CS 640: Computer Networking

Yu-Chi Lai

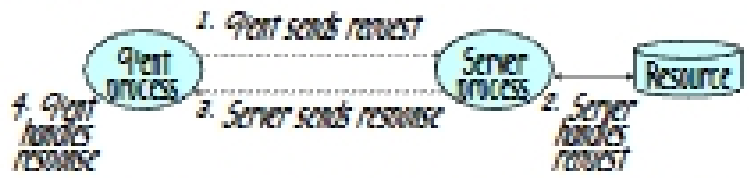
Lecture 3
Network Programming

Topics

- Client-server model
- Sockets interface
- Socket primitives
- Example code for echoclient and echoserver
- Debugging With GDB
- Programming Assignment 1 (MNS)

Client/server model

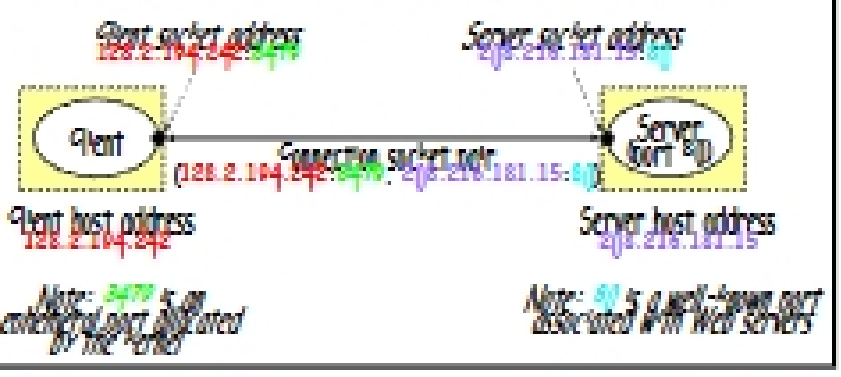
- Client asks (*request*) – server provides (*response*)
- Typically: single server – multiple clients
- The server does not need to know *anything* about the client
  - even that it exists
- The client should always know *something* about the server
  - at least where it is located



Note: clients and servers are processes running on hosts (can be the same or different hosts).

Internet Connections (TCP/IP)

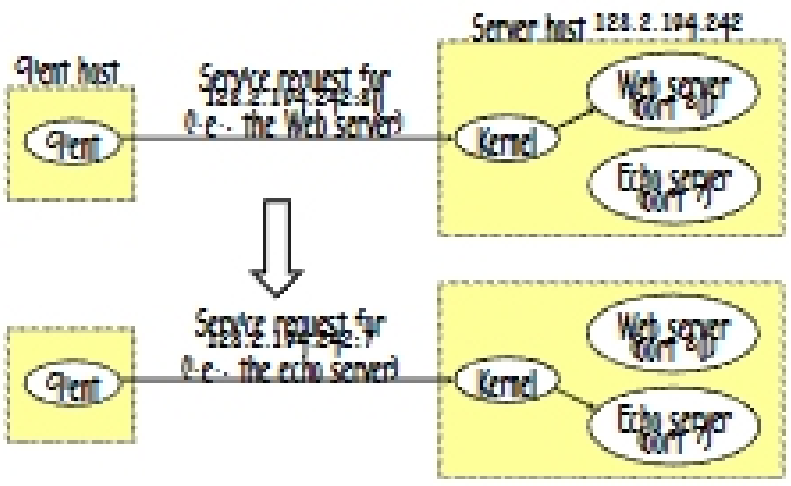
- Address the machine on the network
  - By IP address
- Address the process
  - By the "port"-number
- The pair of *IP-address + port* - makes up a *"socket-address"*



Clients

- Examples of client programs
  - Web browsers, `ftp`, `telnet`, `ssh`
- How does a client find the server?
  - The IP address in the server socket address identifies the host
  - The (well-known) port in the server socket address identifies the service, and thus implicitly identifies the server process that performs that service.
  - Examples of well known ports
    - Port 7: Echo server
    - Port 23: Telnet server
    - Port 25: Mail server
    - Port 80: Web server

Using Ports to Identify Services



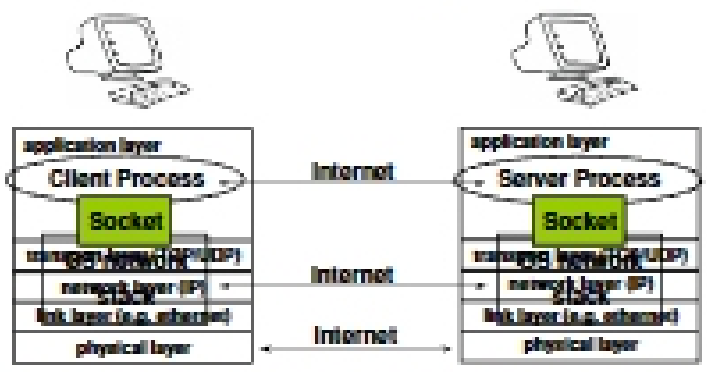
Servers

- Servers are long-running processes (daemons).
  - Created at boot-time (typically) by the init process (process 1)
  - Run continuously until the machine is turned off.
- Each server waits for requests to arrive on a well-known port associated with a particular service.
  - Port 7: echo server
  - Port 23: telnet server
  - Port 25: mail server
  - Port 80: HTTP server

See /etc/services for a comprehensive list of the services available on a Linux machine.

- Other applications should choose between 1024 and 65535

Sockets as means for inter-process communication (IPC)



The interface that the OS provides to its networking subsystem

Sockets

- What is a socket?
  - To the kernel, a socket is an endpoint of communication.
  - To an application, a socket is a file descriptor that lets the application read/write from/to the network.
    - Remember: All Unix I/O devices, including networks, are modeled as files.
- Clients and servers communicate with each by reading from and writing to socket descriptors.
- The main distinction between regular file I/O and socket I/O is how the application "opens" the socket descriptors.