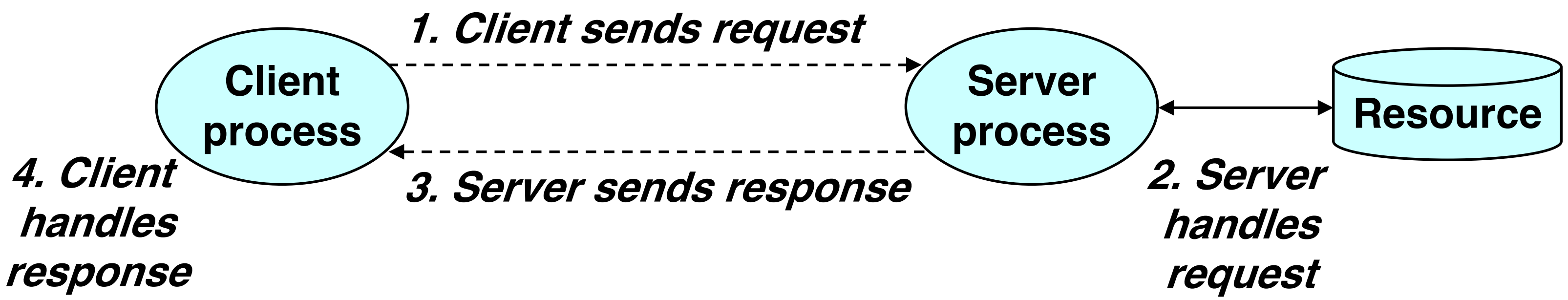
# Client Server Programming

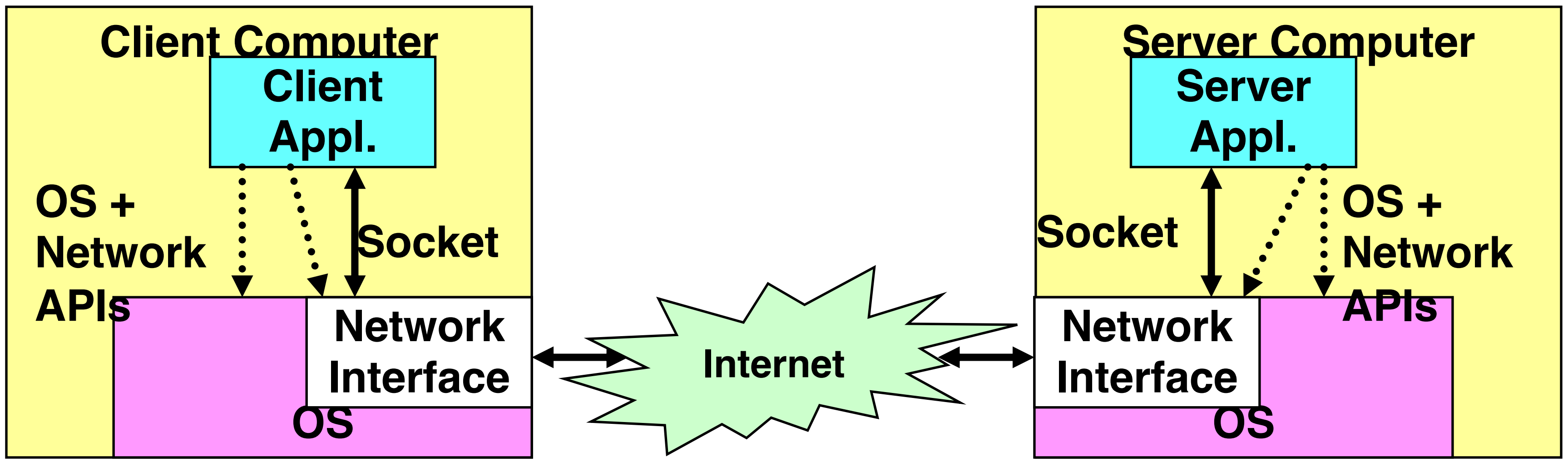
ECE 50863 – Computer Network Systems

# A Client-Server Transaction

- Every network application is based on the client-server model:
  - A *server* process and one or more *client* processes
  - Server manages some *resource*.
  - Server provides *service* by manipulating resource for clients.



# Network Applications

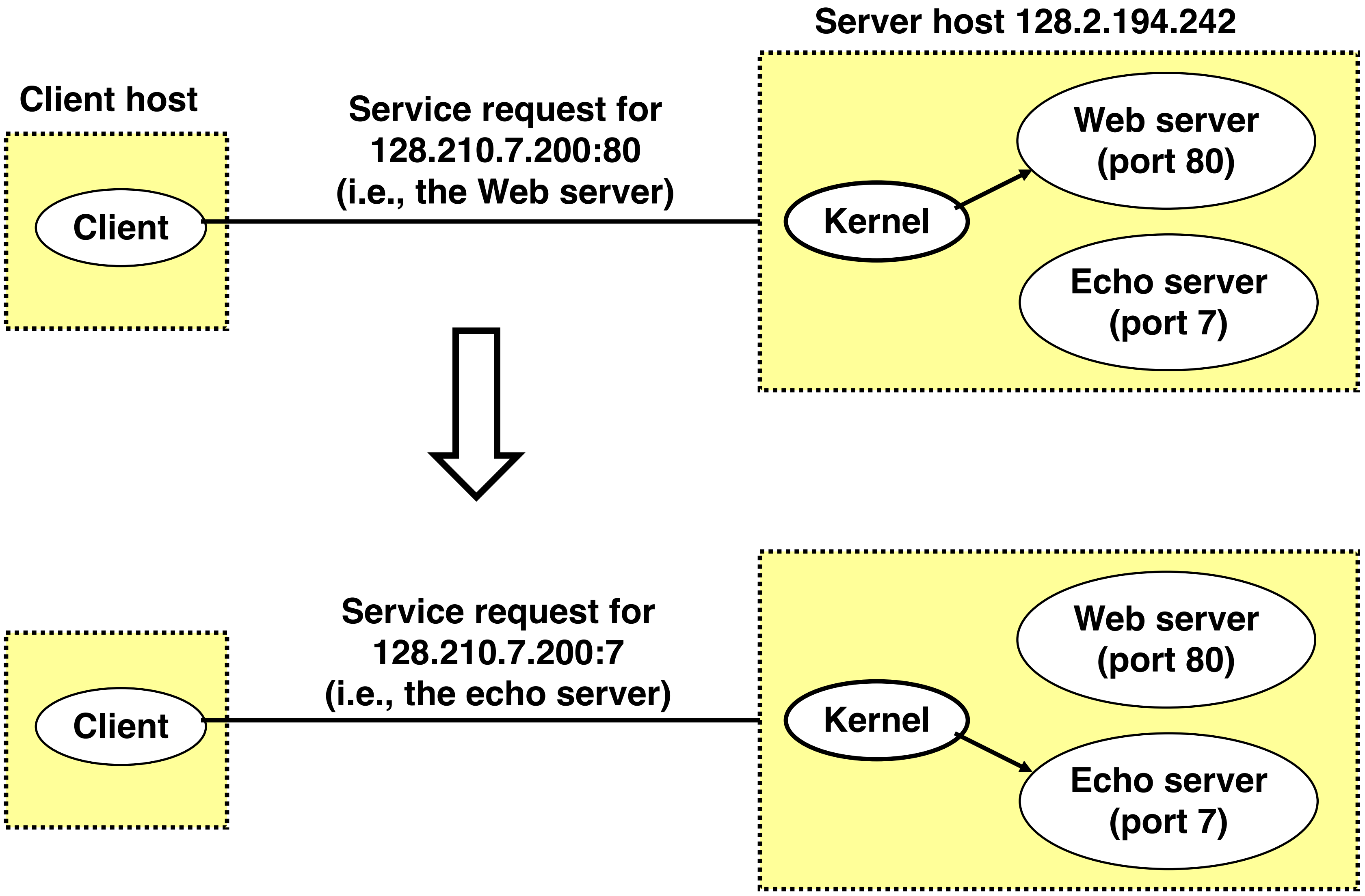


- Access to Network via Program Interface
  - Sockets make network I/O look like files
  - Call system functions to control and communicate
  - Network code handles issues of routing, reliability, ordering, etc.

## How does a client find a server?

- Two pieces of information:
  - The IP address of the server
    - E.g., 128.210.7.200
    - 32 bits long, how a host is identified by all other hosts.
  - The server port.

# Using Ports to Identify Services



# Multiplexing using Ports

- Well-known Vs. Ephemeral Ports
  - 0-1023  “well-known” port numbers
  - Typically used by servers of well-known apps
  - Higher numbers: “ephemeral”

# Servers

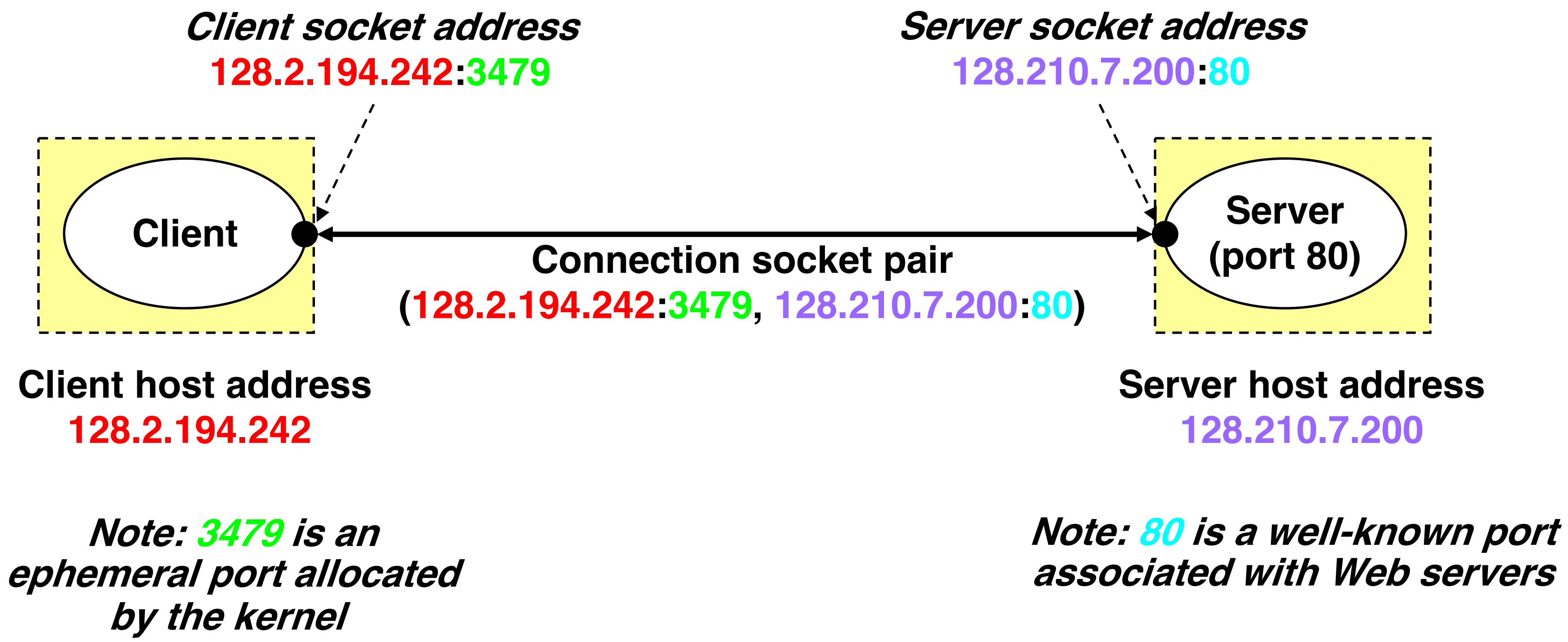
- Servers are long-running processes (daemons).
  - Created at boot-time (typically) by the init process
  - Run continuously until the machine is turned off.
- Each server waits for requests to arrive on a well-known port associated with a particular service.
  - Port 7:   echo server
  - Port 23: telnet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

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

- A machine that runs a server process is also often referred to as a "server."

# Internet Connections

- Clients and servers communicate by sending streams of bytes over *connections*.
- Connections are point-to-point, full-duplex (2-way communication), and reliable.



# Sockets

- What is a socket?
  - To the kernel: endpoint of communication.
  - To an application: a file descriptor that lets the application read/write from/to the network.
- 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.

## TCP Vs. UDP sockets

- Programmer decides which transport protocol to use.
- Specified using a parameter to the socket call
- TCP:
  - "Byte Stream" Abstraction
  - Reliable and in-order
- UDP:
  - Does not guarantee reliability or in-order delivery