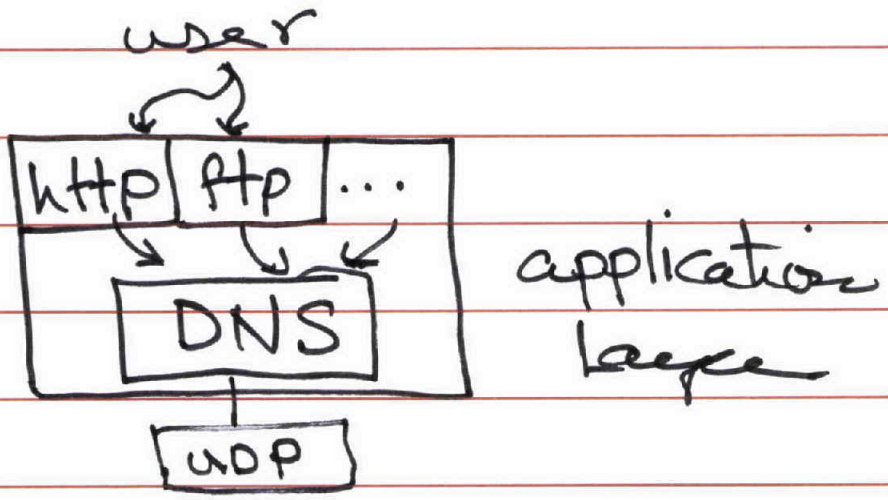
EE450, Fall 2015, Zahid

lecture #9

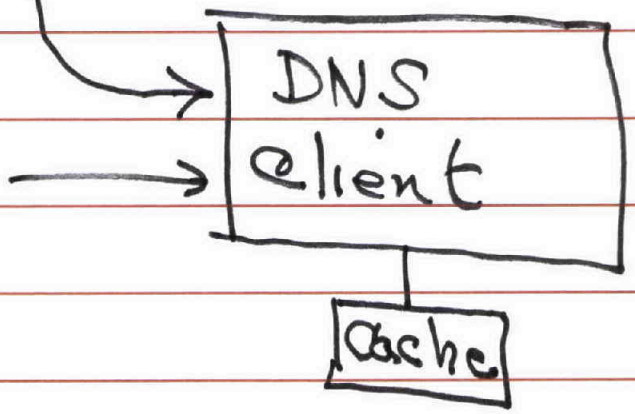
Sept 22, Tuesday

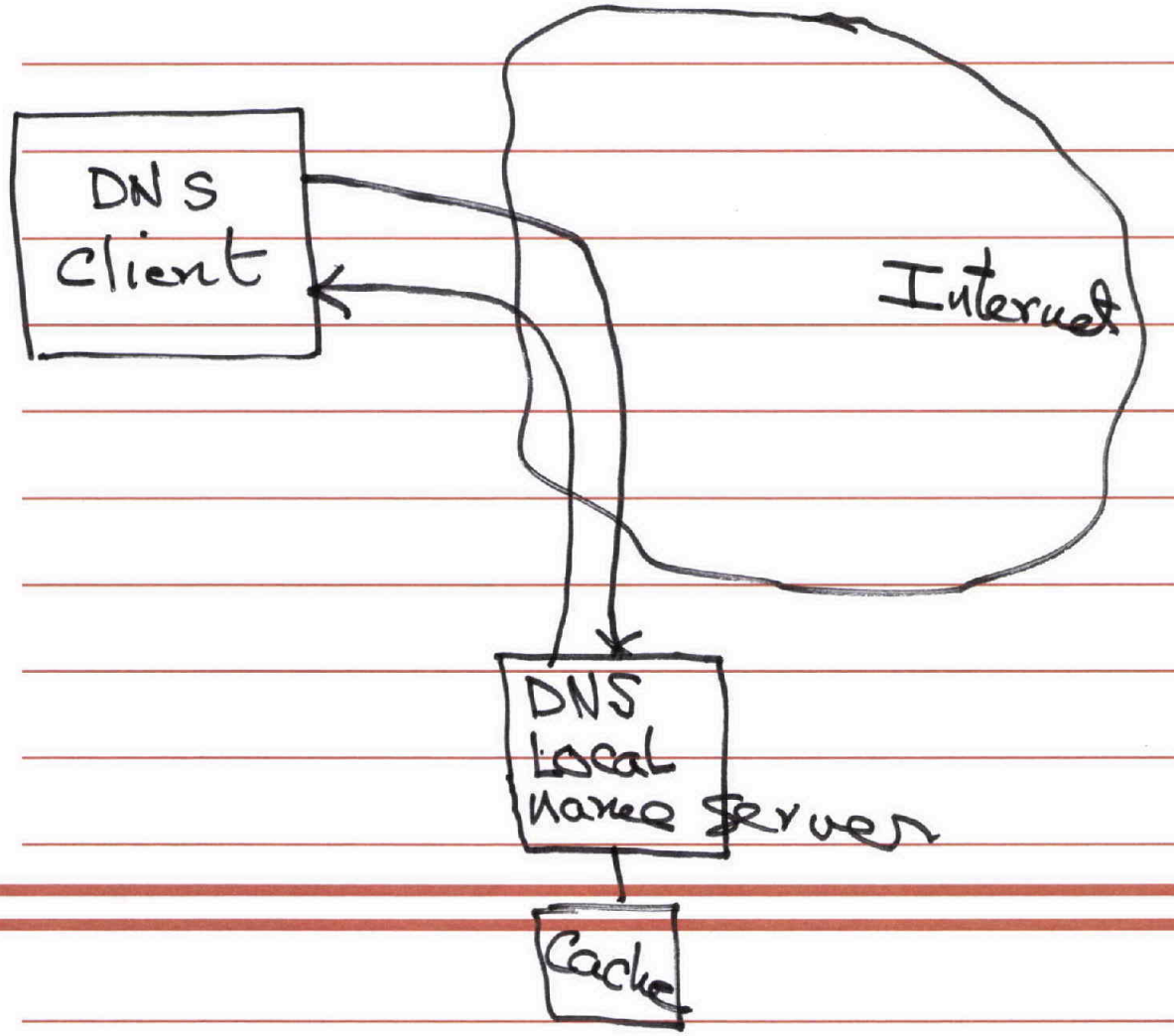


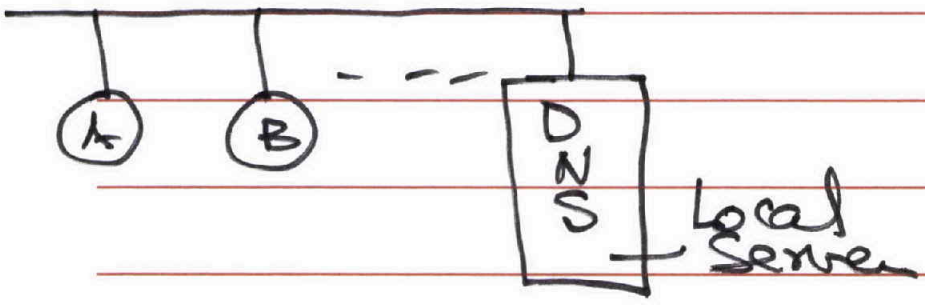
http://www.usc.edu/guedes.html

protocol | host names | file names | format.

a.k.a. "resolver"




DNS
client
Internet
DNS
Local
Name Server
Cache

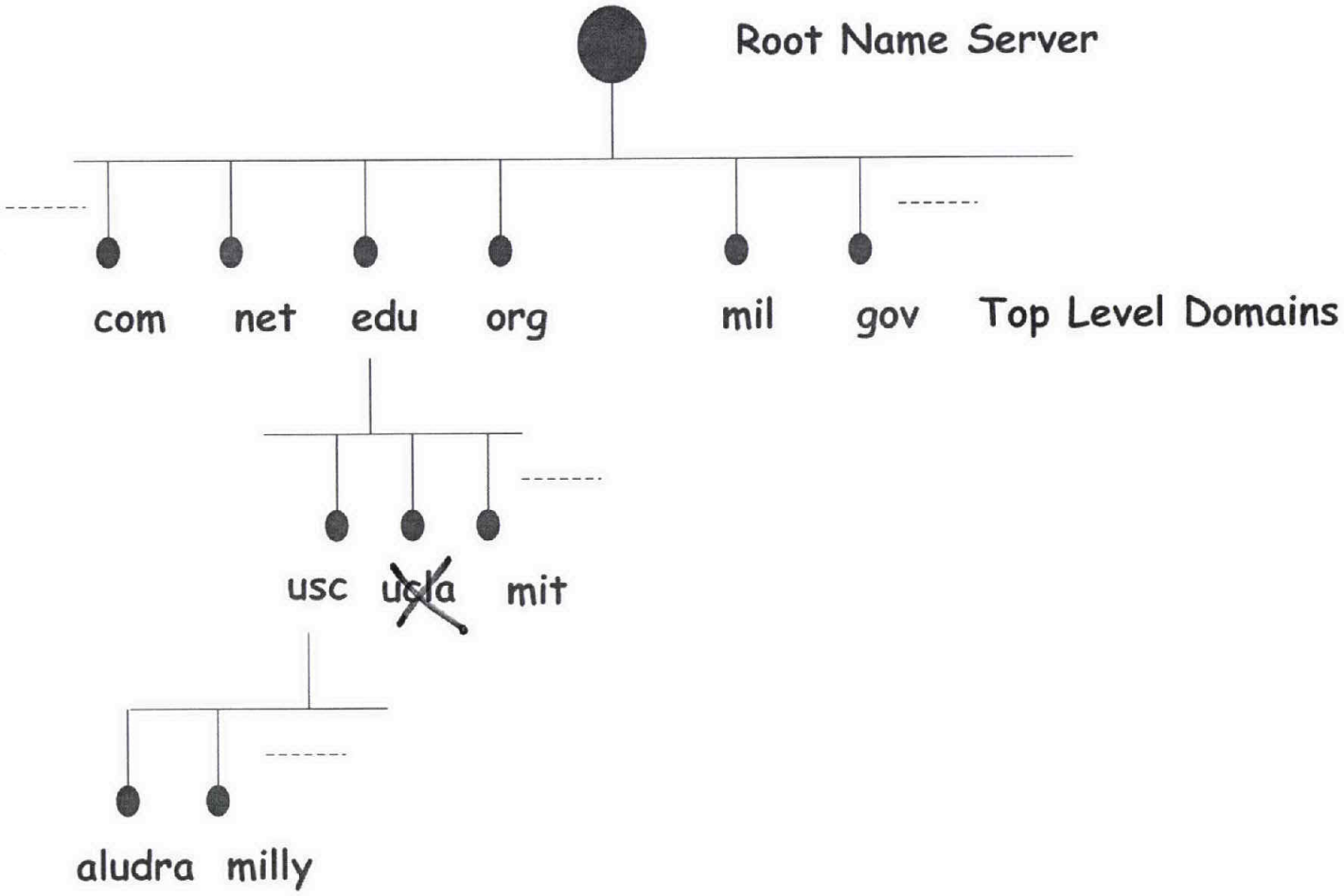

A
B
DNS
Local
Server

Question #2:
How does a Host get the IP address of another Host across the Internet?

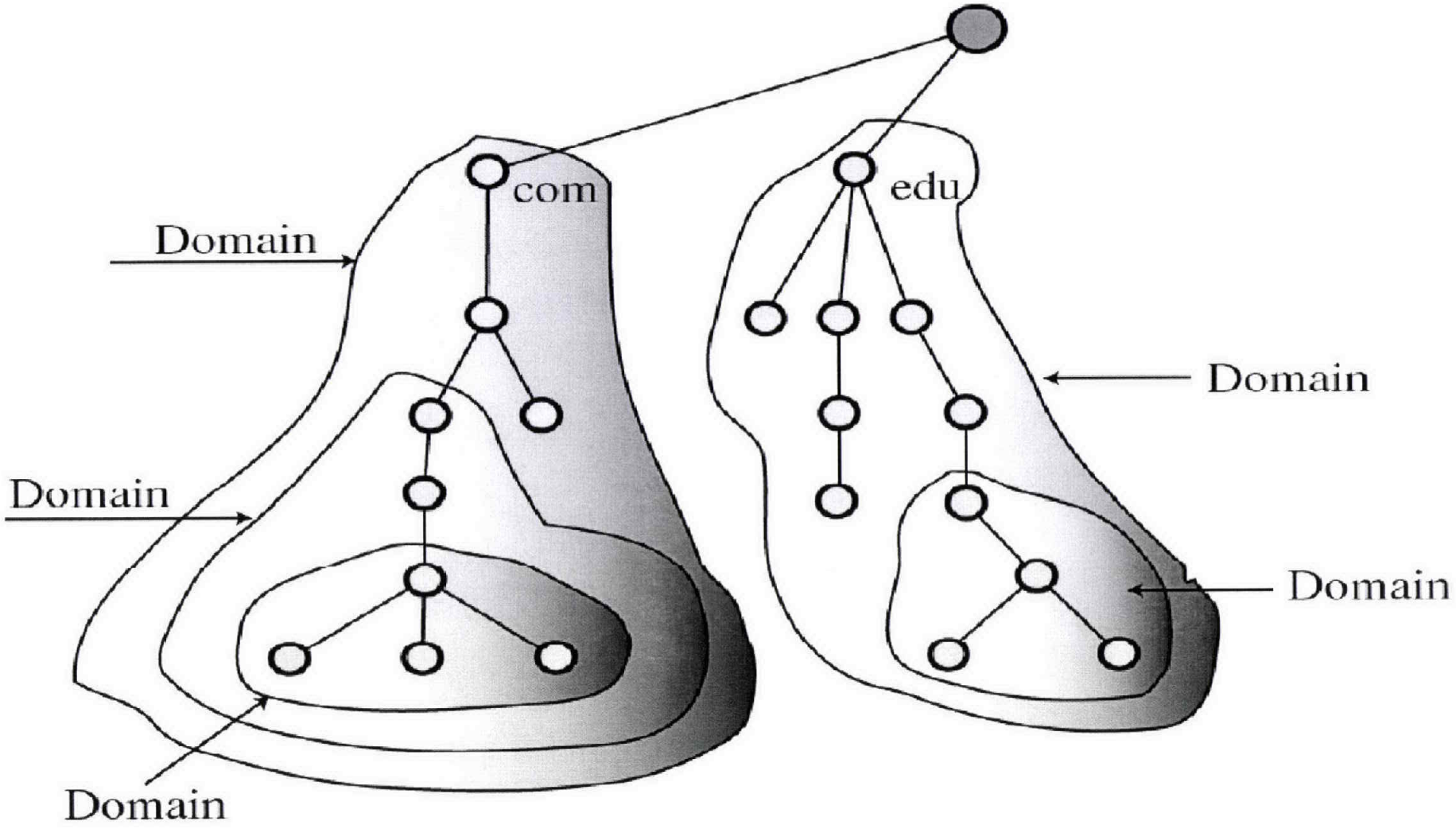
Domain Name Services (DNS)

- DNS is a TCP/IP client server application protocol that allow host and name servers to communicate in order to provide host name to IP address translation
- DNS uses a distributed, hierarchical naming structure by defining several *Domains*. A domain is a collection of sites that are related in some manner
- DNS use the services of UDP, port # 53
- Application protocols such as HTTP, FTP, SMTP, etc... use the services of DNS

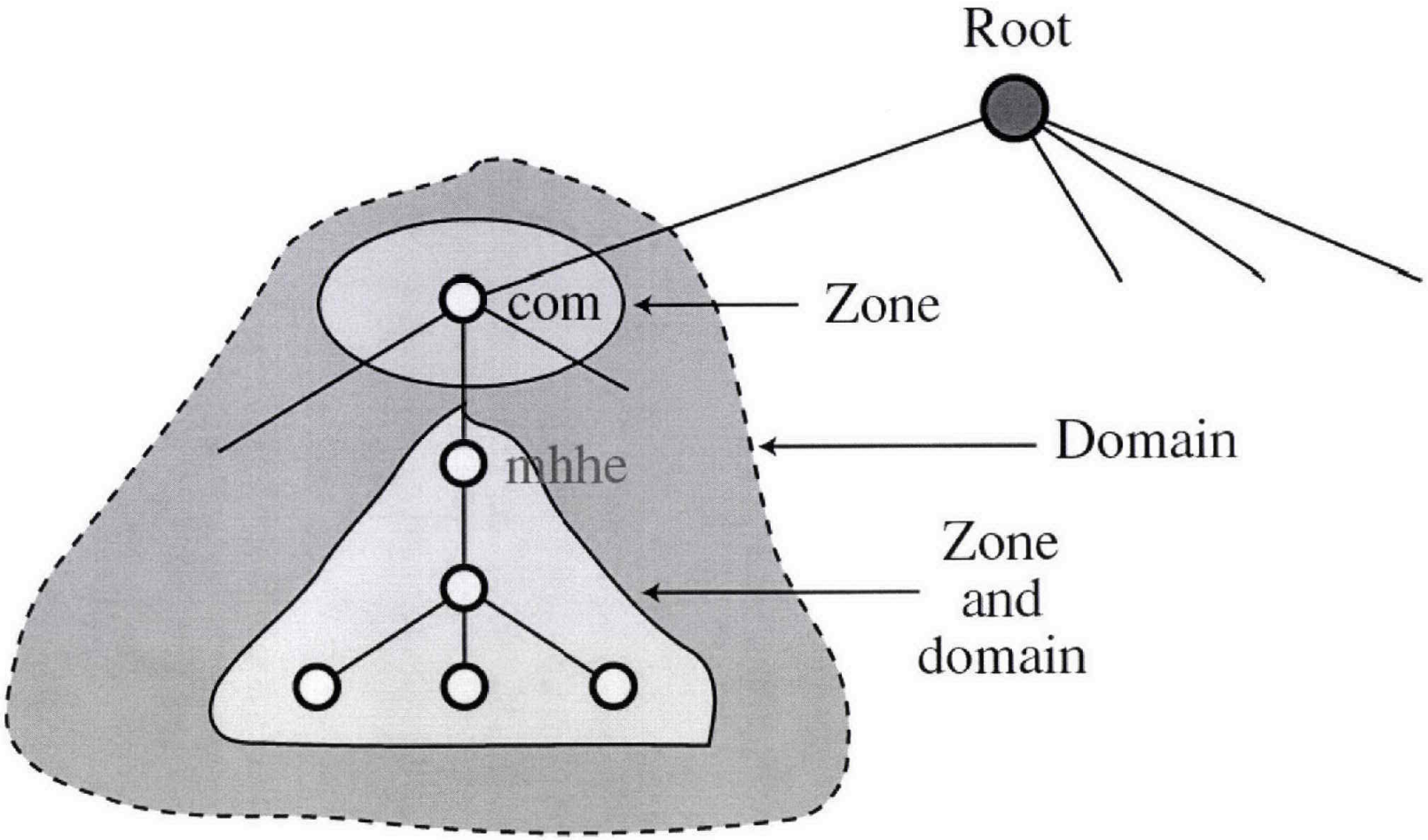
Domain Name Space



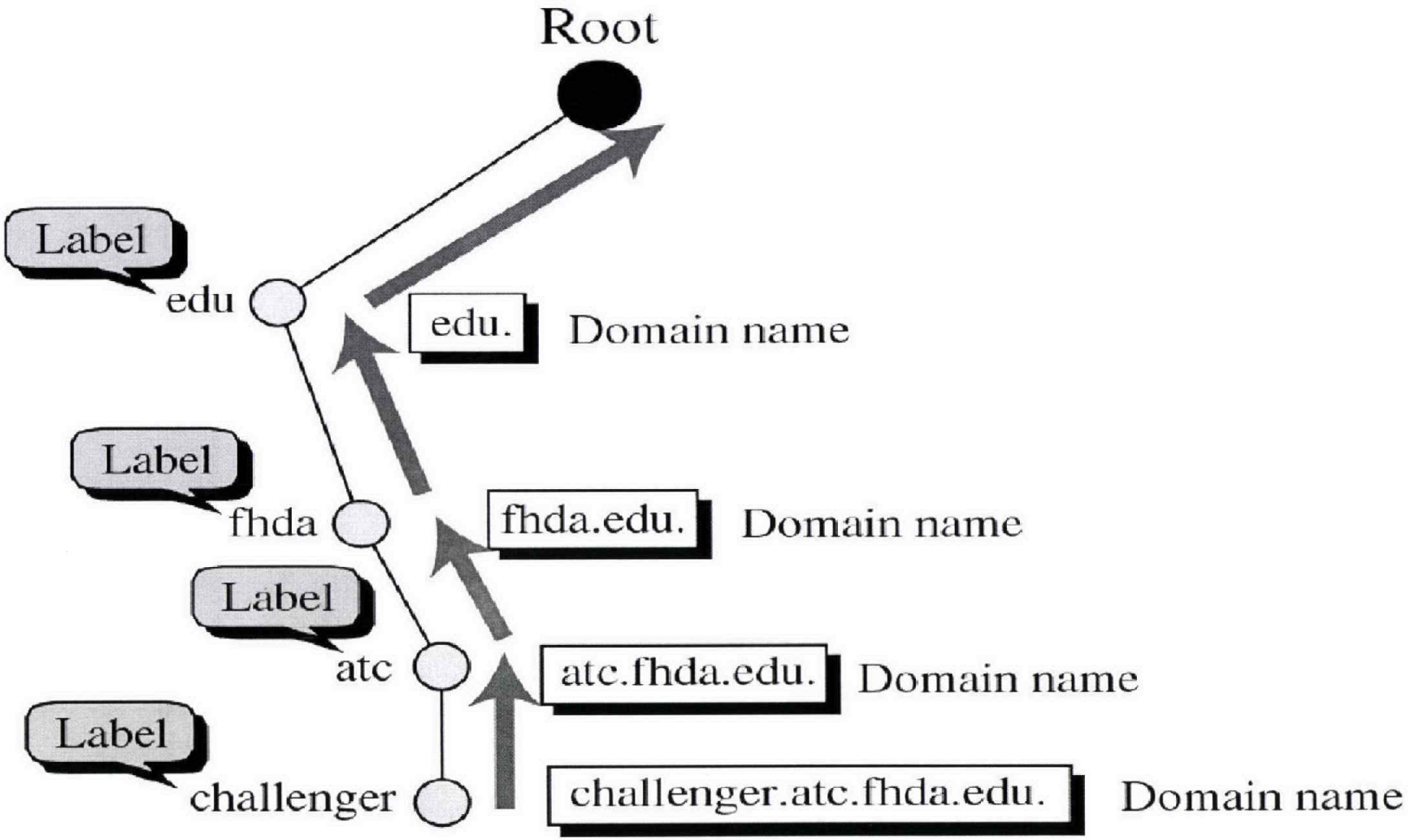
Domains



Domains and Zones



Domain Names & Labels

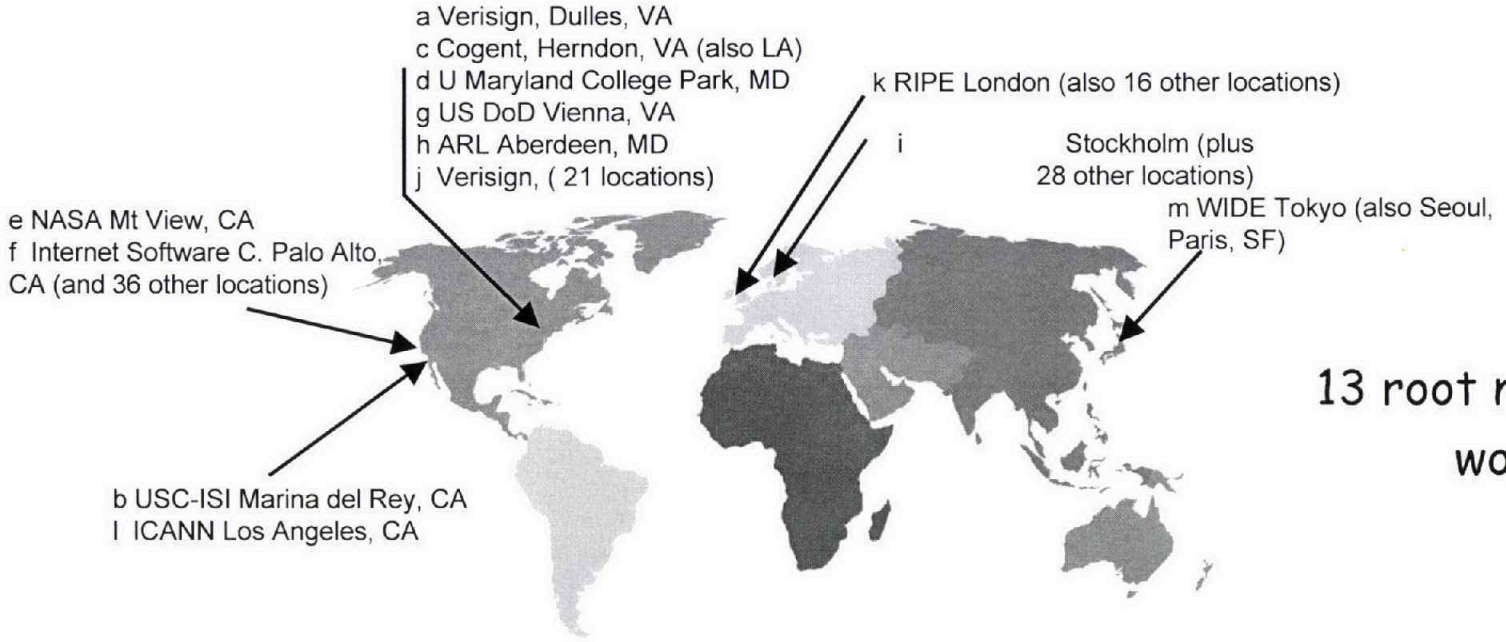


Name Servers

- Local Name Servers: This is the default name server (in department, university, company, residential ISP, etc...) that will receive the DNS query from the host
  - The IP address of the default local name server is configured manually in the host
- Root Name Servers: There are 13 root name servers most of which are located in US (two of them in Marina Del Rey). When a local name server can't satisfy the query from a host, it will behave as a DNS client and queries one of the root servers. If the root name server can't satisfy the query, it consult with
- Authoritative Name Server: This is where the host register its name/IP address

Root Name Servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Gets mapping
  - Returns mapping to local name server



13 root name servers worldwide

educause

TLD and Authoritative Name Servers

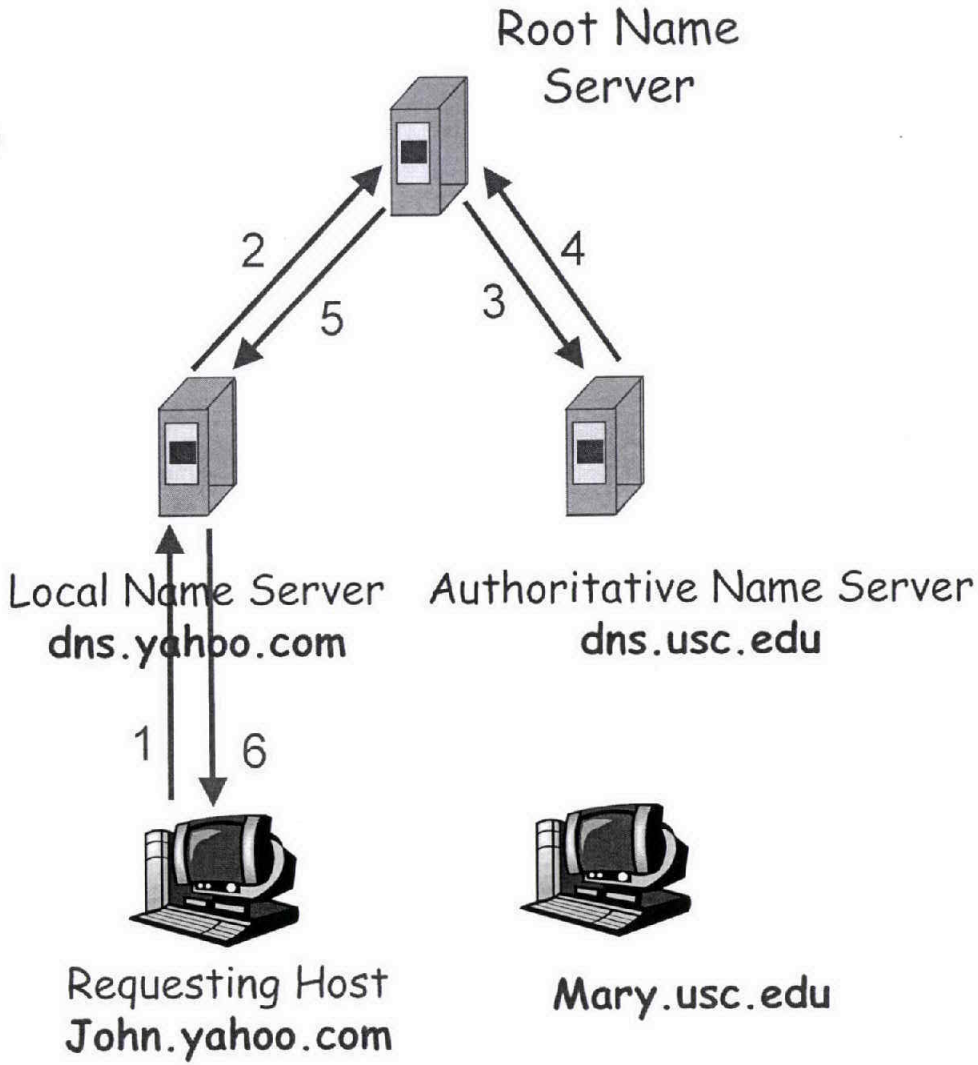
- Top-level domain (TLD) servers:
  - Responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp, in, cn
  - Network Solutions maintains servers for com TLD
  - Educause for edu TLD
- Authoritative DNS servers:
  - Organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - Can be maintained by organization or service provider

Local Name Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy

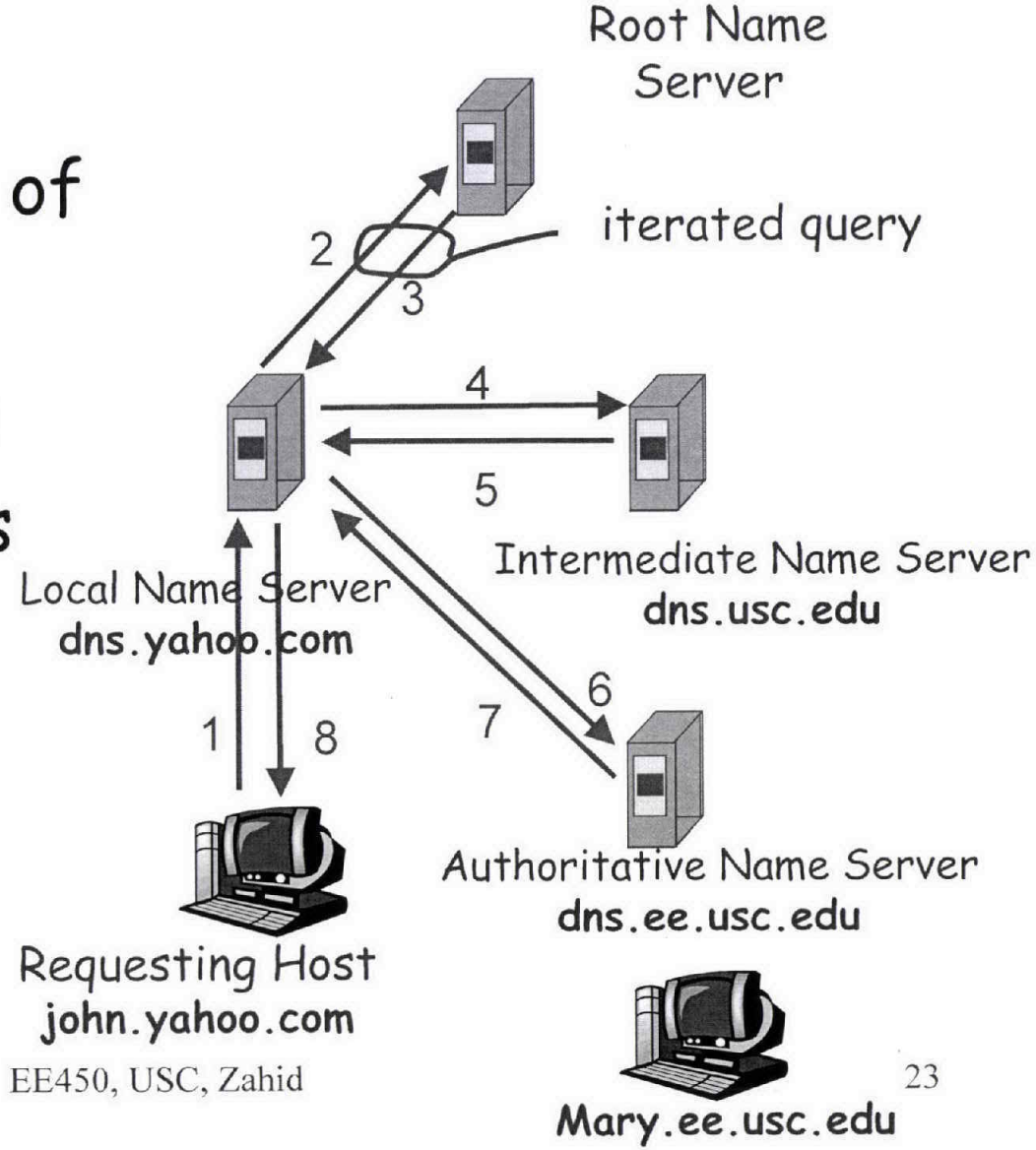
Recursive DNS

- Host "A" whose name is John.yahoo.com wants the IP address of another host "B" whose name is Mary.usc.edu
- Host "A" Contacts its local DNS server, dns.yahoo.com
- dns.yahoo.com contacts root name server, if necessary
- Root name server contacts authoritative name server, dns.usc.com, if necessary

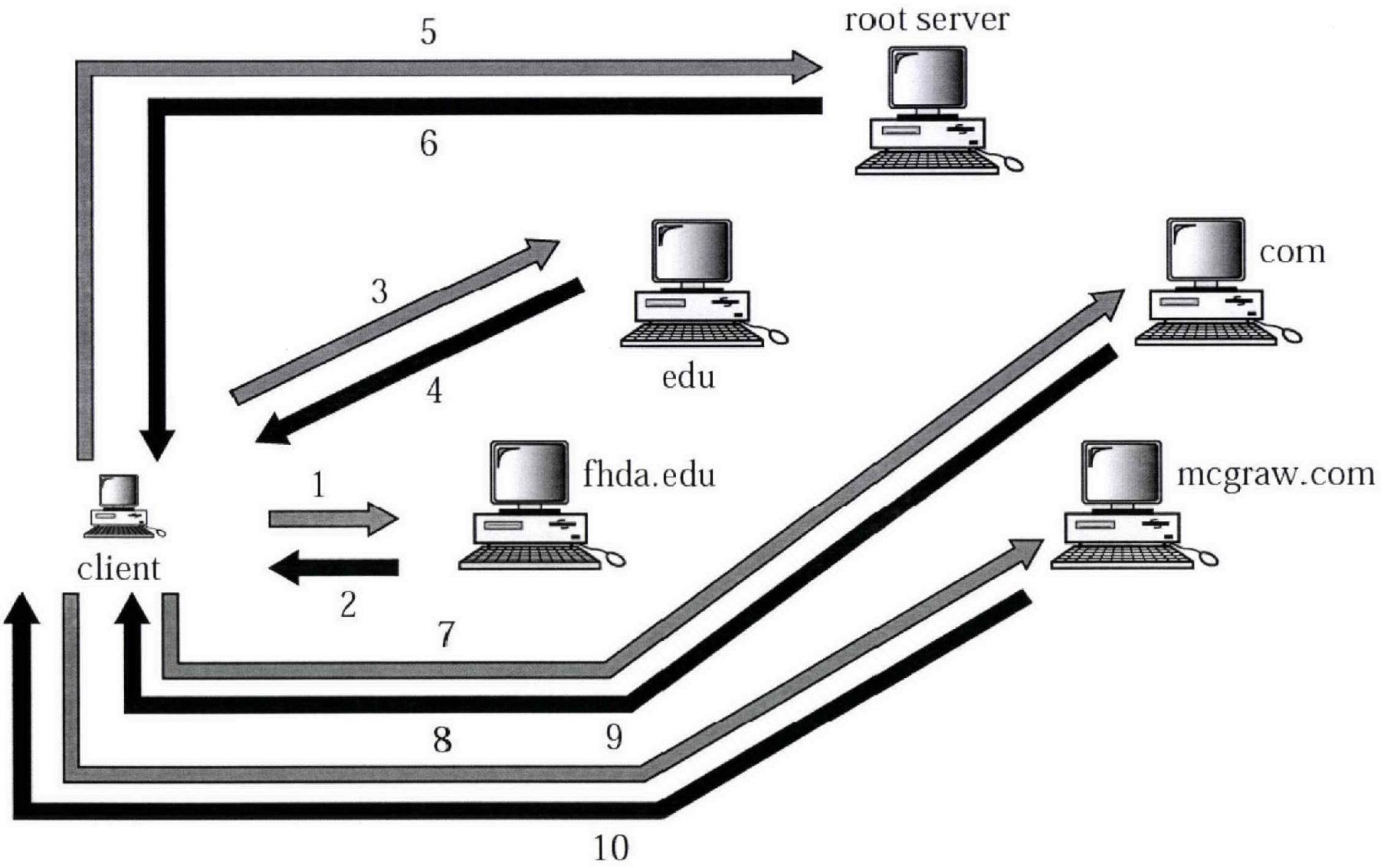


Iterative DNS

- Contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



Pure Iterative Resolution



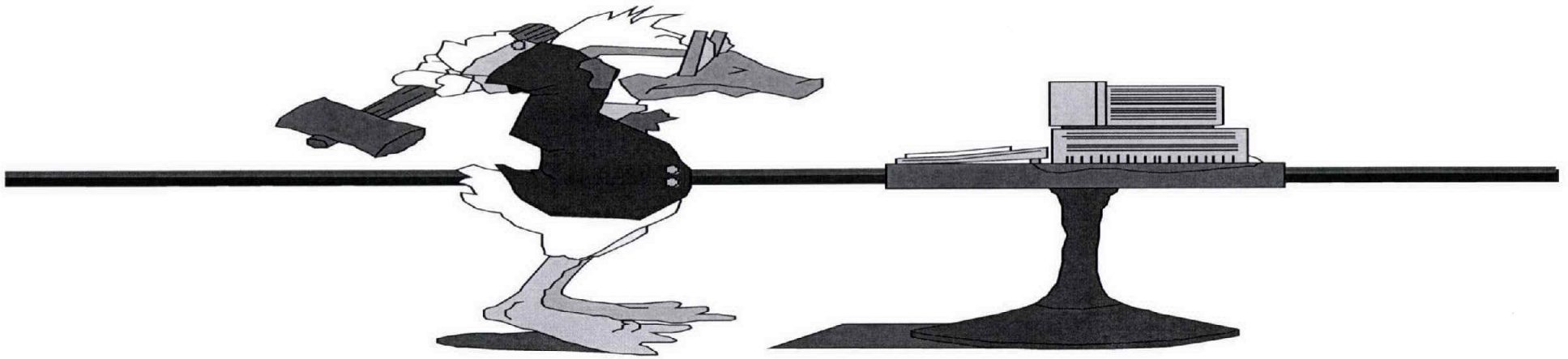
DNS Caching

- Once (any) name server learns mapping, it caches mapping
  - Cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited

Why not Centralized DNS?

- A centralized DNS represent a single point of failure. If the name server crashes so would the entire internet
- All traffic volume would have to be handled by this name server
- A single name server can't be close to all query clients ⇒ increased delays ⇒ World Wide Wait !!!!!
- Maintaining and updating a single name server is a huge task. Just dealing wit authentication/authorization is a nightmare

⇒ A single Name Server doesn't scale !



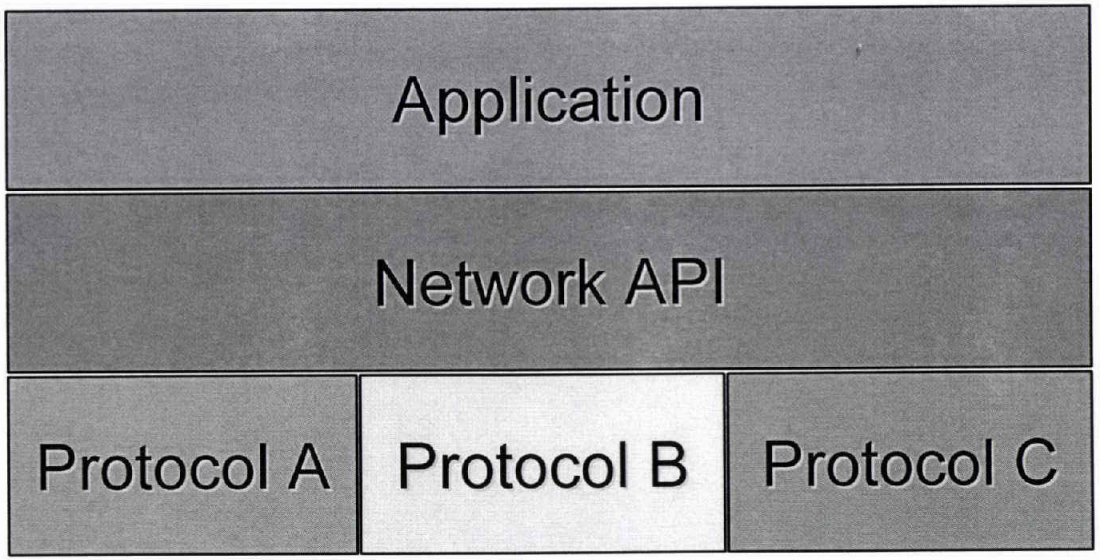
Overview of Network Programming: Sockets

EE450: Introduction to Computer Networks

Professor A. Zahid

Application Programming Interface

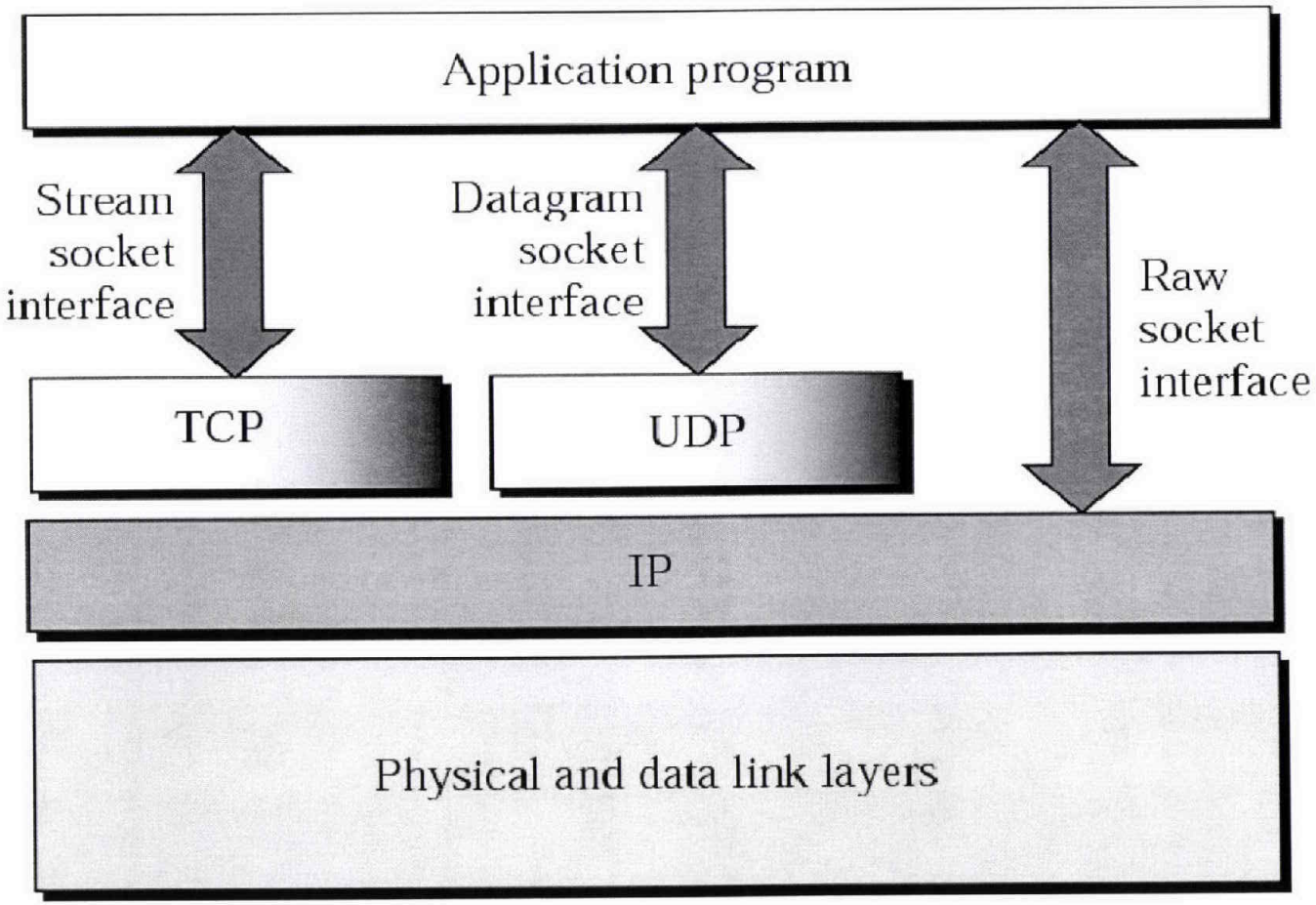
- The services provided by the operating system that provide the interface between application and the TCP/IP Protocol Suite.



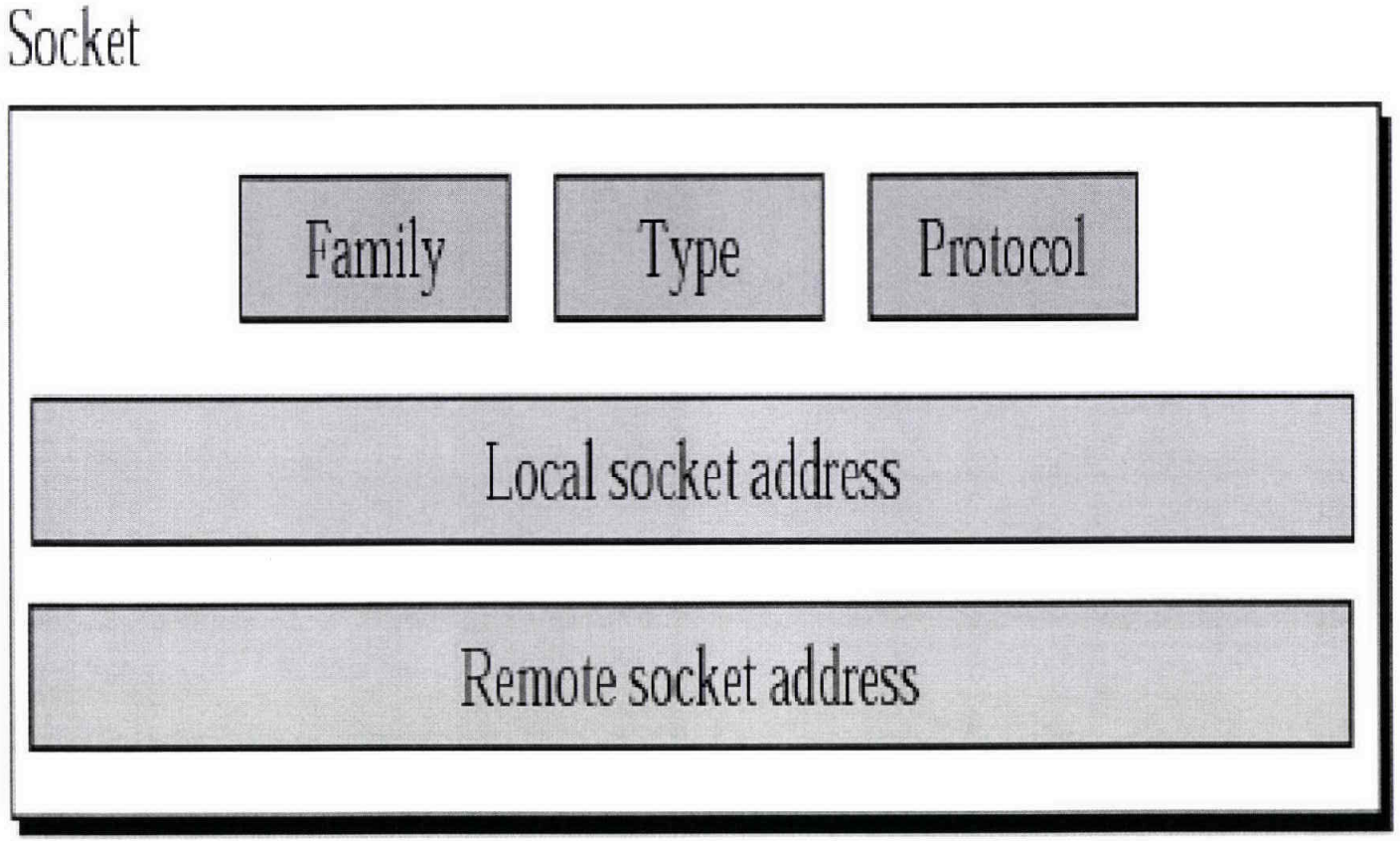
API: Sockets

- TCP/IP does not include an API definition.
- There are a variety of APIs for use with TCP/IP:
  - UNIX Sockets
  - Winsock (Windows Sockets)
- A socket is an abstract representation of a communication endpoint.
- A socket allows the application to "plug in" to the network and communicate with other applications
- A socket is uniquely identified by the IP address, Port number and the underlying transport layer protocol

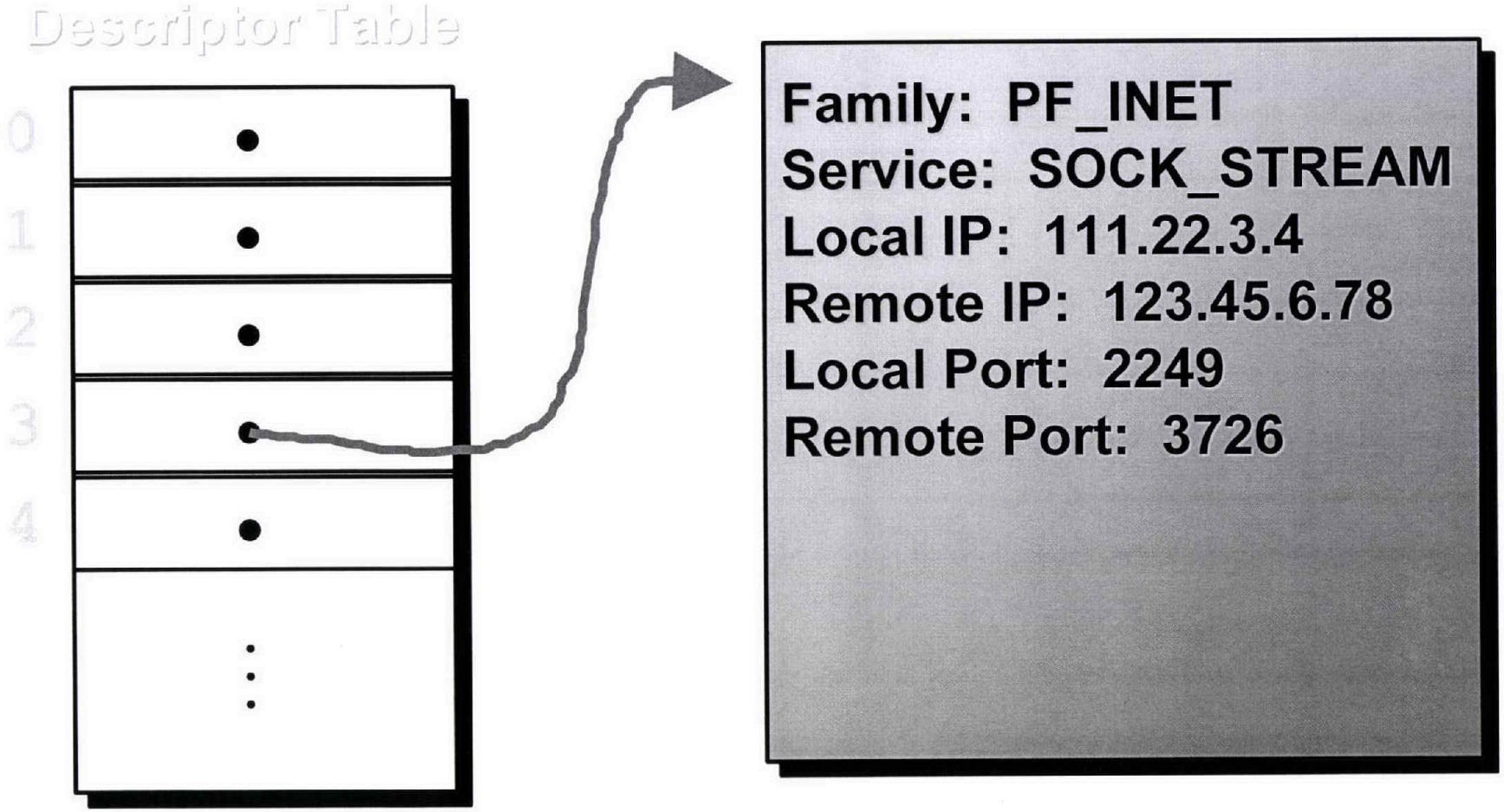
Socket Types



Socket Structure



Socket Descriptor Data Structure



Basic Sockets API

- Clients and Servers differ in some aspects of their use of API and they are the same in other aspects
- Both client and server programs begin by asking the NOS to "create" a socket. The function to accomplish that is Socket ( )
  - int socket (int protocol family, int type, int protocol)
  - Protocol family: PF_INET
  - Type: SOCK_STREAM, SOCK_DGRAM
  - Protocol: TCP, UDP
- Return value of Socket ( ) is a non-negative integer called Socket Descriptor (or -1 if errors)

TCP Server

- Create a TCP socket using Socket ( )
- Assign a port number to the socket using Bind ( )
  - int bind (int socket, local address, address length)
  - Local address is the IP address and the port number of the server. It is this address that the client and the server need to agree on to communicate. Neither one need to know the client address.
  - Address length is length of address structure
  - If successful, Bind ( ) returns a 0, otherwise it returns -1
  - If successful, the socket at server side is "associated " to the local IP address and the local port number
- Listen to connections from clients using Listen ( )
  - int listen (int socket, int queue limit)
  - Queue limit is an upper bound on # of clients that can be waiting
  - If successful, Listen ( ) returns a 0, otherwise it returns -1

TCP Server (Continued)

- The socket that has been bounded to a port and marked for listening is never actually used for sending and receiving. The socket (known as the welcoming socket or the "parent" socket.
- "Child" sockets are created for each client. It is this socket that is actually used for sending and receiving
- Server gets a socket for an incoming client connection by calling Accept ( )
  - int accept (int socket, client address, address length)
  - If successful, accept ( ) returns a descriptor for the new socket, otherwise it returns -1

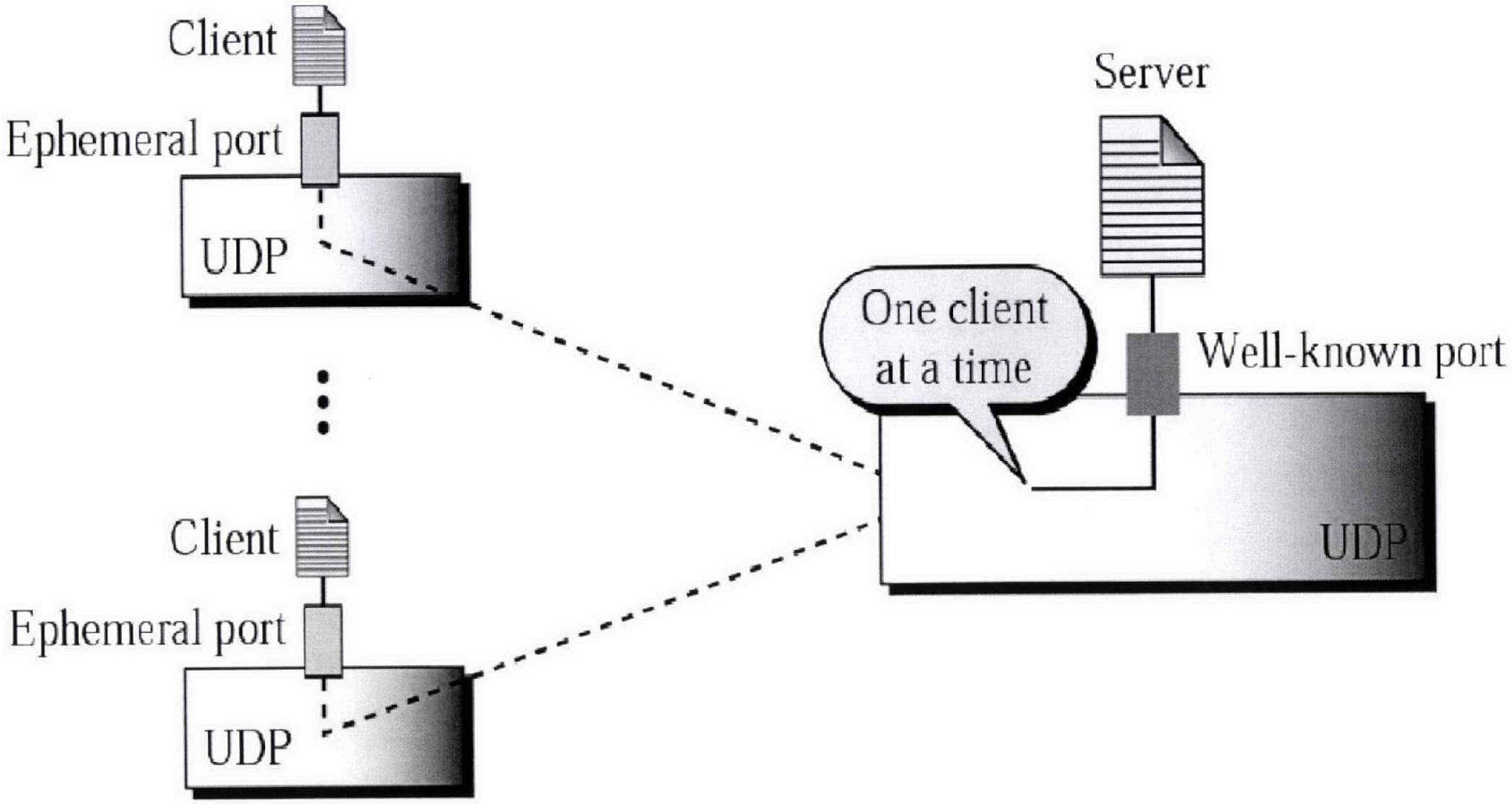
TCP Server (Continued)

- Once the child socket is created, communications (send and receive) can take place
  - int send (int socket, message, message length)
  - int recv (int socket, recv buffer, buffer length)
- When the application is done with the socket, it needs to close it (The child socket, not the parent socket which is passively open to welcome new clients). This is done by calling Close ( )
  - int close (int socket)

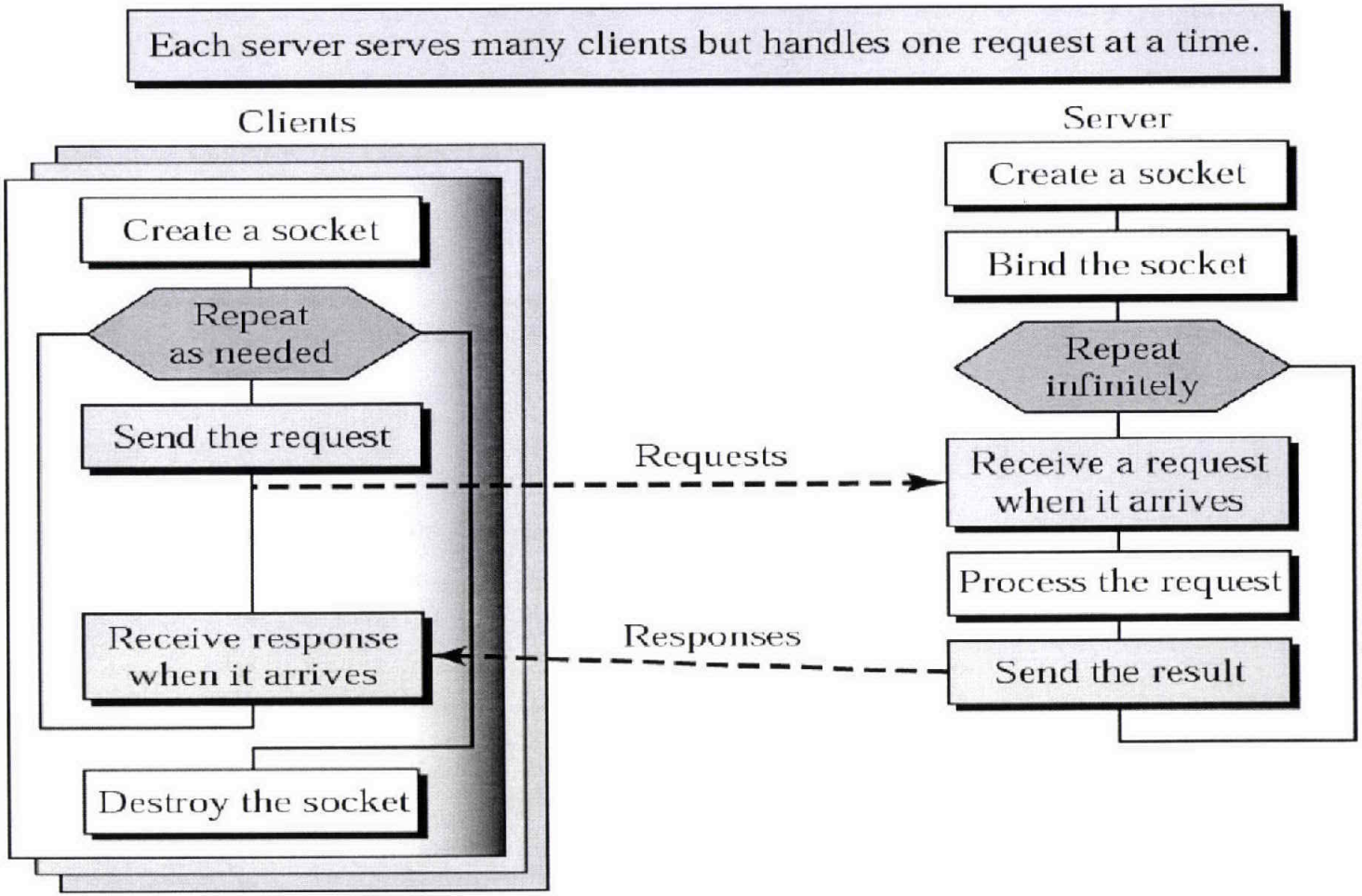
TCP Client

- Create a TCP socket using Socket ( )
- Establish a connection to the server using Connect ( )
  - int connect (int socket, foreign address, address length)
  - Foreign address is the address of the server and the port number of the server (The well-known port number)
- Communications using Send and Recv
  - int send (int socket, message, message length)
  - int recv (int socket, recv buffer, buffer length)
- Close the socket using Close ( )
  - int close (int socket)

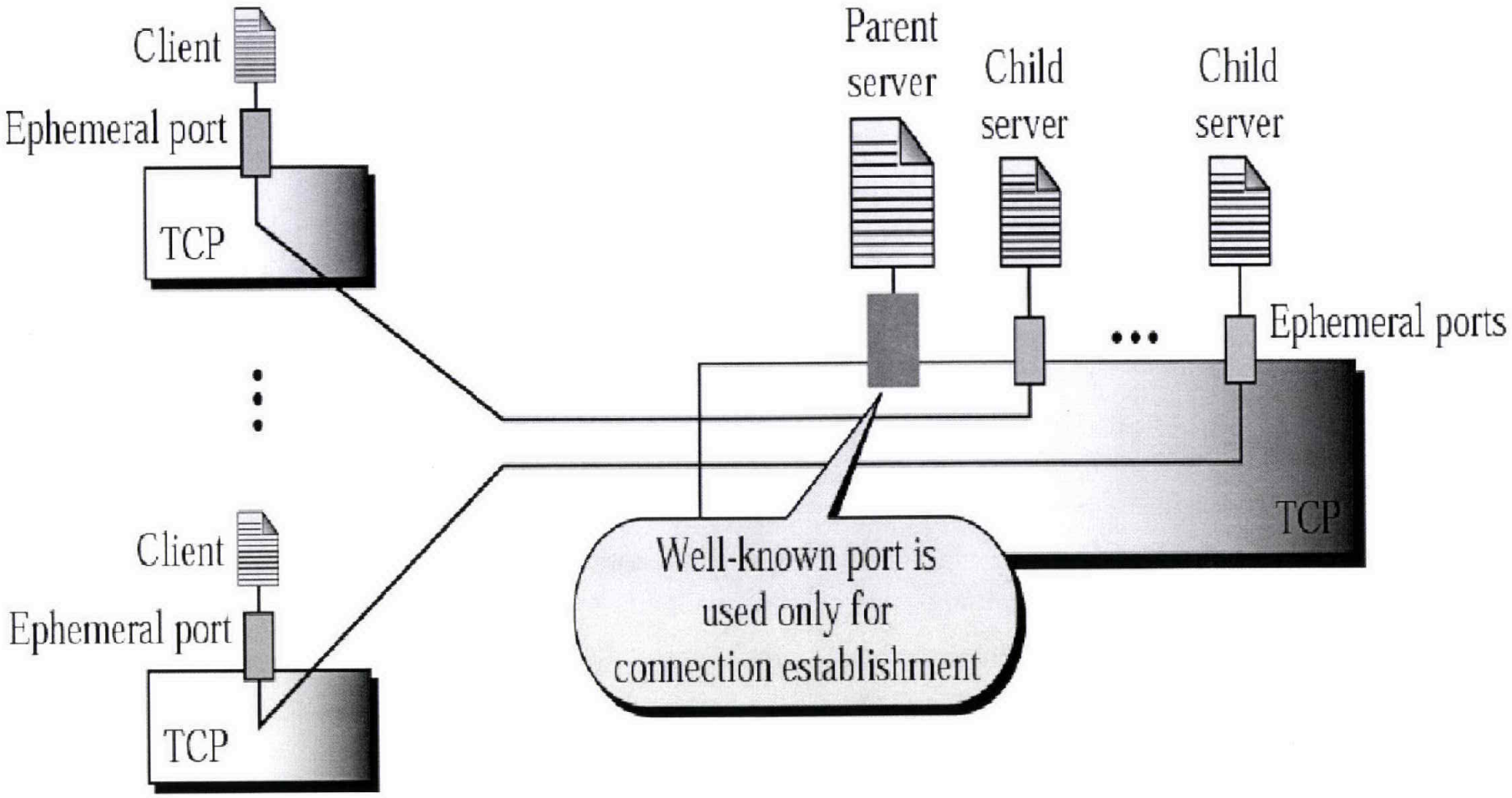
Connection-less Iterative Server



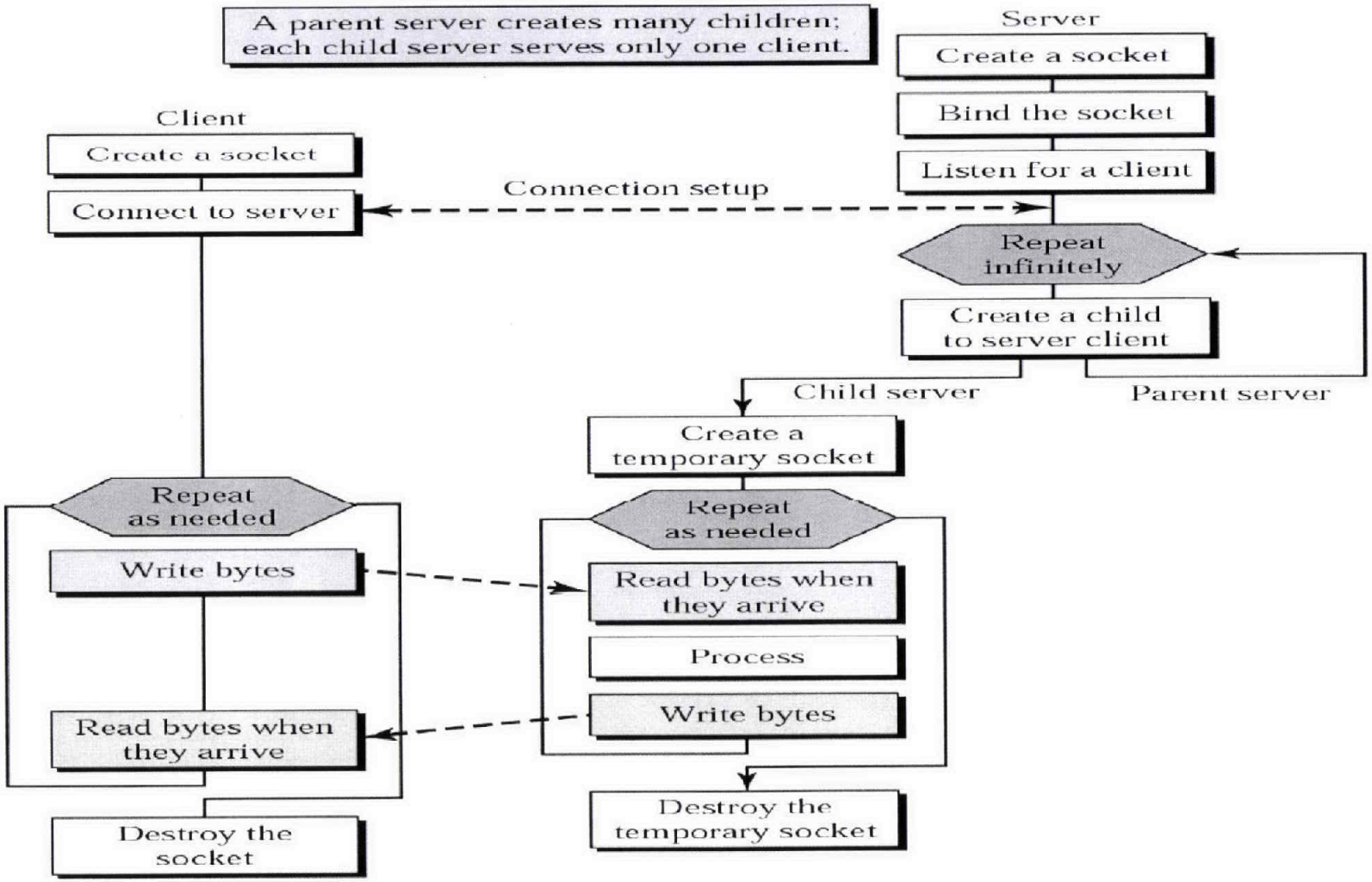
Flow Chart: Socket Interface for Connection-Less Iterative Server



Connection-oriented Concurrent Server



Flow Chart: Socket Interface for Connection-Oriented Concurrent Server



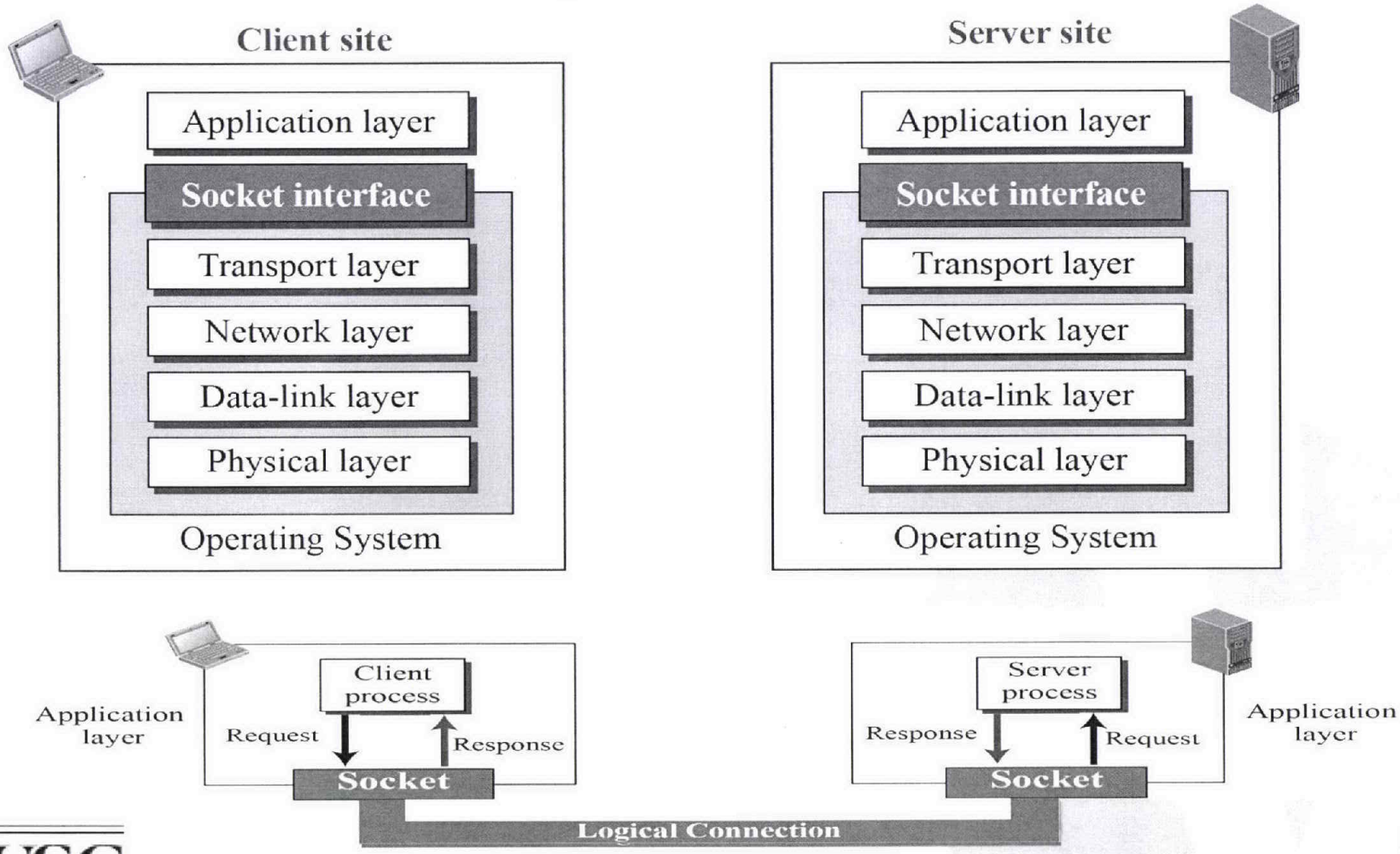
Additional Charts on Sockets

EE450: Computer Networks

University of Southern California

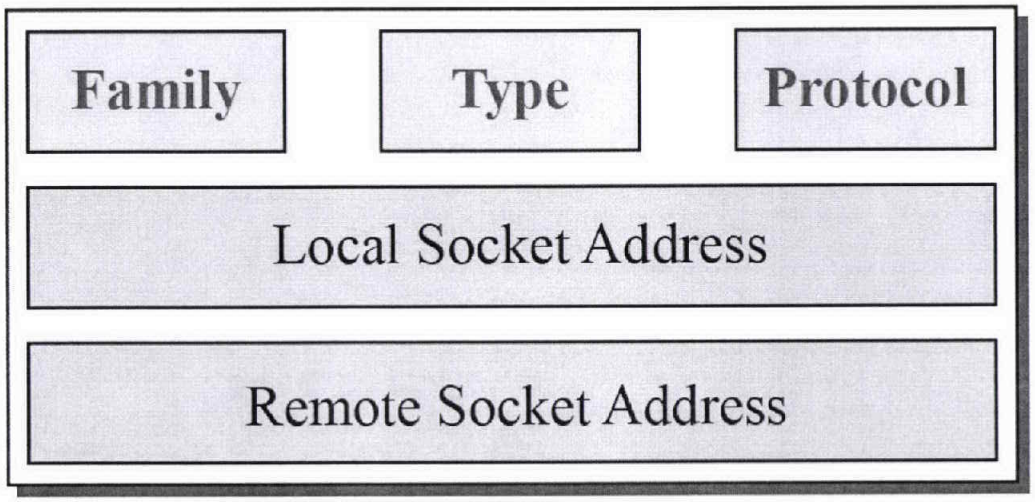
Professor: A. Zahid

Concept of Sockets

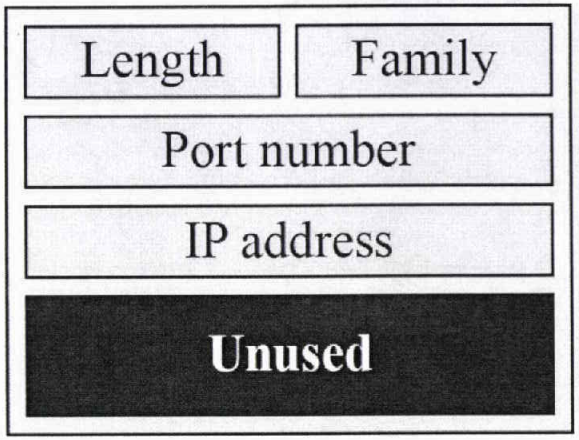


Process to Process Communications through Sockets

Socket Structure and Address

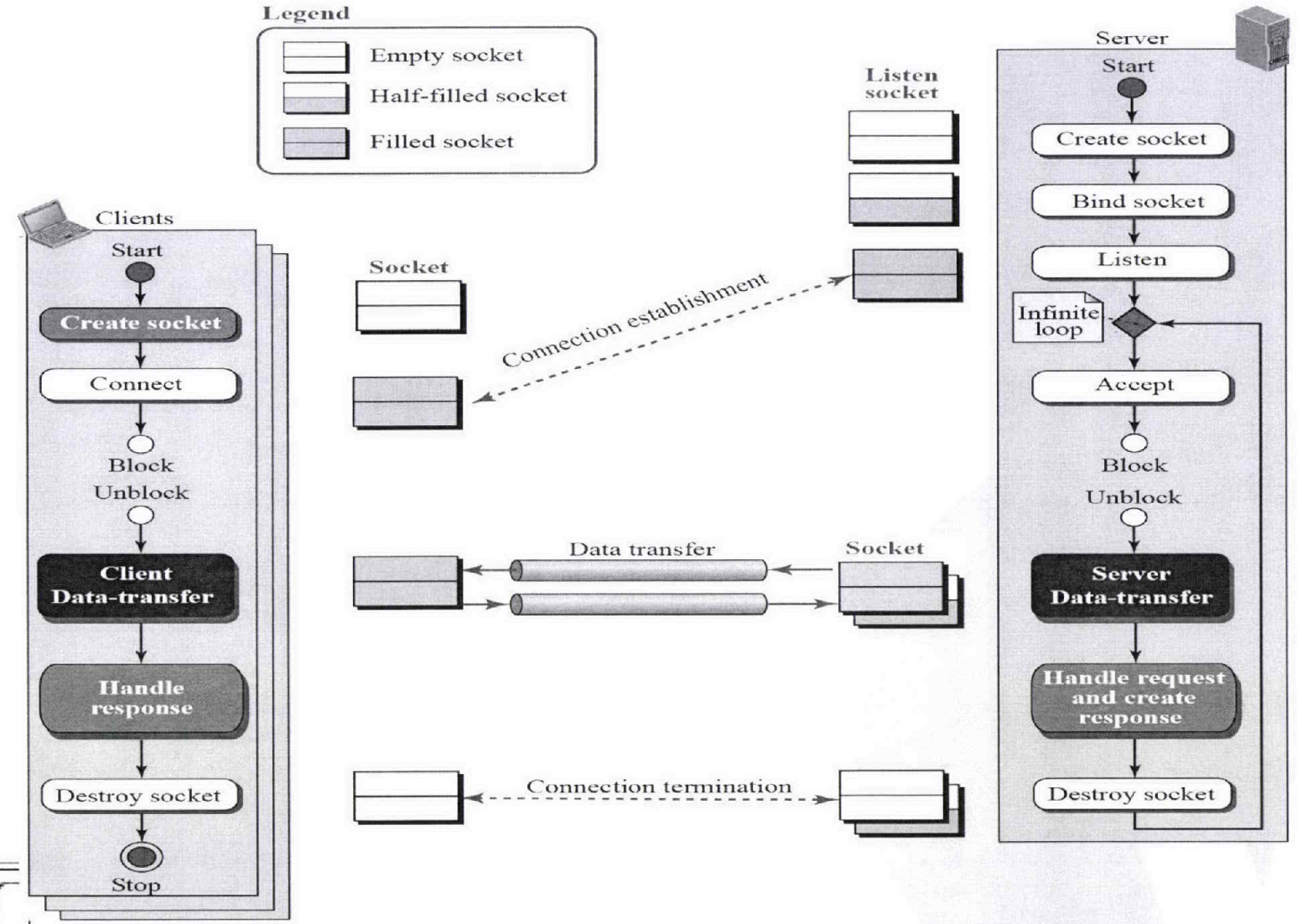


Socket

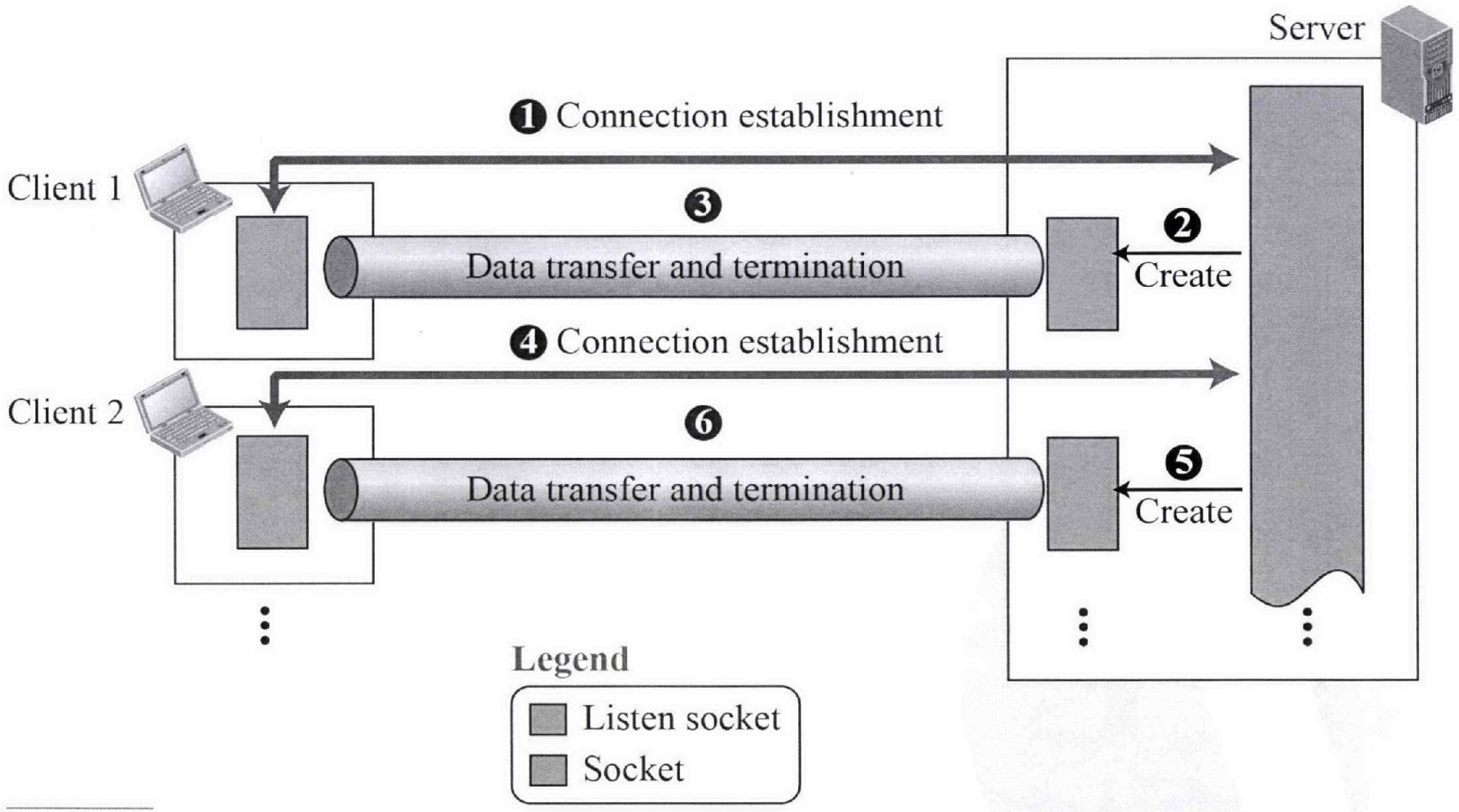


Socket address

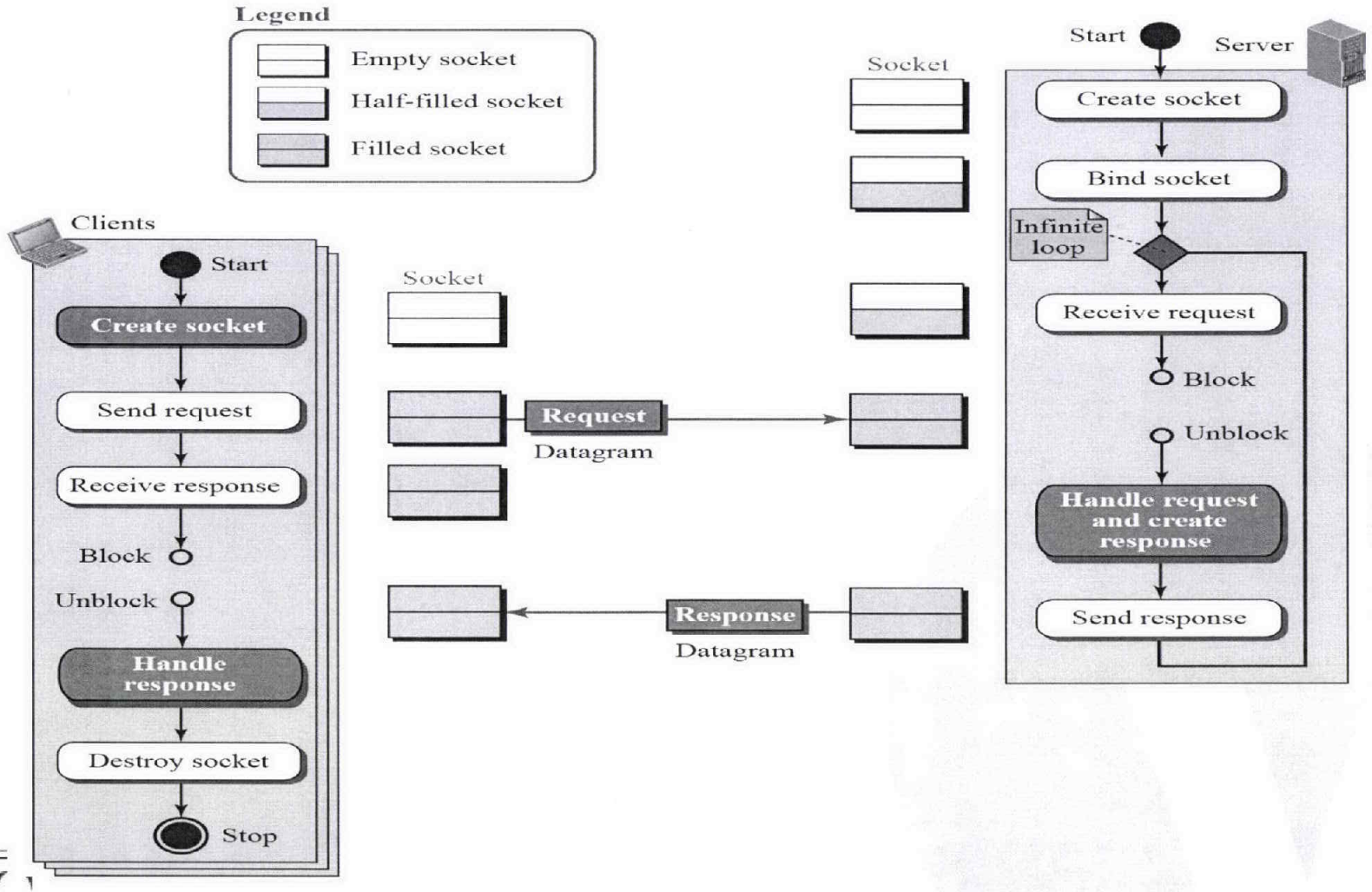
TCP Sockets (Stream Sockets)



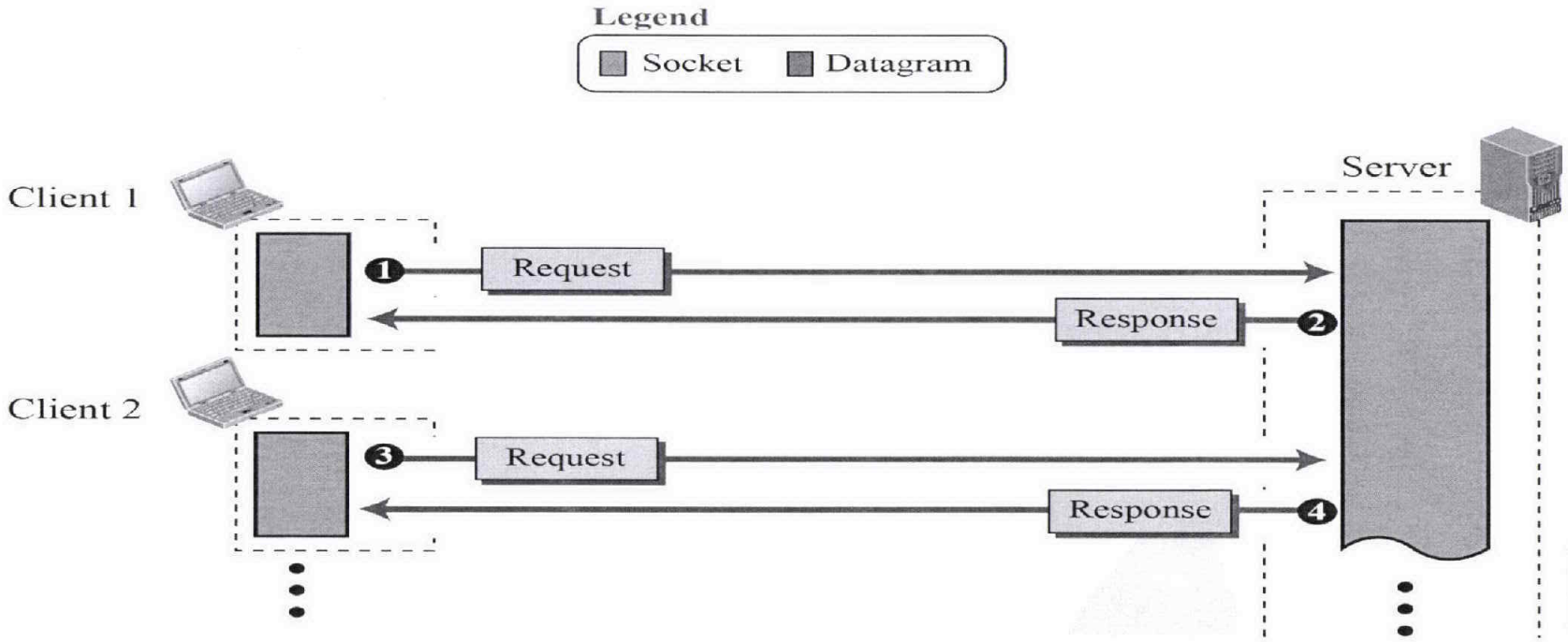
TCP Sockets (Concurrent)



UDP (Iterative) Datagram Sockets



UDP Datagram Sockets (Iterative)



An iterative server can process one client request at a time; it receives a request, processes it, and sends the response to the requestor before handling another request. When the server is handling the request from a client, the requests from other clients, and even other requests from the same client, need to be queued at the server site and wait for the server to be freed.