UNIX Systems Programming

Client/Server Communication

(Robbins & Robbins, chp.12)

Dr. Kivanç Dinçer

CENG-332 Lecture Notes

Spring 2000

Client-Server (C/S) Model

- Processes called servers provide services to clients:
  - Examples include many network services
    - mail, file transfer (ftp,) remote login (telnet,) access to remote file systems (NFS)
- Low-level protocol:
  - programmer is aware of the server and its location and must explicitly name the particular server to be accessed.

Low-Level Communication Protocols

- There are two classes of low-level comm. protocols supporting the C/S model:
  - connection-oriented:
    - a server waits for a connection request from a client
    - once connection is established, communication takes place using handles (file descriptors),
      - i.e., server address is not included in client messages.
  - connectionless:
    - client sends a single message to the server
    - server performs the service and returns a reply.

C/S Strategies

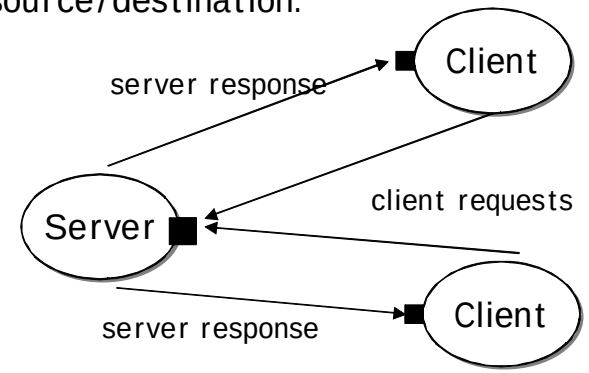
- When the server starts up, it opens its well-known FIFO (or socket connection) and waits for client requests:
  - When a client needs a service, it opens the FIFO (or a socket connection) and writes its request.
  - The server than performs the service.

What if there is more than one client?

- a connection for sending the client's PID or (host name +port no) must be established so that the requests from different clients can be distinguished
- A single port will not suffice: Clients need responses!

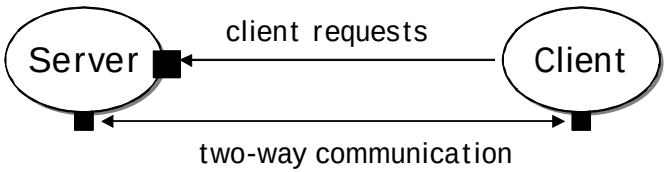
Using connectionless protocol

- The server has a well-known port for client requests, but client-specific ports for responses:
  - recvfrom() and sendto() identifies the source/destination.



Using connection-oriented protocol

- The server has a well-known port for client requests, but client-specific ports for responses:
  - A hand-off mechanism is used: The server provides a private, two-way communication channel.



Strategies for handling a request

1- Serial Server strategy

- can service only one request at a time
  - not suitable for handling long-lived requests

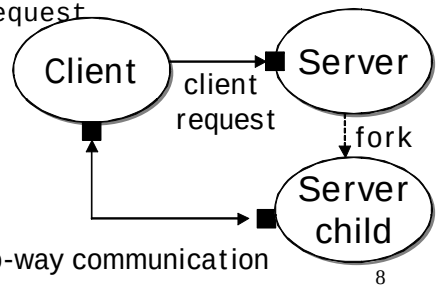
```
for ( ; ; ) {
  listen for client request
  create private two-way communication channel
  while (no error on communication channel)
     read client request
     handle request and respond to client
   close communication channel
}
```


Strategies for handling a request

2- Parent-server strategy

- the server forks a child to handle the actual service to the client while the server resumes listening for additional requests
  - ideal for file transfers

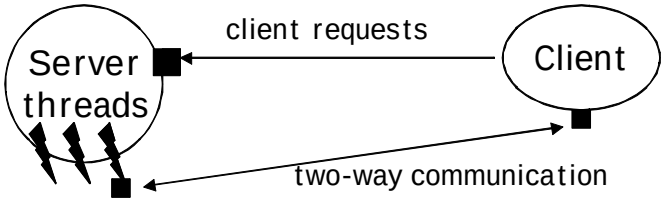
```
for ( ; ; ) {
  listen for client request
  create private two-way communication channel
  fork a child to handle the request
  close communication channel
  clean up zombies
}
```



Strategies for handling a request

3- Threaded-server strategy

- low-overhead alternative to parent-server strategy
- the server opens a thread in its address space to handle the actual service to the client while the server resumes listening for additional requests
  - ideal for small or I/O intensive requests
  - be careful about the possible interference among threads



Network Communication

- The International Standards Organization (ISO) has a standard for network design called the Open Systems Interconnection (OSI) reference model.
  - There are seven protocol layers
    - Each layer consists of a set of functions to handle a particular aspect of the network communication
    - Functions in a layer communicate only with the layers directly above and below.

ISO/OSI Model

- In a peer-to-peer comm. with layered protocols,
  - Logical view: each layer on one host appears to be communicating w/ the same level on the other host:
    - this view simplifies and isolates the implementation of a layer's functions.
  - Actually: each layer performs its function and passes the information to the layer below or above
    - Eventually, the information flows on the physical network to the other host and it is passed up through successive layers on the other host.

Physical Layer & Data Link Layer

- concerned w/pt-to-pt xmission of data
- Ethernet is the common low-cost impl. of these layers
  - Other alternatives: ISDN, ATM, and FDDI.
  - Each host on the network has a hardware Ethernet adapter that is connected to the communication link (coaxical cable/twisted pair wire)
  - The host is identified by a unique 6-byte Ethernet address (MAC) that is hard-wired into the adapter hardware.

Network Layer

- handles network addressing and routing through bridges and routers interconnecting networks.
  - most common protocol on UNIX is IP (Internet Protocol)
    - Every host has one or more 4-byte IP addresses.

Transport Layer

- handles end-to-end communication between hosts
  - Two main protocols: TCP (Transmission Control Protocol) and UDP (User Datagram Protocol)
    - UNIX networks use both
    - UDP: connectionless w/o any guarantee for delivery
      - tftp – trivial file transper protocol is implemented using UDP
    - TCP: reliable, connection oriented
      - ftp –file transper protocol is implemented using TCP.

Ports

- More than one user at a time may be using TCP or UDP between a given pair of machines. To distinguish between the various processes that might be communicating, TCP and UDP use 16-bit integers called ports.
  - Some of these port numbers have been permanently assigned to specific applications and are called well-known addresses
    - ftp: 21, telnet: 23, tftp: 69, finger: 79, http: 80
    - User processes should choose port numbers above 7,000 so as not to interfere with system services and X services (ports upto 1024 are reserved for O/S)

Session layer

- contains interfaces to the transport layer.
- The presentation layer and application layer consist of general utilities and application programs.
  - Presentation layer may handle compression or encryption.

Presentation and Application Layer

- consist of general utilities and application programs.
  - Presentation layer may handle compression or encryption.