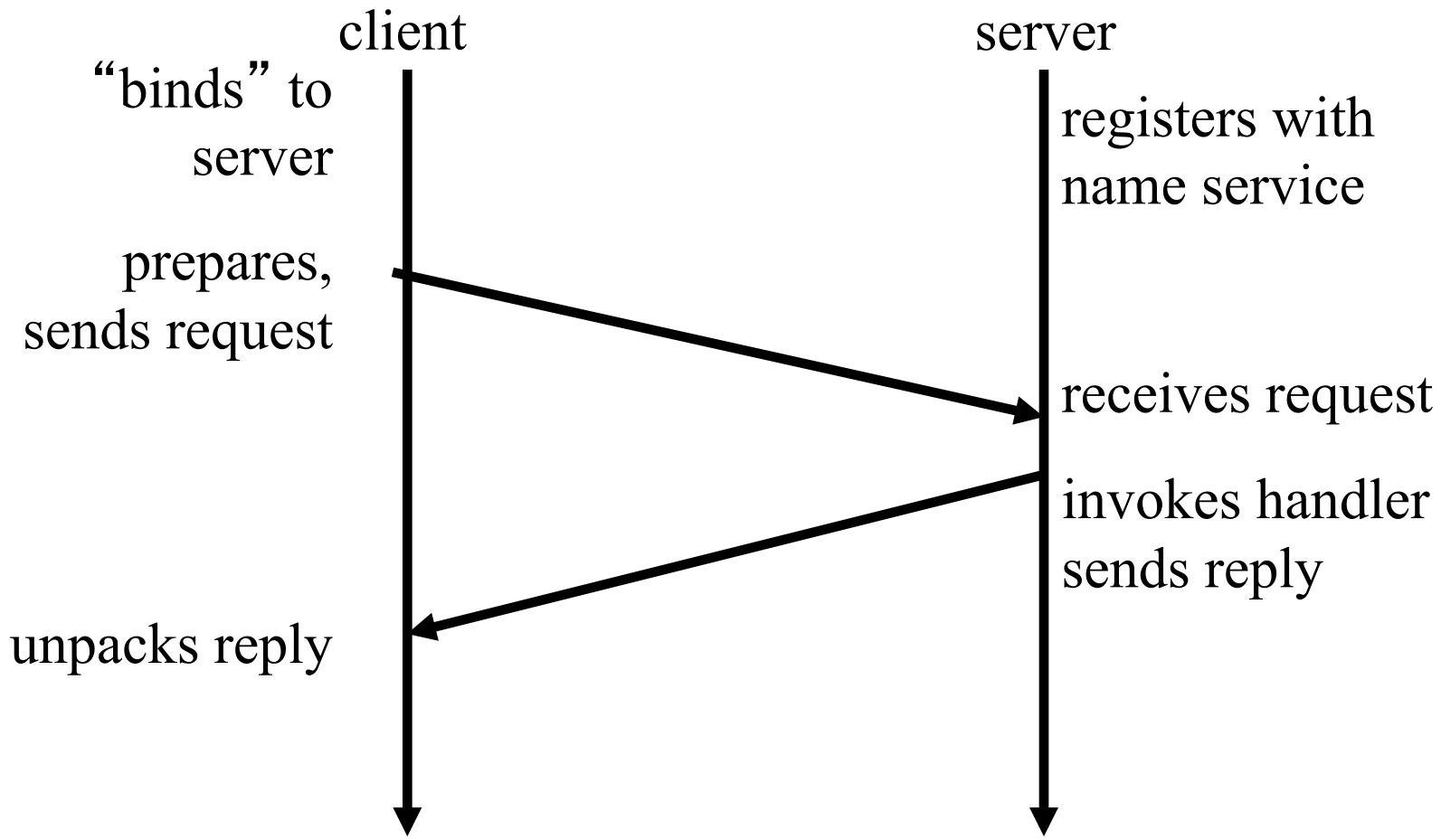
- Can you think about examples in each case?

Example: 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 1984)

The Basic RPC Protocol



RPC

- Idea of RPC goes back as far as 1976
- 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:
  - RPC server publishes 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

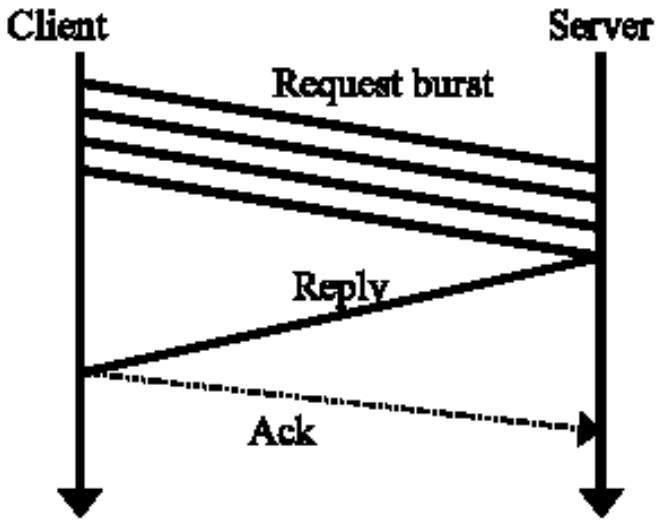


Figure 4.4. RPC using a burst protocol; here the reply is sent soon enough so that an acknowledgement to the burst is not needed.

- Delay sending acks, so that imminent reply itself acts as an ack.
- Do not 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
  - Long delay
  - Machine failure!
- How long should the client wait?

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

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 does not know whether the server processed its request.

RPC Semantics

- "Exactly Once"
  - Each request handled exactly once
  - Impossible to satisfy, in the face of failures.
  - Cannot 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)

Required Reading

- Chapter 1, 2 and 3 from Reliable Distributed Systems
- Why do Internet services fail, and what can be done about it? D. Oppenheimer, A.Ganapathi and D.A. Patterson, 2003.
- Why Do Computers Stop and what can be done about it? Jim Gray, 1985
- **End to end arguments in system Design. Saltzer, Reed, Clark TOCS 1990.**