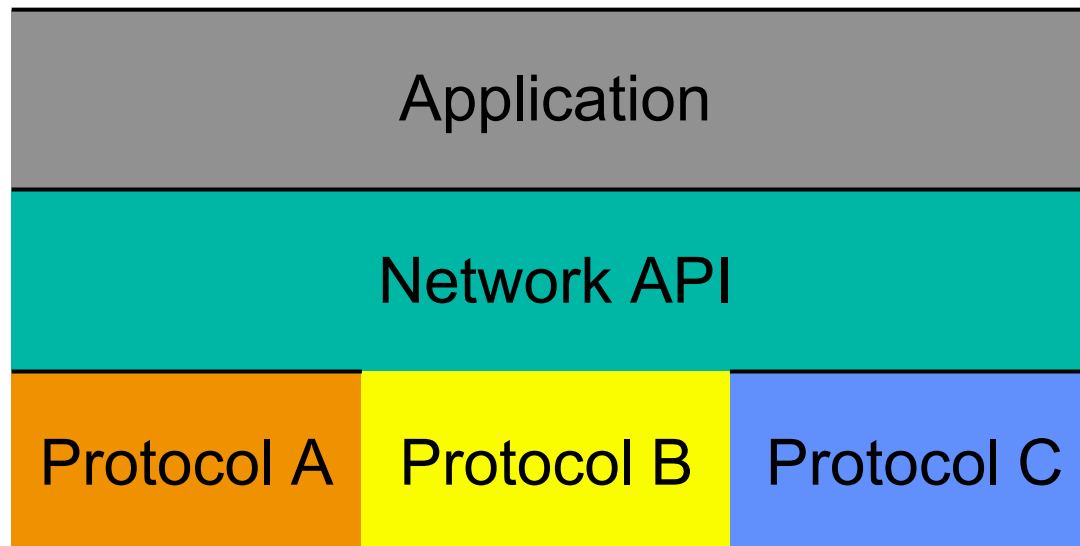# Overview of Network Programming: Sockets

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.

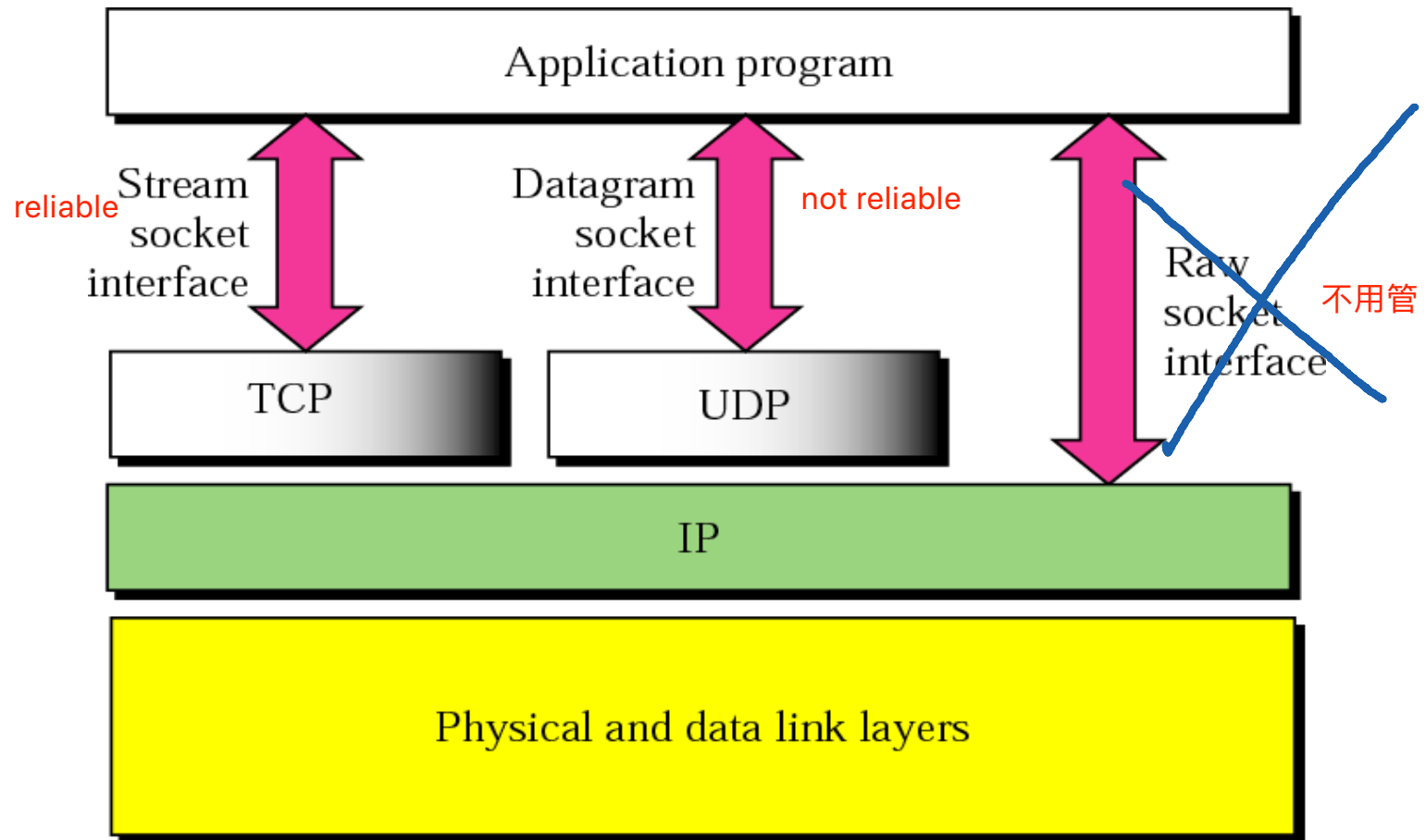| Application		
Network API		
Protocol A	Protocol B	Protocol C

API: Sockets

- TCP/IP does not include an API definition.

- There are a variety of APIs for use with TCP/IP:
  - UNIX Sockets   widely use
  - 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

  room

  application
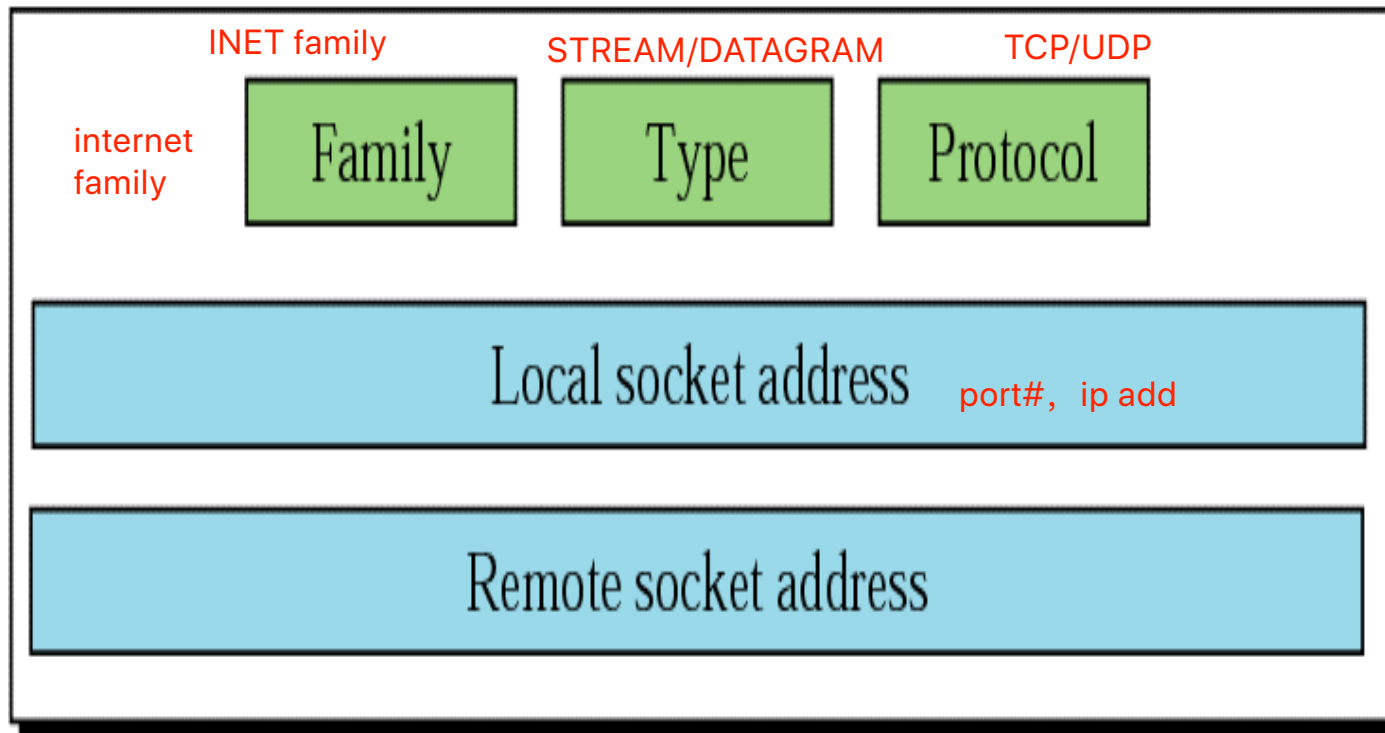
  ip add想成一座楼，port num是房号，between 2building port num可以相同

Socket Types

Application program

Stream
socket
interface
reliable

Datagram
socket
interface
not reliable

Raw
socket
interface
不用管

TCP

UDP

IP

Physical and data link layers

Socket Structure

Socket   like a file

INET family   STREAM/DATAGRAM   TCP/UDP

internet
family

| Family | Type | Protocol |

Local socket address   port#，ip add

Remote socket address

local表示自己
的地方，client
或者server都可
以是local

对应的remote
就是对方的

Socket Descriptor Data Structure

0
1
2
3
4

Family: PF_INET
Service: SOCK_STREAM
Local IP: 111.22.3.4
Remote IP: 123.45.6.78
Local Port: 2249
Remote Port: 3726
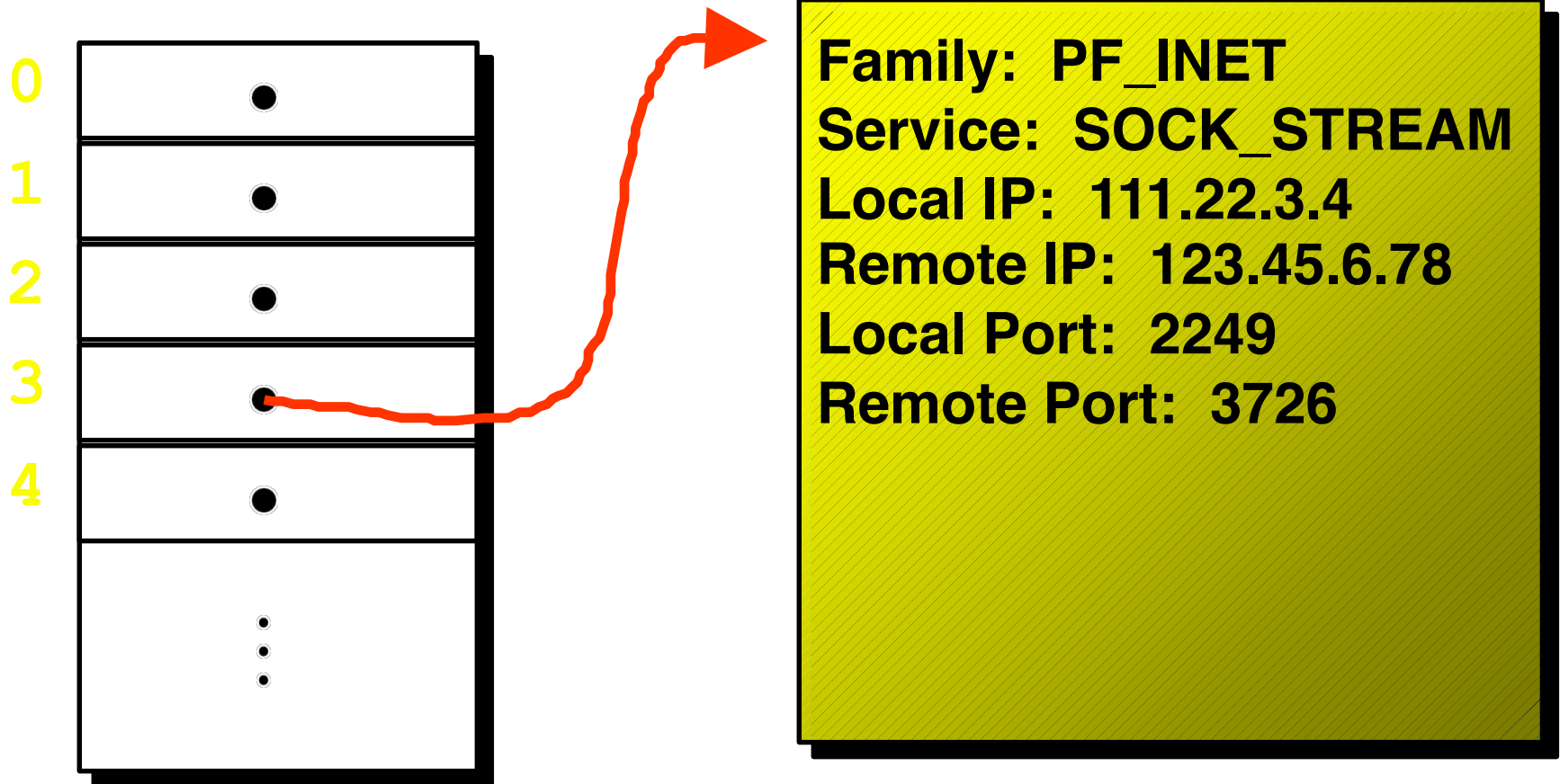
Basic Sockets API

- 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 ( )
  remote   告诉NOS   (client)
  - 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)

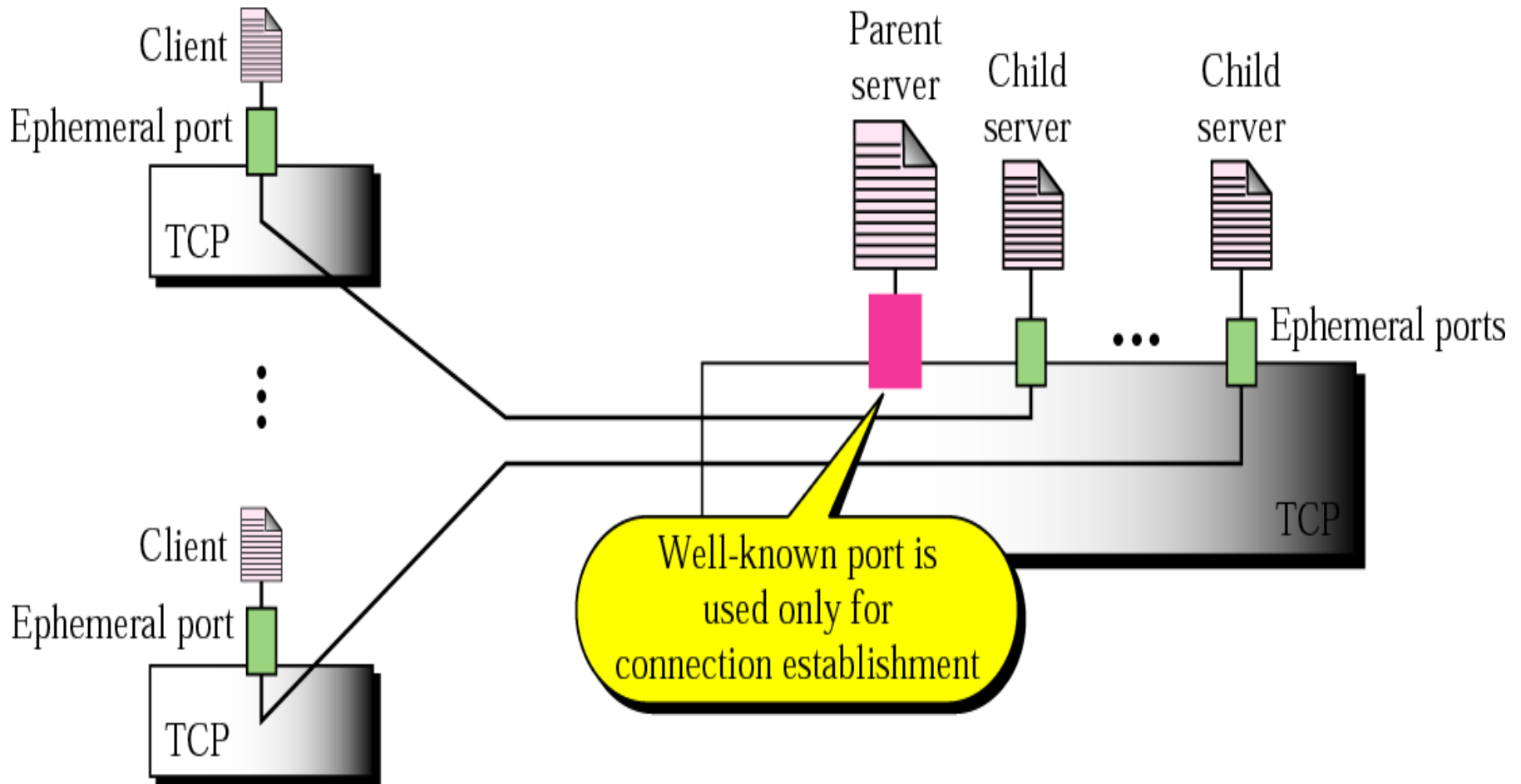- Close the socket using Close ( )
  - int close (int socket)

Connection-less Iterative Server
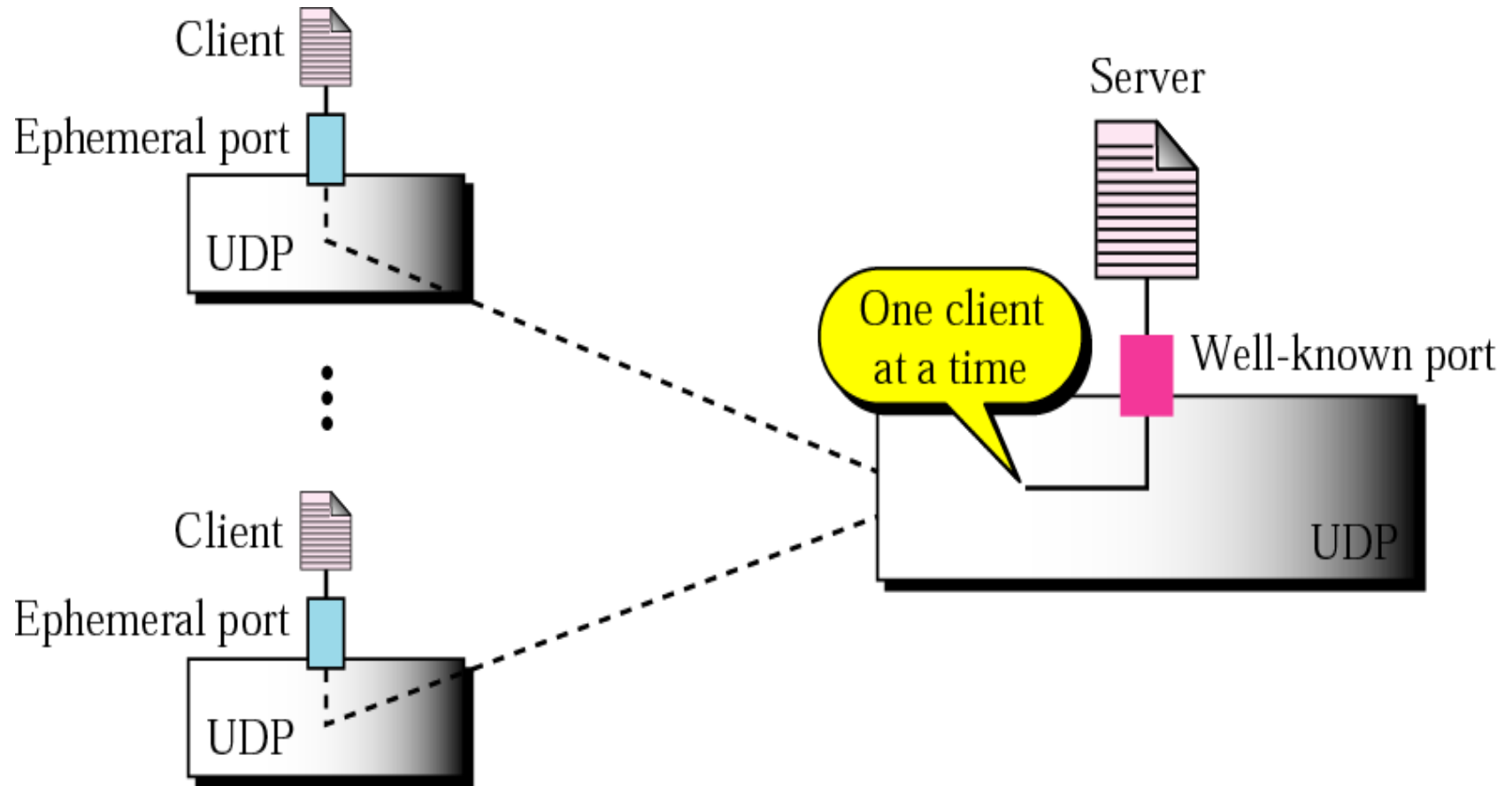
Flow Chart: Socket Interface for Connection-Less Iterative Server
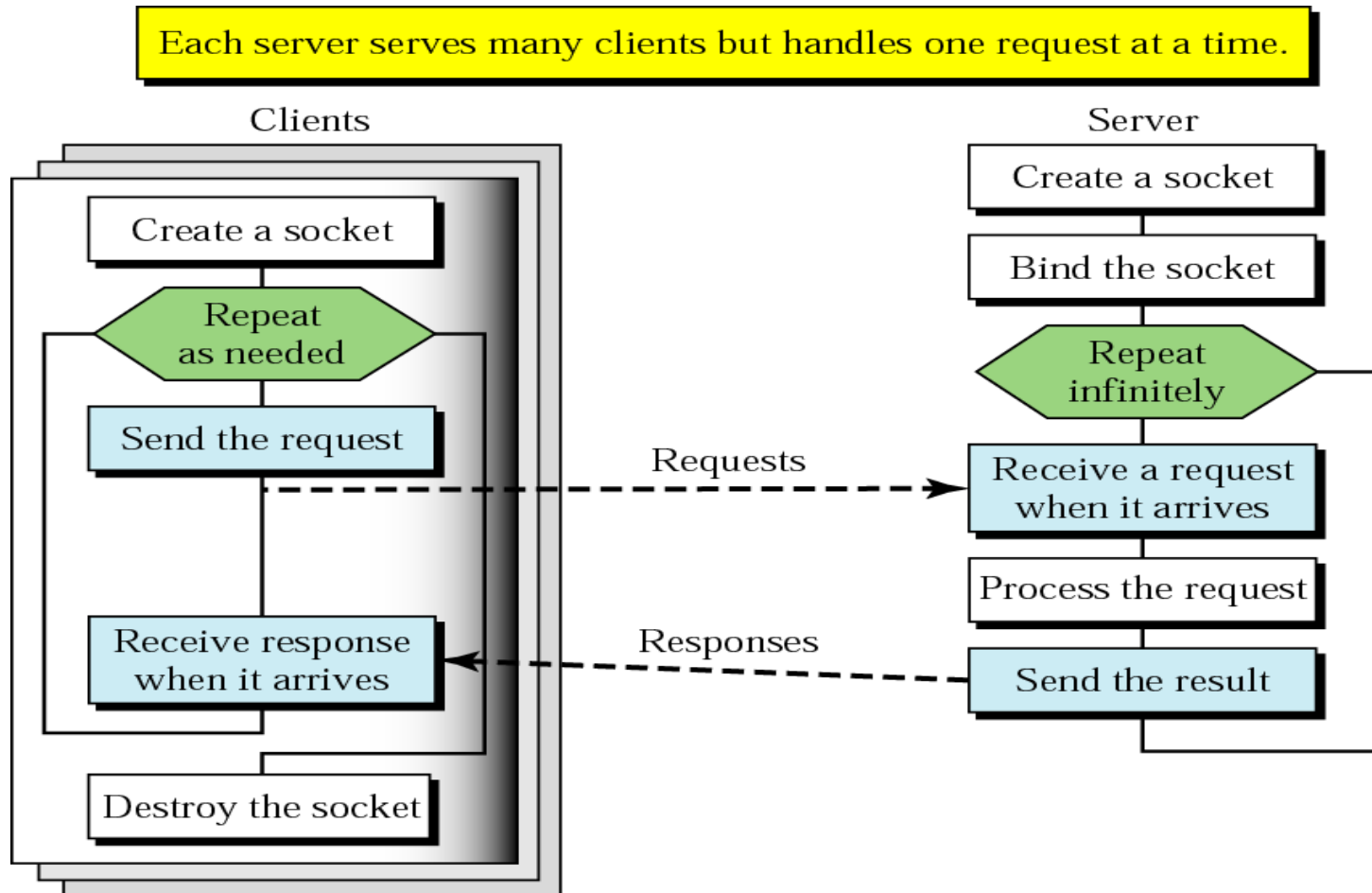
Connection-oriented Concurrent Server

Flow Chart: Socket Interface for Connection-Oriented Concurrent Server

A parent server creates many children; each child server serves only one client.

Server
- Create a socket
- Bind the socket
- Listen for a client

Client
- Create a socket
- Connect to server

Connection setup

- Repeat infinitely
- Create a child to server client

Child server / Parent server

- Create a temporary socket
- Repeat as needed
  - Read bytes when they arrive
  - Process
  - Write bytes
- Destroy the temporary socket

Client:
- Repeat as needed
  - Write bytes
  - Read bytes when they arrive
- Destroy the socket