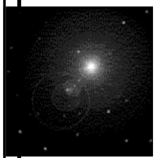


A Course on Internetworking & Network-based Applications

CS 6/75995 Internet-based

Applications & Systems Design

Kent State University

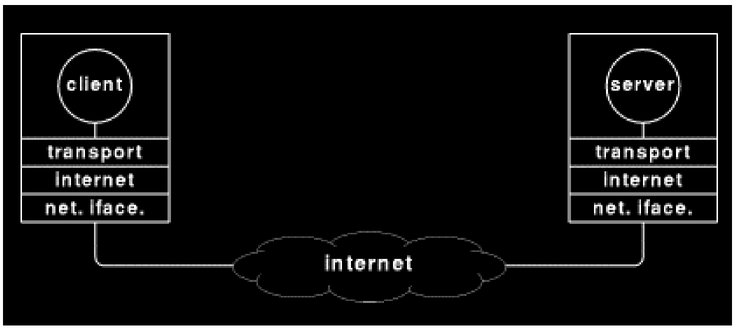
Dept. of Math & Computer Science

LECT-4

Application Architecture

3

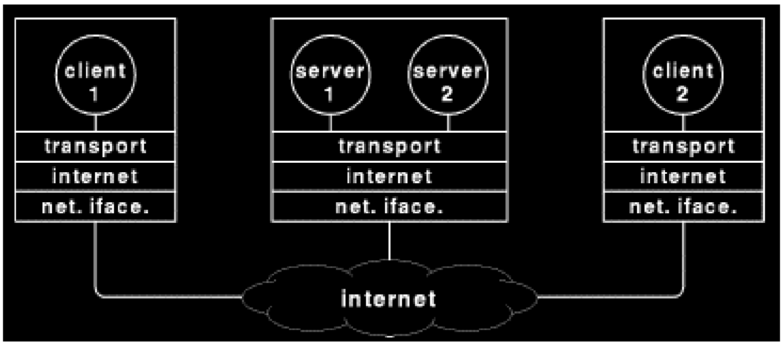
Client Server Interaction



INTERNET APPLICATION & SYSTEM DESIGN

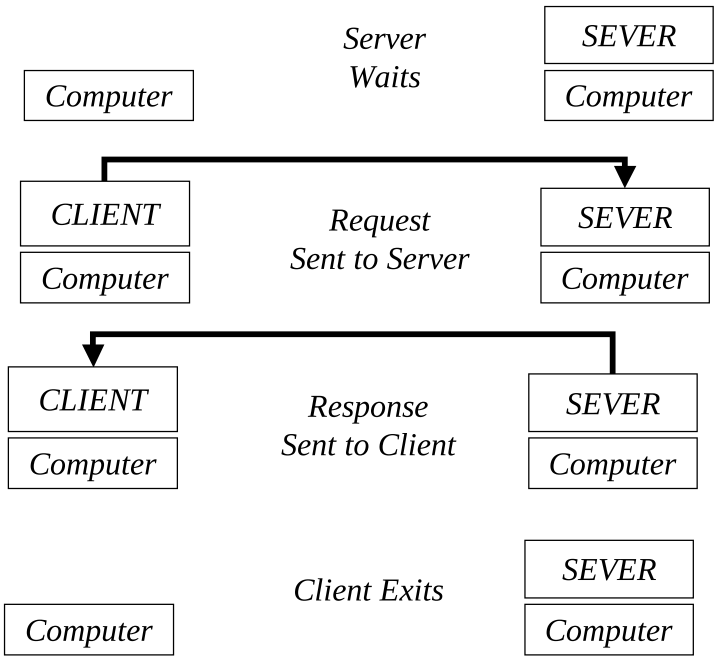
LECT-04, S-4
IAD99S, javed@kent.edu
Javed I. Khan@1999

Multiple Services on One Computer



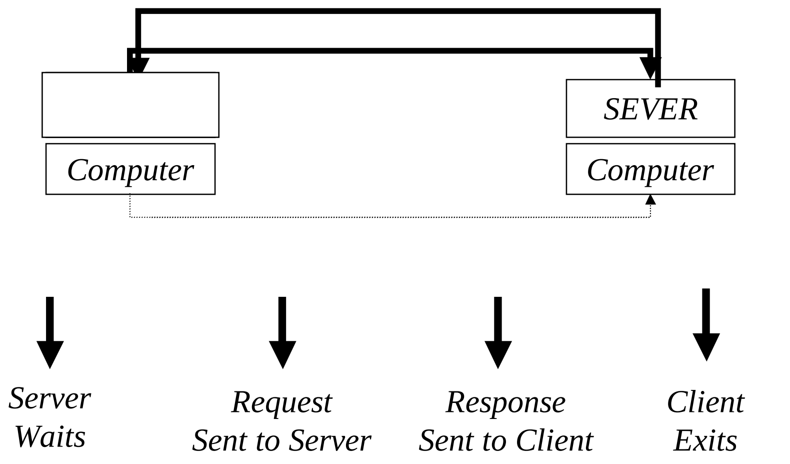
INTERNET
APPLICATION &
SYSTEM DESIGN

LECT-04, S-5
IAD99S, javed@kent.edu
Javed I. Khan@1999



INTERNET
APPLICATION &
SYSTEM DESIGN

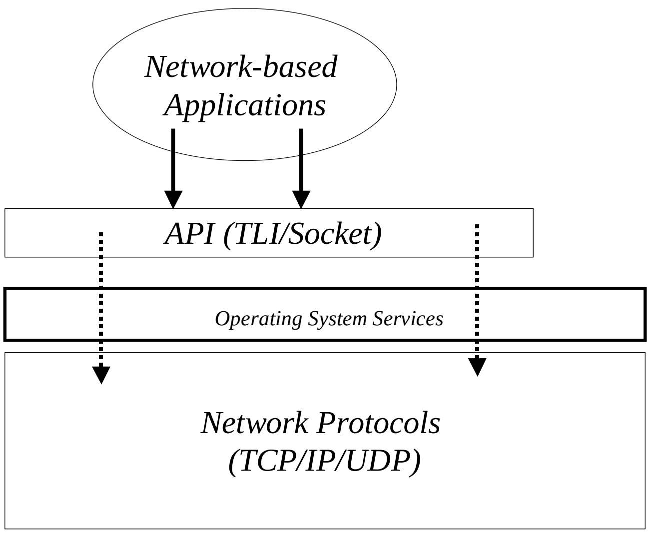
LECT-04, S-6
IAD99S, javed@kent.edu
Javed I. Khan@1999



INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-7
IAD99S, javed@kent.edu
Javed I. Khan@1999

Application Platform



INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-8
IAD99S, javed@kent.edu
Javed I. Khan@1999

API to Network

9

Network APIs

- Transport Layer Interface (TLI) was developed in mid 1980s by AT&T with release 3 of system V UNIX. Later it became a part of Sun systems. Now this is available everywhere.
- Socket API is the original API developed by Berkeley UNIX group in late 70's and early 80s. Available on BSD UNIX Systems.
- WinSock is the API version provided for Microsoft Windows.

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-10

IAD99S, javed@kent.edu

What is Socket API?

- Socket Application Program Interface can be used to access a network protocol stack from any programming language.
- Socket API has been inspired by Unix *open-read-write-close* file access paradigm (original is Multics).
- However, accessing network is substantially more complex than file access.

Sockets & Files

- An integer called file descriptor is returned, when a file is opened.
- In the same way a socket descriptor is returned when a socket is opened.
- A file descriptors binds to a file when it is open.
- But, a socket can be created without binding to a specific destination. Applications can choose when to bind
  - Datagram binds each time when it sends, therefore same socket can be used to send to many.
  - TCP binds once, and it remains, thus avoids repeated binding.

INTERNET
APPLICATION &
SYSTEM DESIGN

Socket Creation & Closing

- **descriptor=socket(pfamily, type, protocol)**
  - **pfamily** =PF_INET| PF_APPLETALK | PF_UNIX| PF_PUP
  - type=SOCK_STREAM, SOCK_DGRAM, SOCK_RAW, etc.
  - protocol=subtype of protocol family if any.
- Unix uses fork() and execv() to create and spawn new program. Child always inherits all parent sockets. UNIX maintains a count of owners.
- A process can close a socket by **close(socket)**

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-13
IAD99S, javed@kent.edu
Javed I. Khan@1999

Specifying a Local Address

- Initially a socket is created without any association to a local or remote address.
- For TCP/IP it means no protocol port number.
- Some application may not care (clients generally). Some do (all servers). The call:
- **bind(socket, localaddress, addrlen)**
  - socket is the socket descriptor returned by socket()
  - localaddress is a complex structure with several fields, and may vary for protocols.
  - For TCP/IP it contains both the port number and the IP address of the host is in it.
  - addrlen is the length of the address.

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-14
IAD99S, javed@kent.edu
Javed I. Khan@1999

Address Structures

- Barkley code defines a generic sockaddr structure to represent address of a connection end-point.

```
struct sockaddr {
    u_char sa_len;              /*total length of the address*/
    u_char sa_family;           /*family of address*/
    char sa_data[14];           /*the address itself*/
};
```

- Example: TCP/IP defines its own exact sockaddr_in:

```
struct sockaddr_in {
    u_char sin_len;             /*same as sa_len*/
    u_char sin_family;          /*same as sa_family*/
    u_short_sin_port;           /*protocol port number*/
    struct in_addr sin_addr     /*6B IP adress of the host*/
    char sin_zero[8];           /*not used 6+8=14*/
};
```

- Generally server calls bind to specify a server port number at which it will accept connection.
- A server on a multi-homed host can write down INADDR_ANY instead of the IP address to say it will accept the connection in any of the computers IP addresses.

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-15

IAD99S, javed@kent.edu
Javed I. Khan@1999

Listening for Connection

- listen allows a server to wait for a connection request from a remote cleint.
- Listen(socket, queuelength)
  - socket is the descriptor that has been created and is bound to a local address.
  - queuelength specifies how many request can wait while server is busy with one.
  - OS maintains a separate request queue for each socket. It the queue is full, OS refuses new requests.

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-16

IAD99S, javed@kent.edu
Javed I. Khan@1999

Waiting for Accepting a Connection

- After a server executes *socket-bind-listen*, it can go to sleep by calling accept. The Operating System will wake up the server when there is a request in the request queue.
- newsock=accept(socket, &addr, &addrlen)
  - socket is on which it was waiting,
  - addr is a pointer, in which the address of the client will be returned by the OS. Addrlen is length of this address.
  - newsock is a new socket created by system which has its destination pre-connected to the client.
- The server can keep on communicating with the requesting client with the newsocket, and close it when done. Meanwhile, the original socket remains intact to accept request from other clients.

Connecting to a Destination Address

- Initially a socket is created also without any destination address. An client application program must call connect to establish connection.

- **connect(socket, destaddr, destaddrlen)**

Sending Data

- **write(socket, buffer, bytelength)**
- **writev(socket, iovector, vectorlen)**
  - iovector is an array of addresses and their lengths. The system gathers all the data.
- **send(socket, message, length, flags)**
- **sendto(socket, message, length, flags, daddr, daddrlen)**
- **sendmsg(socket,messagestruct,flags)**

Receiving Data

- **read(descriptor, buffer, length)**
- **readv(descriptor, iovector, vectorlen)**
- **recv(socket, buffer, length, flags)**
- **recvfrom(socket,buffer,length,flags,fromaddr,addrlen)**
- **recvmsg(socket,messagestruct,flags)**

INTERNET
APPLICATION &
SYSTEM DESIGN

Obtaining Local & Peer Address

- **getpeername(socket, &destaddr,&addlen)**
- **getsockname(socket, &localaddr,&addlen)**

Obtaining & Setting Socket Options

- **Getsockopt(socket,level,optionid,&optionval,&length)**
- **Setsockopt(socket,level,optionid,optionval,length)**
- Example options are timeout parameters, allocated buffer space etc.

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-21
IAD99S, javed@kent.edu
Javed I. Khan@1999

Handling Multiple Request

- **nready=select(ndesc,indesc,outdesc,execdesc,timeout)**
  - a call to select will allow server to wait till one of the descriptor is ready.
  - ndesc=number of descriptors. System checks descriptors from 0 to ndesc-1. There are 3 bit masks to wait on a selected subset.
  - Timeout says how long to wait. 0 means wait indefinitely.

Obtaining & Setting Hostnames

- **gethostname(name,length)**
- **sethostname(name,length)**

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-22
IAD99S, javed@kent.edu
Javed I. Khan@1999

Obtaining & Setting Domain Names

- **setdomainname(name,length)**
- **getdomainname(name, length)**

Network Byte Order

- **Localshort=ntohs(netshort)**
- **Locallong=ntohl(netlong)**
- **Netshort=htons(localshort)**
- **Netlong=htonl(locallong)**

INTERNET
APPLICATION &
SYSTEM DESIGN

IP Address Manipulation

String to 32 bit network byteordered address

- **Address=inet_addr(string)**
- **Address=inet_network(string)**
  - address is a 32 bit IP address in network byte order.
  - string is an ASCII stirng with IP in dotted decimal notation
  - inet_network returns o for host part.
- **Str=inet_ntoa(internetaddr)**
- **Internetaddr=inet_makeaddr(net,local)**
- **Net=inet_netof(internetaddr)**
- **Local=inet_lnaof(internetaddr)**

INTERNET
APPLICATION &
SYSTEM DESIGN

Accessing Domain Name System

- Each computer now also have a symbolic domain name.
  - Such as www.kent.edu or shimana.facnet.mcs.kent.edu
- A set of designated computers (knows as DNS servers) scattered across the internet maintains the mapping of DNS to the actual IP.
- Translation from domain name to IP address or the opposite, requires communication with these servers.

Accessing Domain Name System

- **res_init()** — *Initialize DNS comm.*
- **res_mkquery(op,dname,class,type,**
  data, datalen,newrr, buffer,buflen) — *Form Query.*
- **res-send(buffer,buflen,answer,anslen)** — *Send Query.*

- **dn_expand(msg,eom,compressed,full,fullen)**
- **dn_comp(full,compressed,cmprlen,prevptr,lastptr)**

Conversion between ASCII name and compressed domain name format.

- **ptr=gethostbyname(namestr)**
- **ptr=gethostbyaddr(add,len,type)**

Takes a domain name and returns a structure with information about the domain.

Obtaining Information about Network Services

INTERNET APPLICATION & SYSTEM DESIGN

- WHOIS a special server service, which allows a client in one machine to obtain information about user who has account in server machine. It runs on Port 43.
  - **Ptr=getservebyrname(name,proto)**
- Name is the address of a desired service, and proto is usually TCP or UDP. It returns a structure which contains name of the service, a list of aliases, protocol identifier for this service, and an integer protocol port number.
  - **Ptr=getservbyport(port,proto)**

Obtaining information about Network & Protocol

INTERNET APPLICATION & SYSTEM DESIGN

- **Ptr=getnetbyname(namestr)**
- **Ptr=getnetbyaddr(netaddr, addrtype)**

Namestr is the name of the network in ASCII, ptr is a data structure which contain 32 bit IP address and other information about the net.

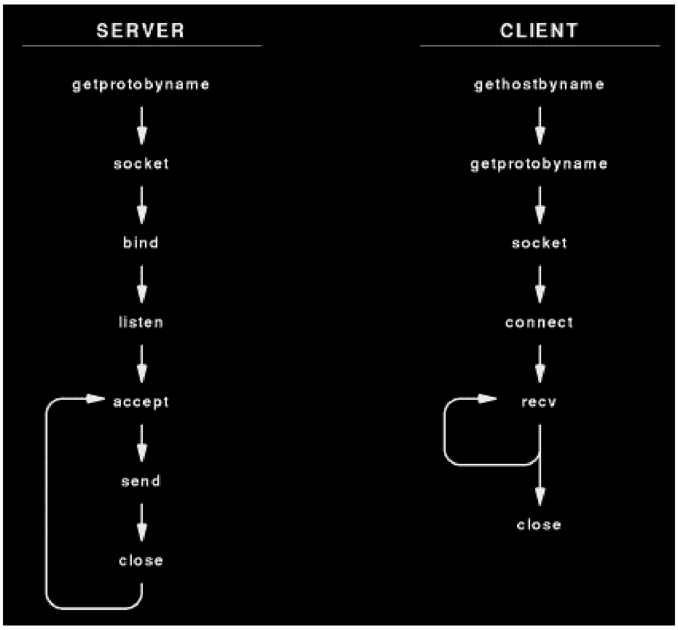
- Each protocol has official name, number and registered aliases. These routines can be used to obtain complete information from name or port number of it.
- **Ptr=getprotobyname(name)**
- **Ptr=getprotobynumber(number)**

An Example Service

- A client connect to a server, and waits for output.
- Server returns the count of the times it has been contacted by any client.
- Upon receiving the data the client prints it to screen.

- Command line arguments:
  - client <hostname> <portname>
  - server <portnumber>
  - hostname and portnumbers are optional.
  - Default host is localhost
  - Default port is 5193.
- Output on client machine:
  - This server has been contacted 10 times.

INTERNET APPLICATION & SYSTEM DESIGN



INTERNET APPLICATION & SYSTEM DESIGN

INTERNET
APPLICATION &
SYSTEM DESIGN

server.c

Click Here

http://www.animasters.com/menu/vrml/estuary/estuaryintro.html

LECT-04, S-31
IAD99S, javed@kent.edu
Javed I. Khan@1999

INTERNET
ENGINEERING
INTERNET
APPLICATION &
SYSTEM DESIGN

client.c

Click Here

http://www.animasters.com/menu/vrml/estuary/estuaryintro.html

LECT-04, S-32
IAD99S, javed@kent.edu
Javed I. Khan@1999

OS Mechanisms Posix 1003.1c

33

Linux Threads

- Dynamic Creation
  - Pthread_create
- Concurrent Execution
  - All threads appear to execute at the same time.
- Preemption
  - Sched_yield to voluntary release allocated time.
- Private Local Variables
  - Each thread has private stack.
- Shared Globals
  - All threads within the same process share global variables.
- Shared File Descriptions
  - All threads within the same process shares fds.

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-34
IAD99S, javed@kent.edu

Synchronization

- Mutex
  - Generally a separate mutex variable is used for each variables to be protected.
  - pthread_mutext_init
  - pthread_mutext_lock
  - pthread_mutext_unlock.
- Semaphores
  - Generally one semaphore for each resource with count N.
  - Sem_init
  - sem_wait
  - sem_post.
- Condition Variables
  - A tool for applications in need of busy wait

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-35

IAD99S, javed@kent.edu
Javed I. Khan@1999

Concurrency and Advanced Network-based Applications

36

Issues in Client Design-1

- Choosing A Local Port Number
  - A client can choose any port number.
  - No conflict with other port numbers in use.
  - Not a well known server port number.
  - If a process does not explicitly call bind, Connect or listen, which ever is called kernel automatically picks a valid ephemeral port and source IP.
- Choosing A Local IP address
  - In a multi-homed computer applications may not know which IP adapter is in use.
  - While binding, setting IP address =0 (INADDR_ANY) will allow the kernel to pick the correct IP address.

Once a value is assigned by the KERNEL how can application find out the assigned address?

getsockname(socket, &localaddr,&addlen)

Issues in Client Design-2

- Reading from a Stream
  - Only one write operation may require multiple read at the other end.
  - The flag can be 0 or AND of few values such as:
  - MSG_PEEK: look at the data that is available to read, but witout having system discard it from system buffer.
  - MSG_WAITALL: do not return until the specified amount of data is received.
- /* Repeatedly read data from socket and write to user's screen. */

```
n = recv(sd, buf, sizeof(buf), 0);
while (n > 0) {
    write(1,buf,n);
    n = recv(sd, buf, sizeof(buf), 0);
}
```

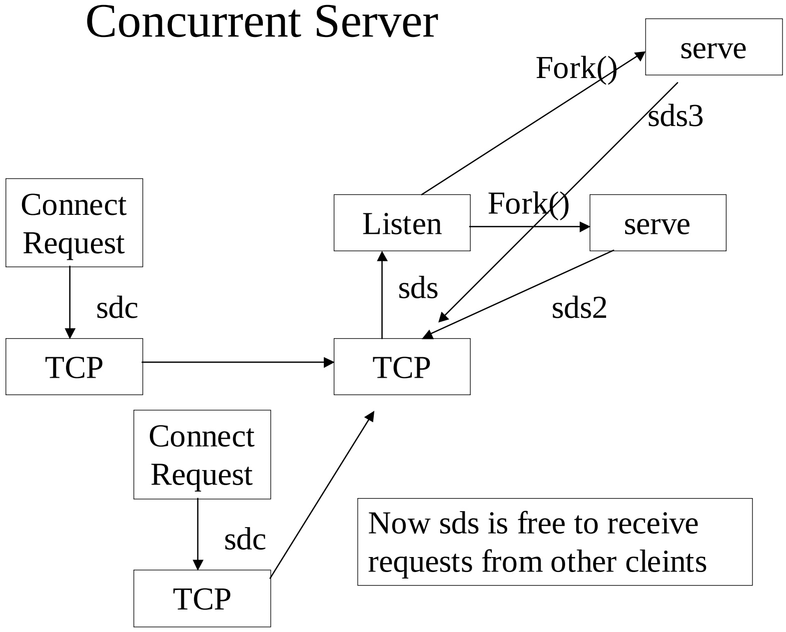
INTERNET APPLICATION & SYSTEM DESIGN

Issues in Server Design-1

- Binding to a Well Known Port
  - If the service have to be available to a wide audience the server must bind to a well known port.
- Setting up the IP address
  - like clients a server may leave it upto the kernel to set the IP address to avoid confusion in multi-homed hosts.

INTERNET APPLICATION & SYSTEM DESIGN

Concurrent Server



INTERNET APPLICATION & SYSTEM DESIGN

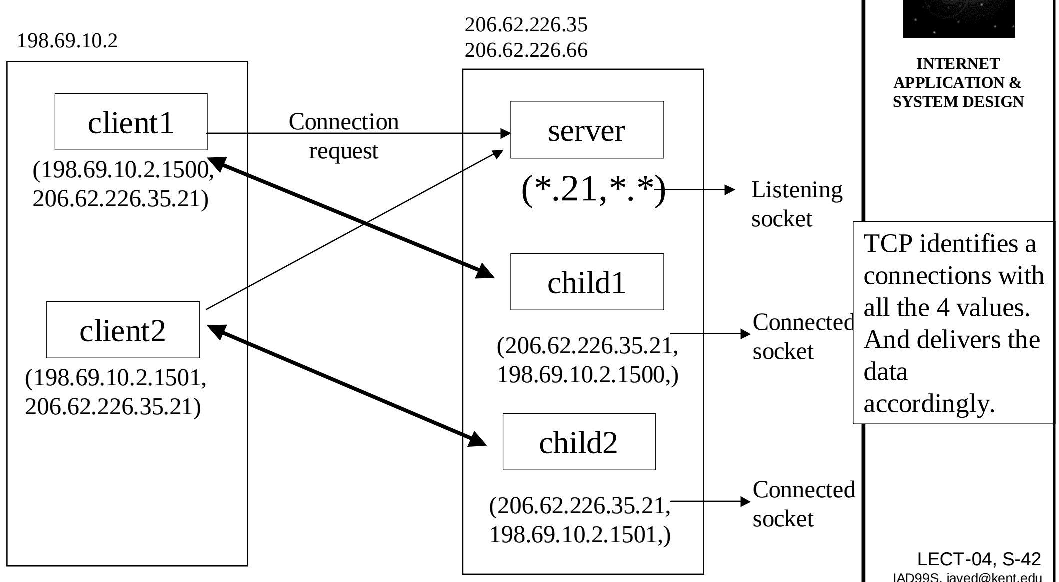
Concurrent server.c

Click Here

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-41
IAD99S, javed@kent.edu
Javed I. Khan@1999

TCP Port Numbers and Concurrent Servers



INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-42
IAD99S, javed@kent.edu
Javed I. Khan@1999

Connection Oriented & Connectionless Communication

- UDP Client
- UDP Server

INTERNET APPLICATION & SYSTEM DESIGN

Issues in Server Design-2

- Concurrent vs. Iterative server
  - simplicity, response time
  - real vs. apparent concurrency
- Connection-Oriented vs. Connectionless Access
  - reliability, persistence, data volume, flow-control

Iterative connectionless	*Iterative connectionoriented*
Concurrent connectionless	*Concurrent connectionoriented*

INTERNET APPLICATION & SYSTEM DESIGN

Concurrency Management

Perspectives:
- User perspective: response time
- System perspective: impact on resources.

Issues:
- How can programmer know whether concurrency is warranted?
- How to determine which design is optimal?
- How can a programmer estimate demand or service time?

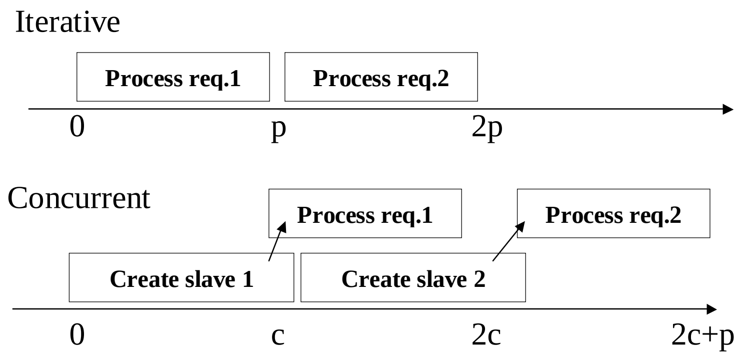
Concepts:
- Level of Concurrency
- Demand-Driven Concurrency

INTERNET APPLICATION & SYSTEM DESIGN

IAD99S, javed@kent.edu

Cost of Concurrency

- Overhead and Delay



Small additional delay can be significant for continuous operation of a server under heavy load.

Quiz: if the rate at which requests arrives exceeds 1/c but is less than 1/p which implementation can handle the load?

INTERNET APPLICATION & SYSTEM DESIGN

IAD99S, javed@kent.edu

Process Preallocation

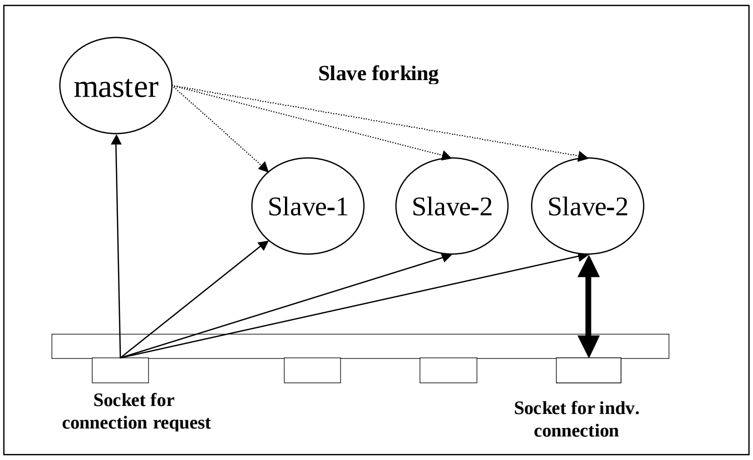
INTERNET APPLICATION & SYSTEM DESIGN

- Demand-driven concurrency can be avoided by limiting the maximum level of concurrency.
- To avoid the run time process creation delay, preallocate service processes.
- Design:
  - Master server creates N slaves at the beginning.
  - Each slave waits/sleeps using OS support.
  - When service request arrives each slave by turn picks it.
  - When done slaves do not exit.
  - Preallocation allows the server process to switch and move to next process faster.

LECT-04, S-47
IAD99S, javed@kent.edu
Javed I. Khan@1999

Preallocation in a Connection Oriented Server

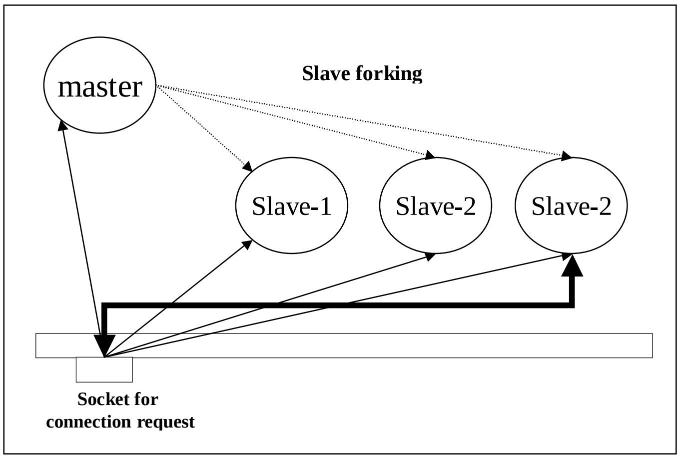
INTERNET APPLICATION & SYSTEM DESIGN



QUIZ: How do the slaves decide that two of them do not jump for serving one client?

LECT-04, S-48
IAD99S, javed@kent.edu
Javed I. Khan@1999

Preallocation in a Connectionless Server



Slaves use the same descriptor to listen for request and send data. Requests arrive in UDP. Descriptors are automatically freed after each communication.

QUIZ: Just like before mutual exclusion requires OS support.

INTERNET APPLICATION & SYSTEM DESIGN

Delayed Process Allocation

- Standby slave approach solve runtime process creation
- but they too costs in terms of OS resource management.
- Iterative server can yield faster service and higher throughput if the service time is small.
- But, how can programmer know the service time?
- Design:
  - the server estimates the processing time dynamically by looking into service parameters (I/O size/bound).
  - server starts processing a new request iteratively.
  - and starts a timer.
  - On time out, it invokes a concurrent slave.

QUIZ: Delayed process creation and preallocation are they are just opposite principles ?

INTERNET APPLICATION & SYSTEM DESIGN

Combined Technique

INTERNET
APPLICATION &
SYSTEM DESIGN

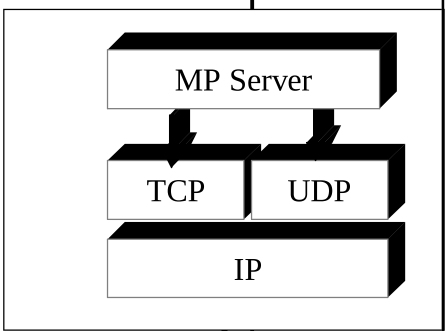
- Design:
  - server starts without any slave.
  - creates a slave only when timer expires.
  - but, once a slave has been created, it does not exit immediately.
  - Slaves can also spawn slaves, if needed.
- How to control concurrency?
  - master specifies limit of concurrency MAX to slaves.
  - Slave exits after a specified period of inactivity.

LECT-04, S-51
IAD99S, javed@kent.edu
Javed I. Khan@1999

Multi Protocol Servers

INTERNET
APPLICATION &
SYSTEM DESIGN

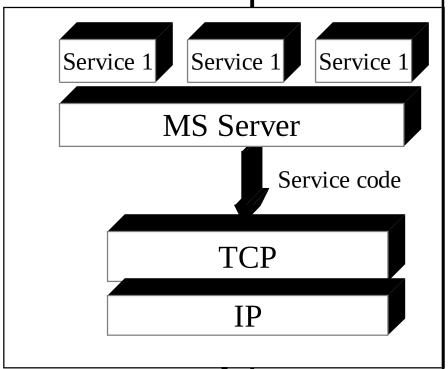
- One server can talk via several underlying transport layer network protocols.
- Example: DAYTIME server maintained in UNIX systems can talk via TCP and UDP both.
- Advantages:
  - easy software maintenance
  - easy debugging
  - service code can be reused..
  - less impact on system resources.
- Disadvantage:
  - less flexibility for network administrator.
- Design:
  - multiple protocol specific listening ports
  - iterative polling.



LECT-04, S-52
IAD99S, javed@kent.edu
Javed I. Khan@1999

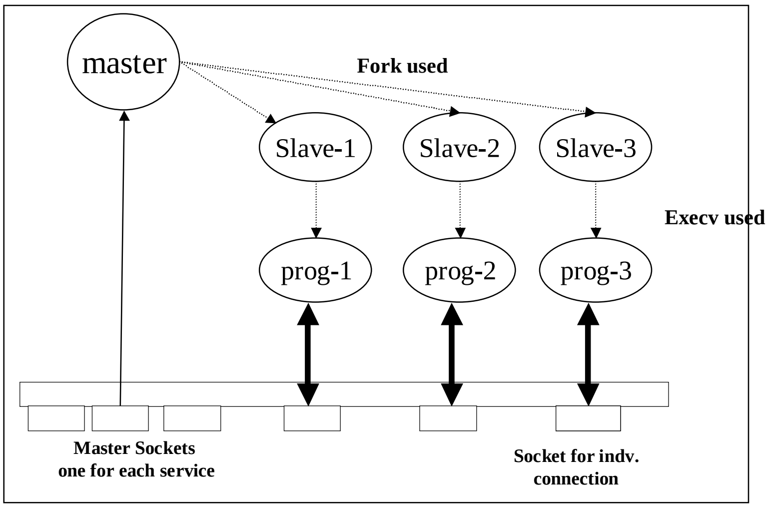
Multi Service Servers

- A typical host may run multiple services. One server can be designed to provide a set of services instead of one.
- Advantages:
  - easy maintenance.
  - communication code can be reused.
  - less impact on system resources.
- Design:
  - requires service descriptor code.
  - server use descriptor code based switch.
  - single process approach
  - concurrent approach
  - super server (UNIX INETD)



INTERNET APPLICATION & SYSTEM DESIGN

Multiservice Server



INTERNET APPLICATION & SYSTEM DESIGN

Advantages of Concurrency to Servers

- Improves observed response time.
- Thus it can increase client throughput.
- It can eliminate potential deadlocks and some denial of service attacks.
- It makes server design modular.
- Concurrent execution can easily be ported on multiprocessor hardware.

QUIZ: Can concurrency be of any use to Clients? Clients do not halt others. Possibility of deadlock is less..

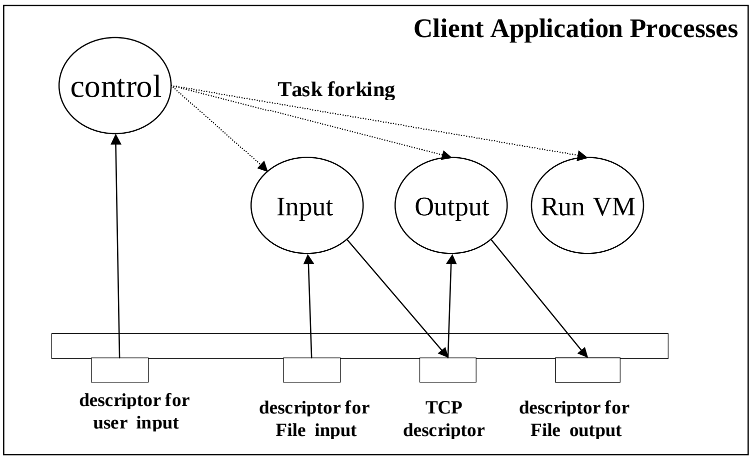
Concurrency in Client Design

INTERNET
APPLICATION &
SYSTEM DESIGN

- Concurrency makes client design simpler and modular too.
- Concurrent client can contact several servers without being held by one.
- Concurrency can allow user to change parameters, inquire about client parameters, or control client processing dynamically.

QUIZ: What if when you are reading a web page and it is taking too long to download, and you want to move on to next hyperlink?

Design of Concurrent Client



- If the server sending data blocks, it can still keep on sending data to server in other direction.

INTERNET APPLICATION & SYSTEM DESIGN

LECT-04, S-57
IAD99S, javed@kent.edu
Javed I. Khan@1999