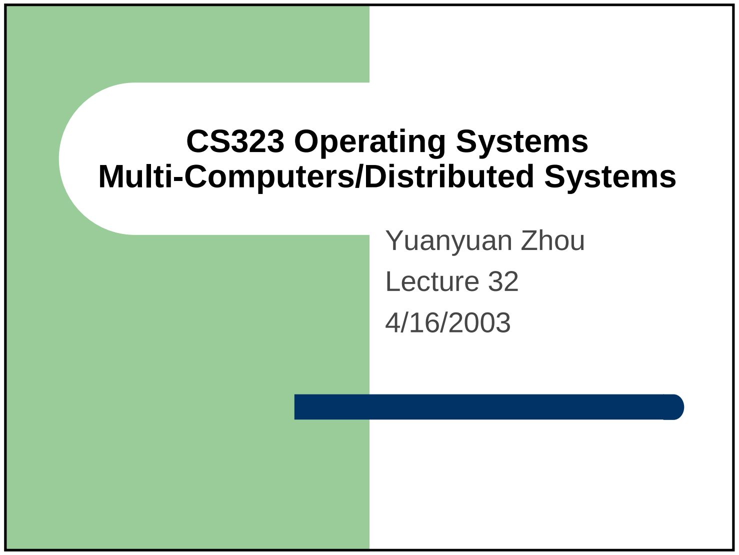
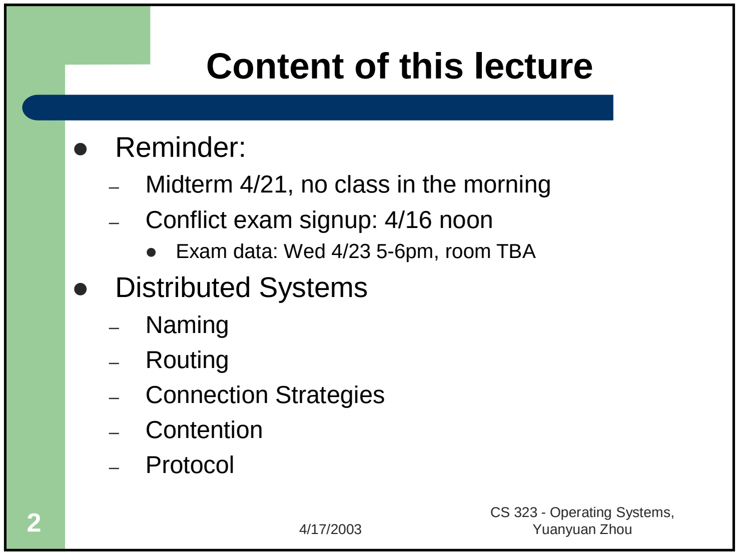


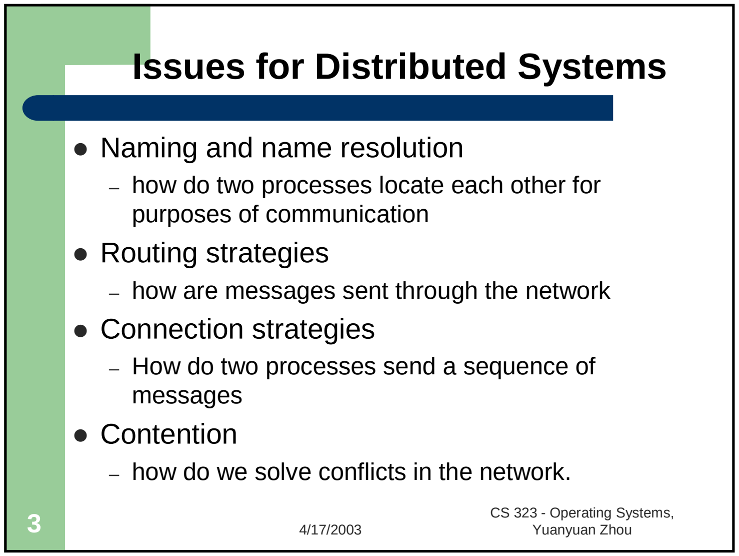
CS323 Operating Systems
Multi-Computers/Distributed Systems

Yuanyuan Zhou
Lecture 32
4/16/2003



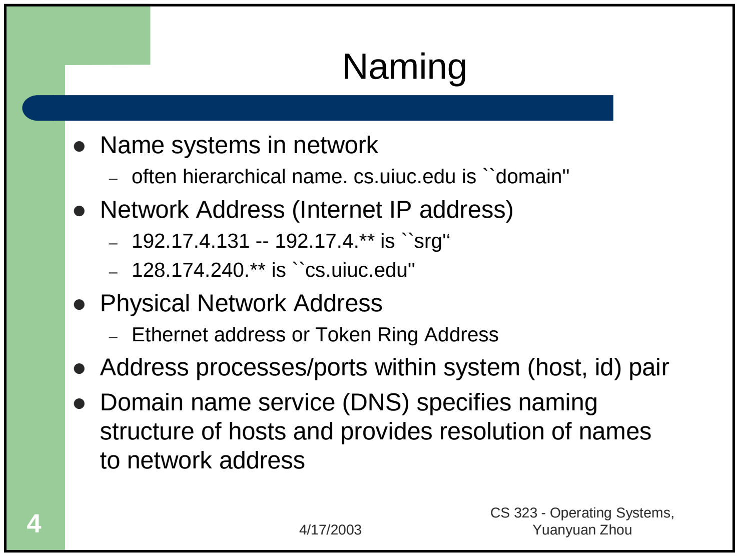
Content of this lecture

- Reminder:
  - Midterm 4/21, no class in the morning
  - Conflict exam signup: 4/16 noon
    - Exam data: Wed 4/23 5-6pm, room TBA
- Distributed Systems
  - Naming
  - Routing
  - Connection Strategies
  - Contention
  - Protocol



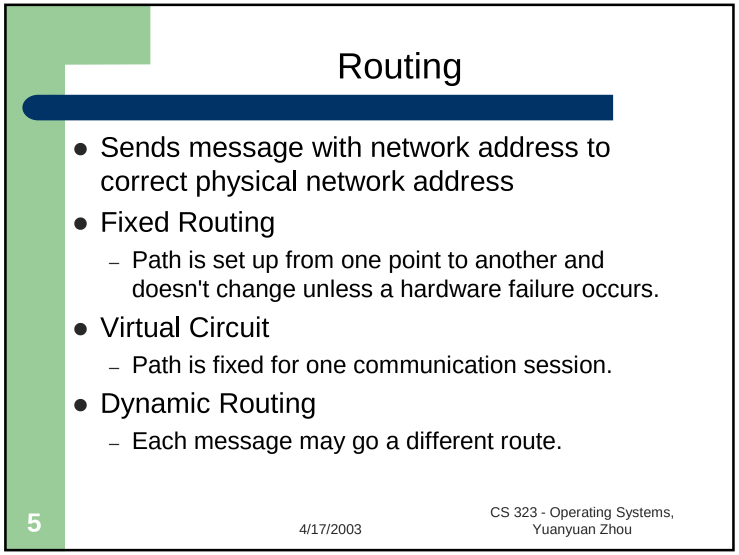
Issues for Distributed Systems

- Naming and name resolution
  - how do two processes locate each other for purposes of communication
- Routing strategies
  - how are messages sent through the network
- Connection strategies
  - How do two processes send a sequence of messages
- Contention
  - how do we solve conflicts in the network.



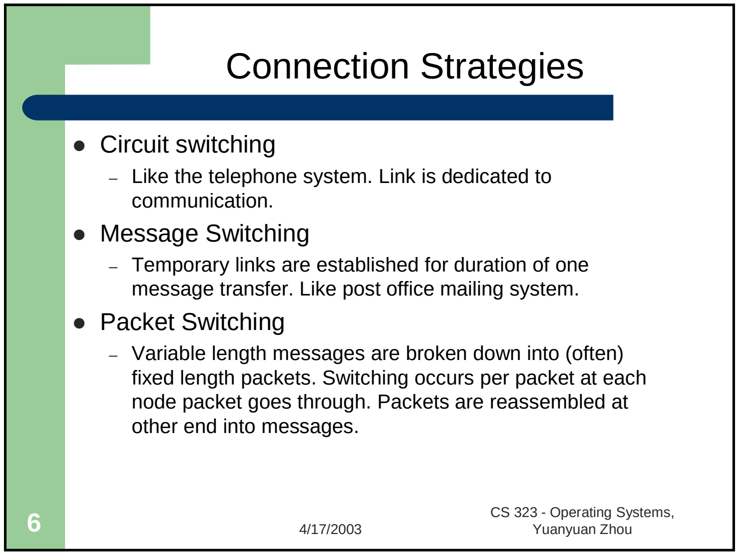
Naming

- Name systems in network
  - often hierarchical name. cs.uiuc.edu is ``domain''
- Network Address (Internet IP address)
  - 192.17.4.131 -- 192.17.4.** is ``srg''
  - 128.174.240.** is ``cs.uiuc.edu''
- Physical Network Address
  - Ethernet address or Token Ring Address
- Address processes/ports within system (host, id) pair
- Domain name service (DNS) specifies naming structure of hosts and provides resolution of names to network address



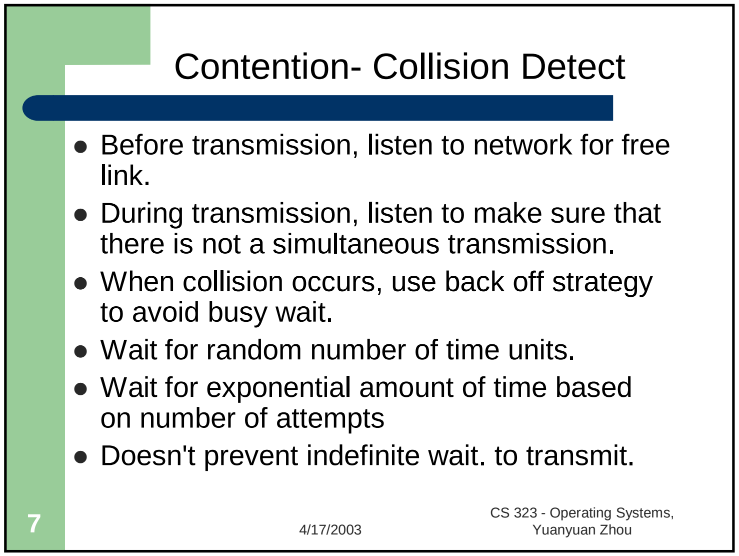
Routing

- Sends message with network address to correct physical network address
- Fixed Routing
  - Path is set up from one point to another and doesn't change unless a hardware failure occurs.
- Virtual Circuit
  - Path is fixed for one communication session.
- Dynamic Routing
  - Each message may go a different route.



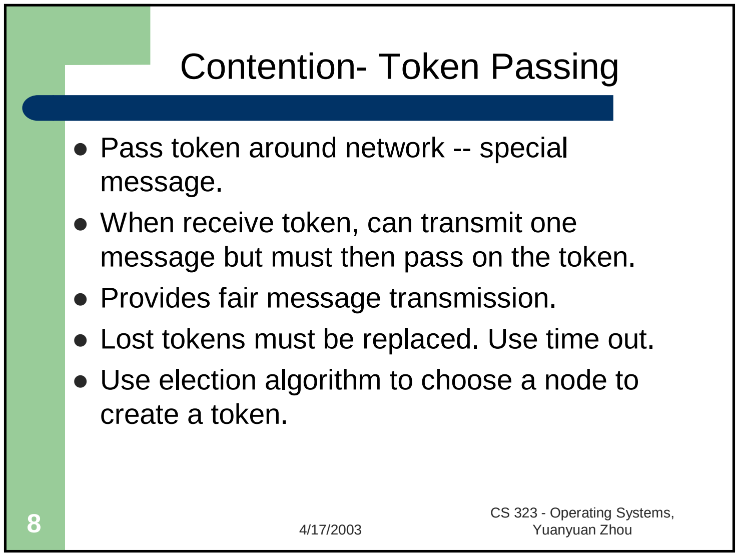
Connection Strategies

- Circuit switching
  - Like the telephone system. Link is dedicated to communication.
- Message Switching
  - Temporary links are established for duration of one message transfer. Like post office mailing system.
- Packet Switching
  - Variable length messages are broken down into (often) fixed length packets. Switching occurs per packet at each node packet goes through. Packets are reassembled at other end into messages.



Contention- Collision Detect

- Before transmission, listen to network for free link.
- During transmission, listen to make sure that there is not a simultaneous transmission.
- When collision occurs, use back off strategy to avoid busy wait.
- Wait for random number of time units.
- Wait for exponential amount of time based on number of attempts
- Doesn't prevent indefinite wait. to transmit.



Contention- Token Passing

- Pass token around network -- special message.
- When receive token, can transmit one message but must then pass on the token.
- Provides fair message transmission.
- Lost tokens must be replaced. Use time out.
- Use election algorithm to choose a node to create a token.

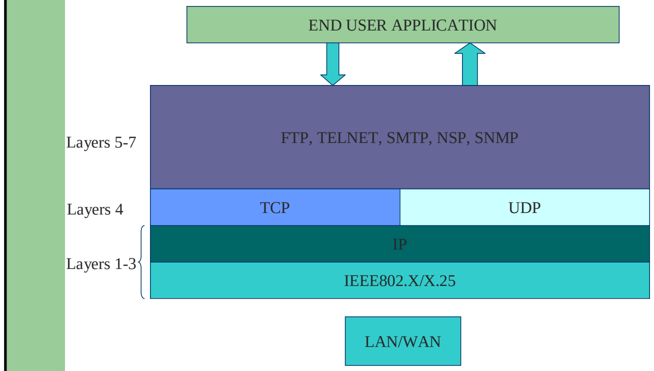
Contention- Message Slots

- A number of empty messages continuously circulate around the network.
- Node grabs empty slot, fills it with message.
- Receiving node removes message and replaces empty message.

Internet Protocol Stack

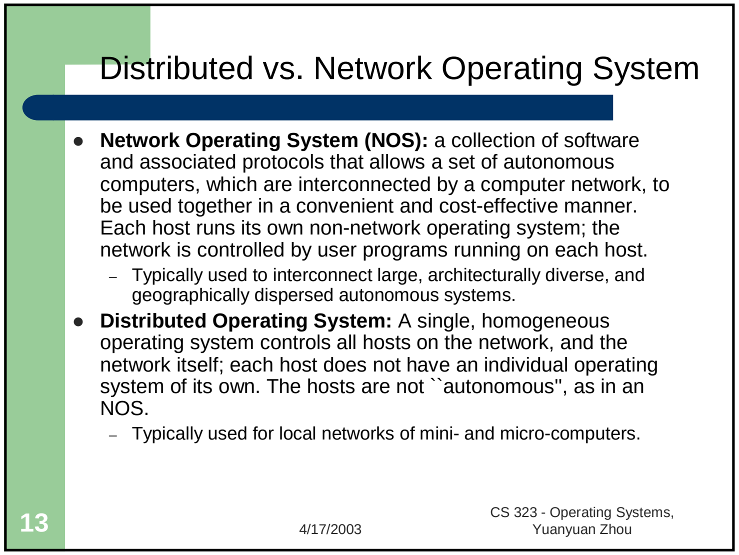
- Network Protocol
  - IP version 4, coming version 6.
  - This protocol is responsible for transmitting IP datagrams.
- Transport Protocols
  - User Datagram Protocol (UDP)
    - UDP/IP is an unreliable, connectionless transport protocol, which uses IP to transport IP datagrams but adds error correction and a protocol **port address** to specify the process on the remote system for which the packet is destined.
  - Transmission Control Protocol (TCP).
    - TCP/IP is a reliable stream protocol for communicating information between two processes

TCP/IP Protocol Layers



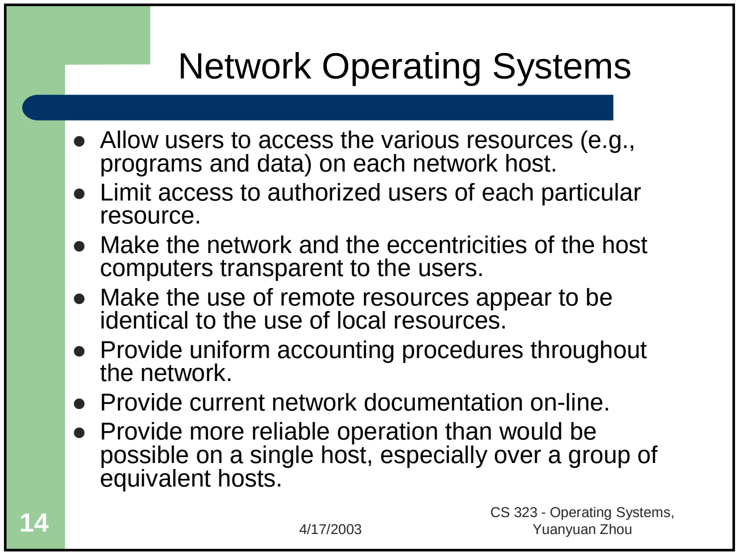
Robustness

- Failure Detection
  - hand-shaking and time-out schemes
- Reconfiguration
  - notification and update of routing tables
- Recovery from Failure
  - repaired links and site must be integrated into the system gracefully and smoothly



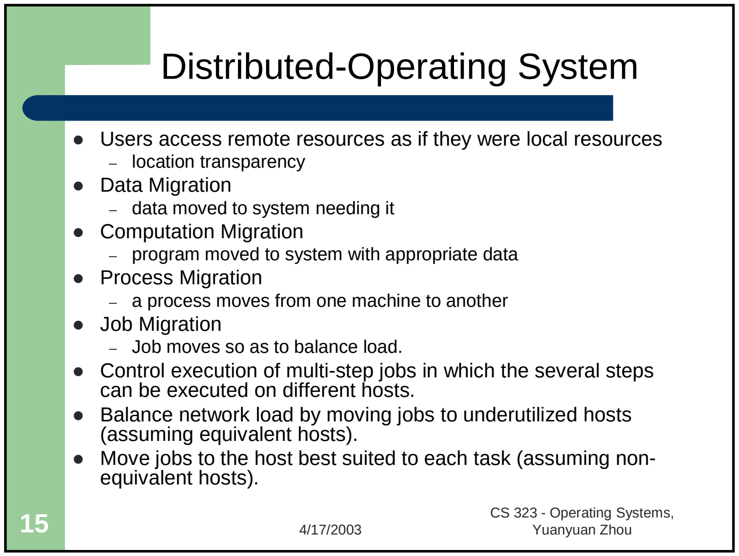
Distributed vs. Network Operating System

- **Network Operating System (NOS):** a collection of software and associated protocols that allows a set of autonomous computers, which are interconnected by a computer network, to be used together in a convenient and cost-effective manner. Each host runs its own non-network operating system; the network is controlled by user programs running on each host.
  - Typically used to interconnect large, architecturally diverse, and geographically dispersed autonomous systems.
- **Distributed Operating System:** A single, homogeneous operating system controls all hosts on the network, and the network itself; each host does not have an individual operating system of its own. The hosts are not ``autonomous'', as in an NOS.
  - Typically used for local networks of mini- and micro-computers.



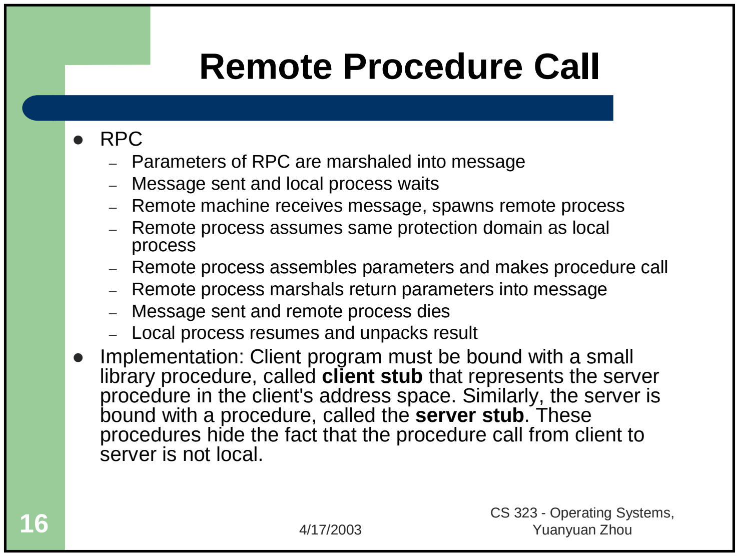
Network Operating Systems

- Allow users to access the various resources (e.g., programs and data) on each network host.
- Limit access to authorized users of each particular resource.
- Make the network and the eccentricities of the host computers transparent to the users.
- Make the use of remote resources appear to be identical to the use of local resources.
- Provide uniform accounting procedures throughout the network.
- Provide current network documentation on-line.
- Provide more reliable operation than would be possible on a single host, especially over a group of equivalent hosts.



Distributed-Operating System

- Users access remote resources as if they were local resources
  - location transparency
- Data Migration
  - data moved to system needing it
- Computation Migration
  - program moved to system with appropriate data
- Process Migration
  - a process moves from one machine to another
- Job Migration
  - Job moves so as to balance load.
- Control execution of multi-step jobs in which the several steps can be executed on different hosts.
- Balance network load by moving jobs to underutilized hosts (assuming equivalent hosts).
- Move jobs to the host best suited to each task (assuming non-equivalent hosts).



Remote Procedure Call

- RPC
  - Parameters of RPC are marshaled into message
  - Message sent and local process waits
  - Remote machine receives message, spawns remote process
  - Remote process assumes same protection domain as local process
  - Remote process assembles parameters and makes procedure call
  - Remote process marshals return parameters into message
  - Message sent and remote process dies
  - Local process resumes and unpacks result
- Implementation: Client program must be bound with a small library procedure, called **client stub** that represents the server procedure in the client's address space. Similarly, the server is bound with a procedure, called the **server stub**. These procedures hide the fact that the procedure call from client to server is not local.

RPC Semantics

- At most once
- At least once
- Once
- Idempotent

Robustness

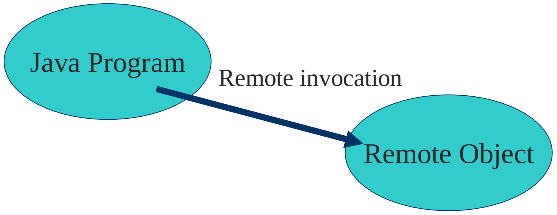
- failure detection -- keep-alive, handshake, are-you-up, heartbeat
- reconfiguration -- changing name resolution data base to map redundant resources for resource requests

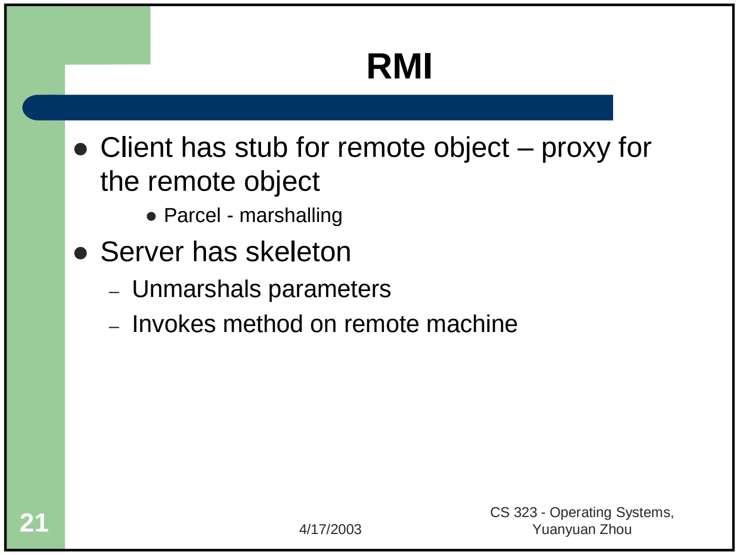
Robustness

- recovery from failure -- keep trying until link back up or explicit notification on recovery
- replicating data and processes so that if one system fails, another system has the results

Java Remote Method Invocation

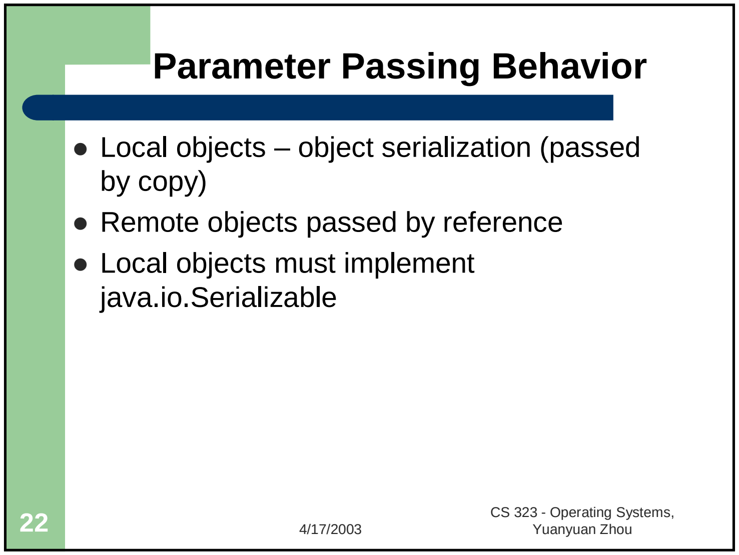
- RPC with objects
- Includes objects as parameters





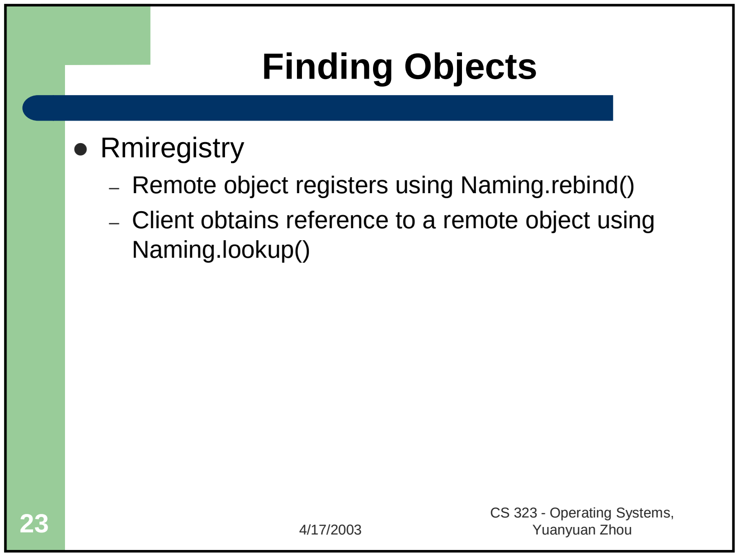
RMI

- Client has stub for remote object – proxy for the remote object
    - Parcel - marshalling
- Server has skeleton
    - Unmarshals parameters
    - Invokes method on remote machine



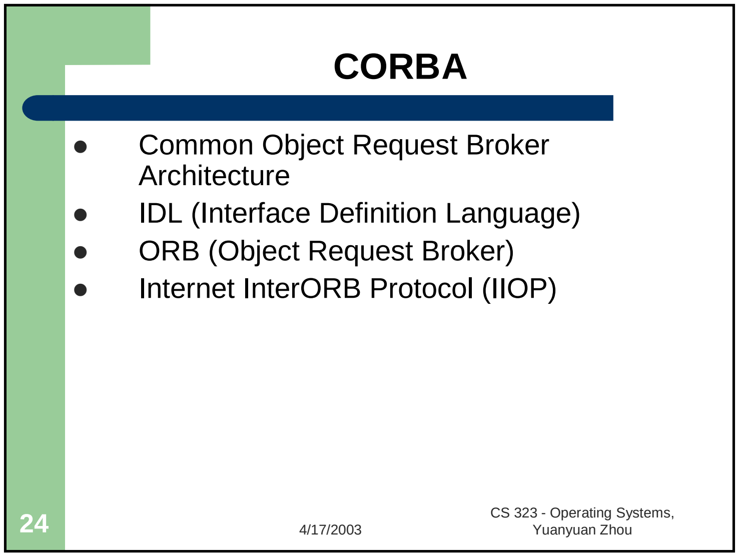
Parameter Passing Behavior

- Local objects – object serialization (passed by copy)
- Remote objects passed by reference
- Local objects must implement java.io.Serializable



Finding Objects

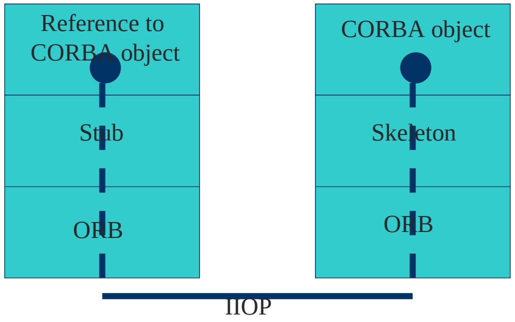
- Rmiregistry
    - Remote object registers using Naming.rebind()
    - Client obtains reference to a remote object using Naming.lookup()



CORBA

- Common Object Request Broker Architecture
- IDL (Interface Definition Language)
- ORB (Object Request Broker)
- Internet InterORB Protocol (IIOP)

CORBA MODEL



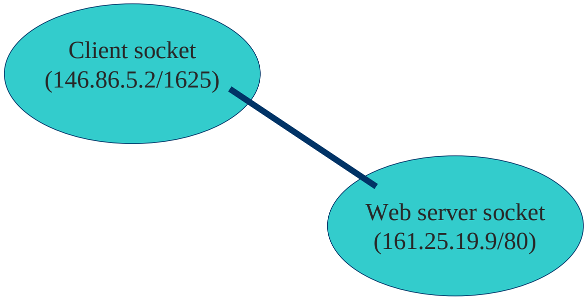
Sockets

- Communication endpoint
- (IP address, Port number)
- Client-server – server listens to a port
- Telnet Port 23, ftp port 21, web server port 80

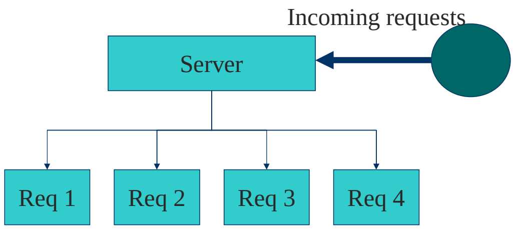
Ports

- Ports < 1024, standard
- Ports > 1024, user created
- All connections unique
- (161.25.19.8:20)

Socket Communication



Servers and Threads



Create a thread for each request to avoid blocking in a single thread

Java Sockets

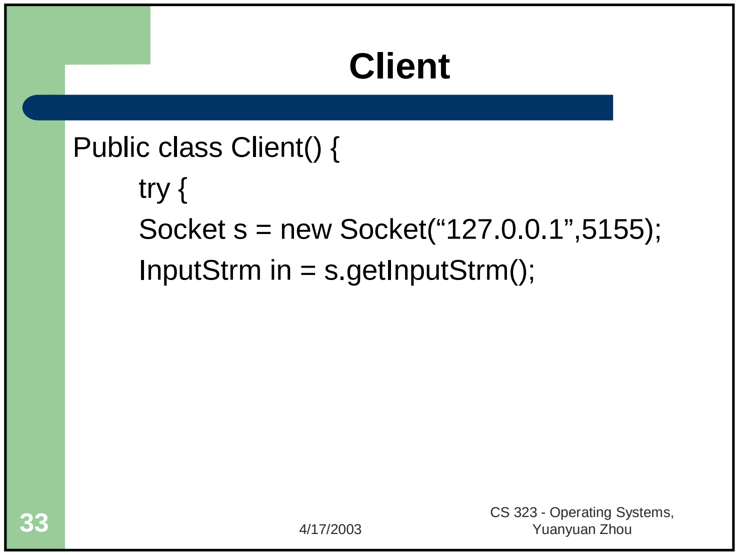
- Connection-oriented (TCP) Sockets
- Connectionless (UDP) Sockets
- Multicast (UDP) Socket

Time-of-Day Server

```
Try {
        s = new ServerSocket(5155);
        }
```

Time-of-Day Server

```
Try {
        while (true) {
                client = s.accept();
                c = new Connection(client);
                c.start();
        } }
```



Client

```
Public class Client() {
        try {
        Socket s = new Socket(“127.0.0.1”,5155);
        InputStrm in = s.getInputStrm();
```