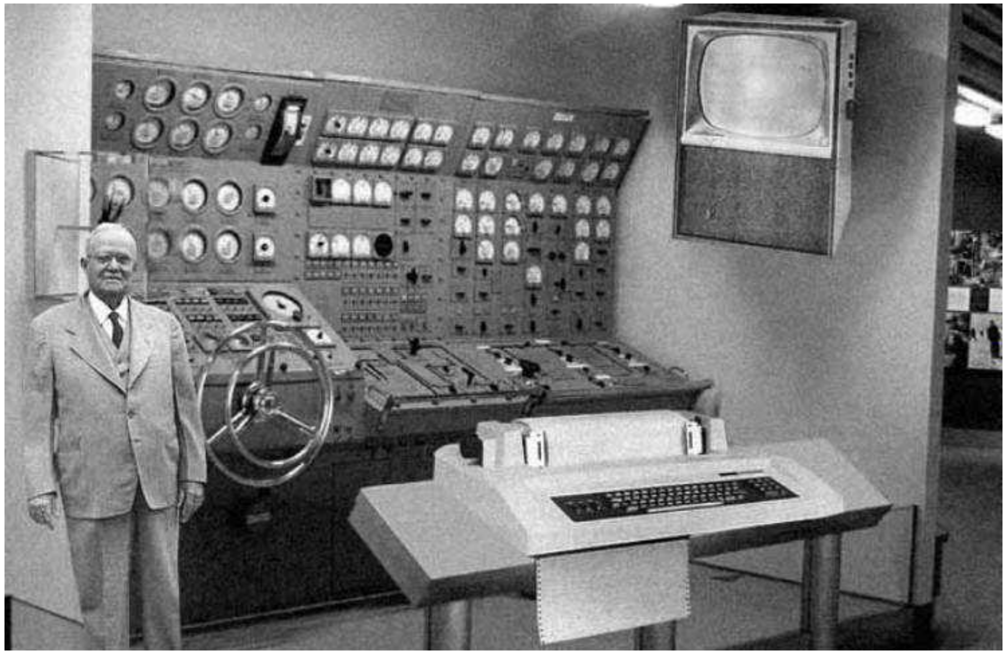


Computer Networks

EDA387/DIT661

Socket API

Part 1

Outline

- Introduction
- The Client Server (review)
- The Socket Interface
- Lab 1 - Host Resolution
- Summary
- BSD Networking History
  (extra review on your own)

Networking API

- This part review the programming of processes that communicate with each other using an application program interface (API) known as sockets.
- Some students may be very familiar with sockets already, as that model has become synonymous with network programming.
- Others may need an introduction to sockets from the ground up.

Networking API

- The objective of this sequence of lectures is to offer guidance on network programming for beginners as well as advanced programmers, for those developing new network-aware applications as well as those maintaining existing code.
- By the end of this part, your understanding of this area shall include the ability to discuss in detail how the networking components of their system function as well as how to program them.

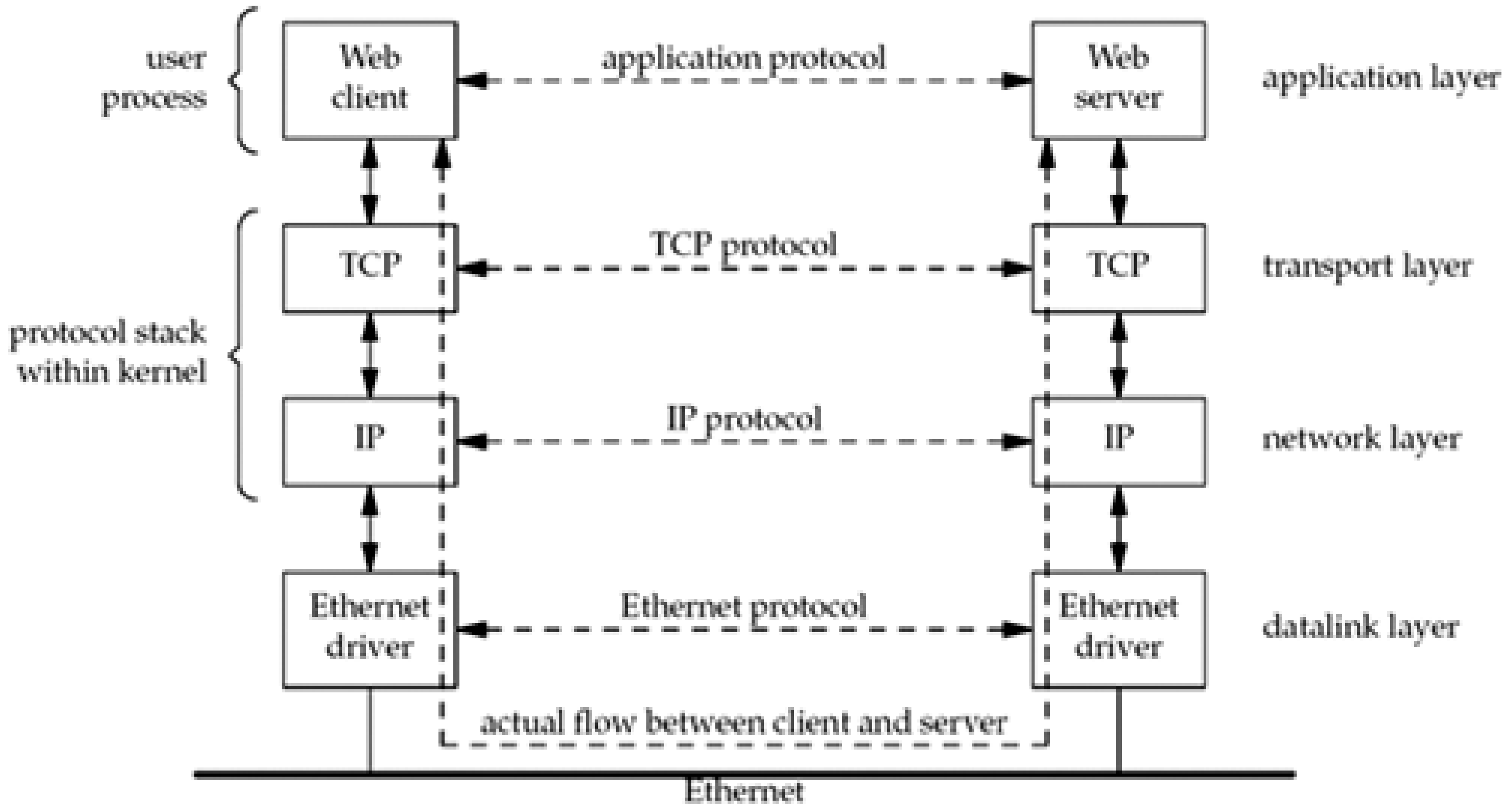
Objectives

Networking API

- We would look into several code examples that uses the sockets API.
- We focus in the usual partitioning of these network-oriented applications into client and server and write our own small examples that uses TCP Client-Server

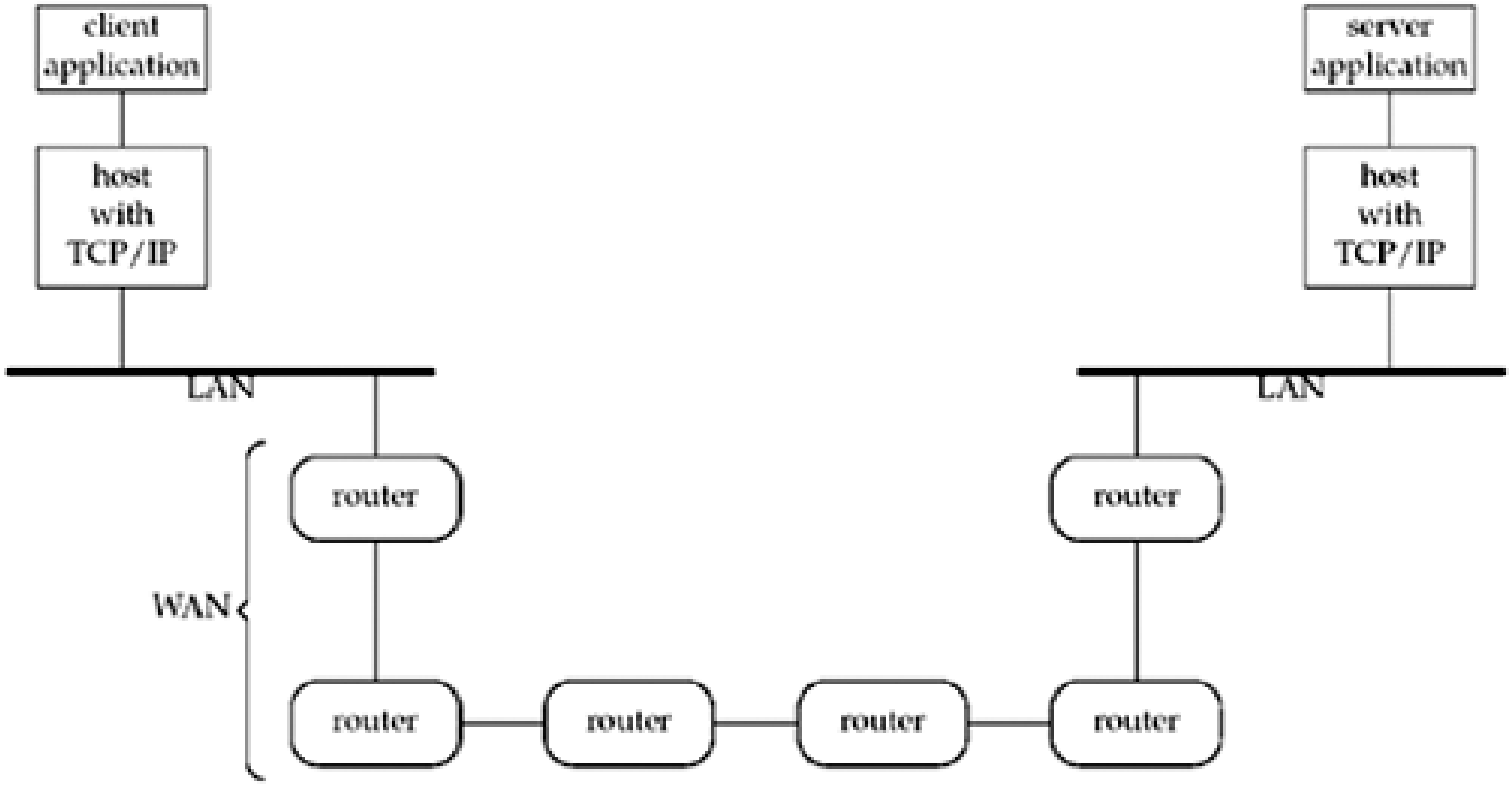
Client and Server Ethernet communication using TCP

Review



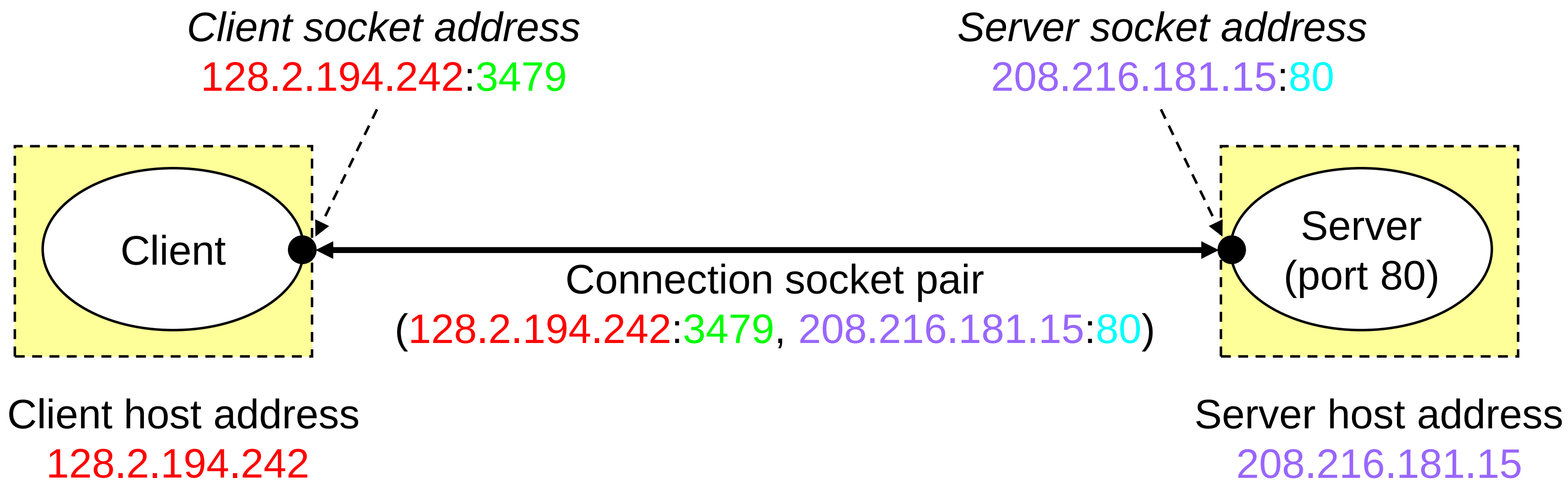
Client and Server Ethernet communication using TCP

Review



Internet Connections (TCP/IP)

- Two common paradigms for clients and servers communication
  - Datagrams (UDP protocol SOCK_DGRAM)
  - Connections (TCP protocol, SOCK_STREAM)
- Connections are point-to-point, full-duplex (2-way communication), and reliable and our main focus



Note: 3479 is an ephemeral port allocated by the kernel

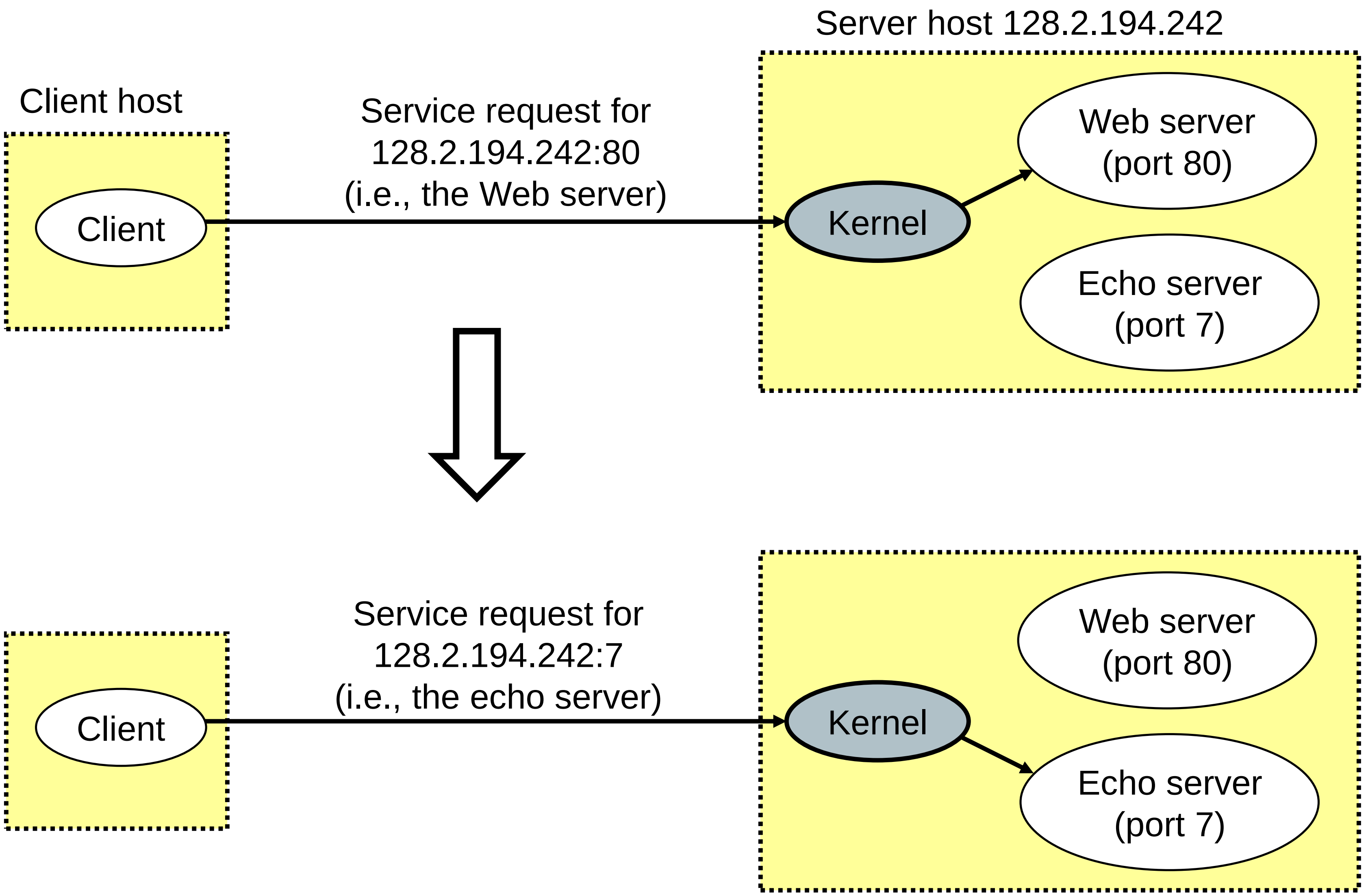
Note: 80 is a well-known port associated with Web servers

Clients

Review

- Examples of client programs
  - Web browsers, `ftp`, `telnet`, `ssh`
- How does a client find the server?
  - The IP address in the server socket address identifies the host *(more precisely, an adaptor on the host)*
  - The (well-known) port in the server socket address identifies the service, and thus implicitly identifies the server process that performs that service

Using Ports to Identify Services


Server host 128.2.194.242
Client host
Service request for 128.2.194.242:80 (i.e., the Web server)
Client
Kernel
Web server (port 80)
Echo server (port 7)
Service request for 128.2.194.242:7 (i.e., the echo server)
Client
Kernel
Web server (port 80)
Echo server (port 7)

Servers

Review

- Servers are long-running processes (daemons).
  - Created at boot-time (typically) by the init process (process 1)
  - 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
- A machine that runs a server process is also often referred to as a "server"

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

Server Examples

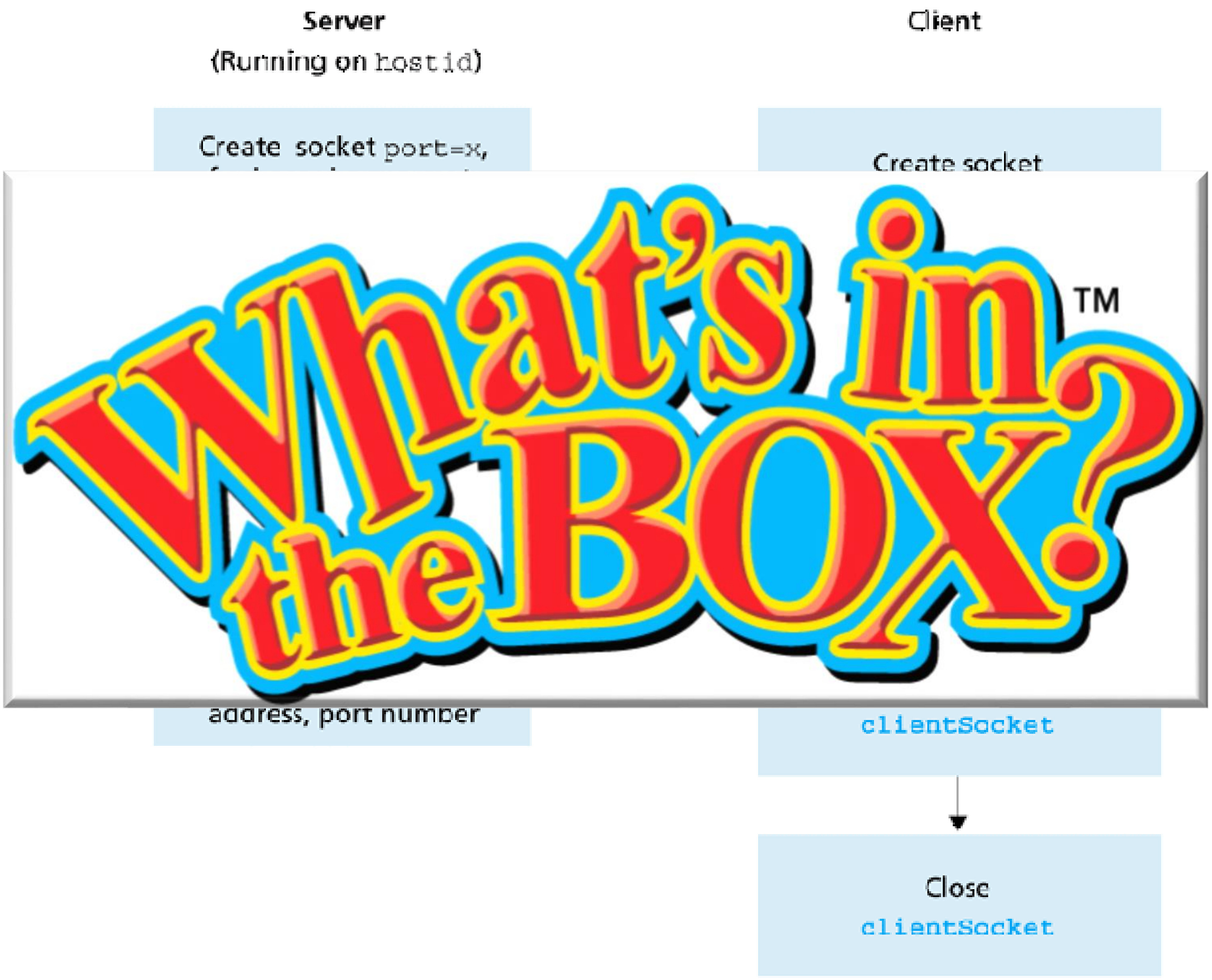
Review

- Web server (port 80)
  - Resource: files/compute cycles (CGI programs)
  - Service: retrieves files and runs CGI programs on behalf of the client
- FTP server (20, 21)
  - Resource: files
  - Service: stores and retrieve files
- Telnet server (23)
  - Resource: terminal
  - Service: proxies a terminal on the server machine
- Mail server (25)
  - Resource: email "spool" file
  - Service: stores mail messages in spool file

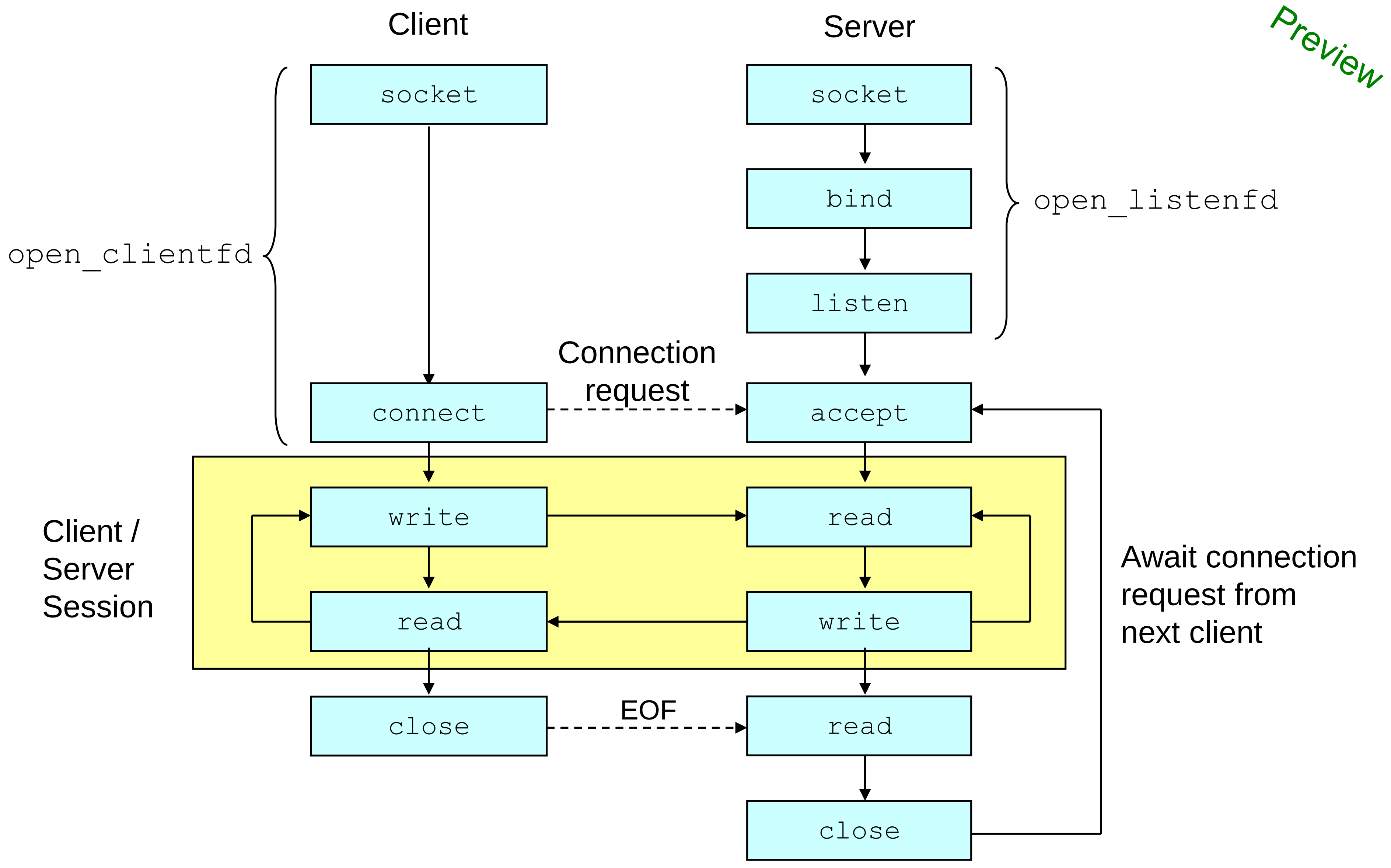
See `/etc/services` for a comprehensive list of the services available on a Linux machine.

Java View of Internet API

Review



Overview of BSD Sockets Interface



The Socket Interface

- Created in the early 80's as part of the original Berkeley distribution of Unix that contained an early version of the Internet protocols
  - Provides a user-level interface to the network
  - Underlying basis for all Internet applications
  - Based on client/server programming model

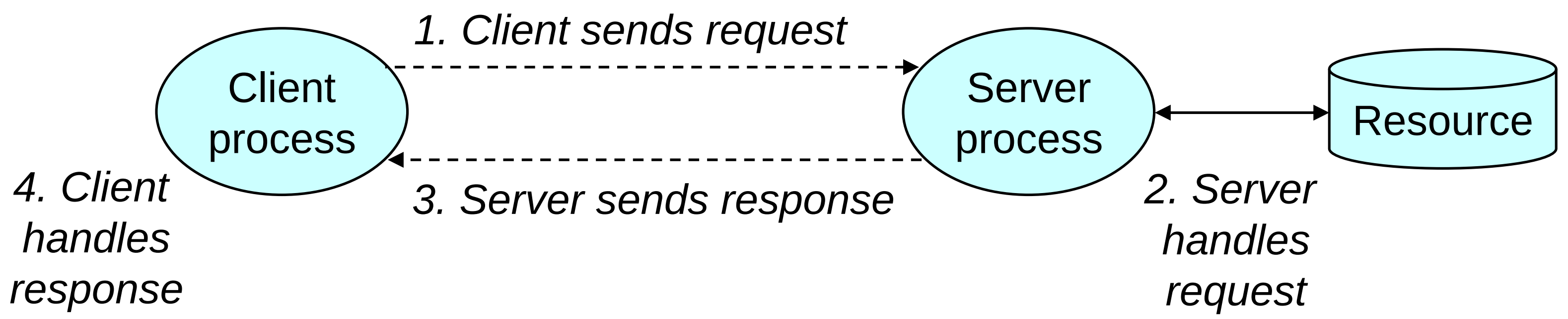
Sockets

- What is a socket?
  - To the kernel, a socket is an endpoint of communication
  - To an application, a socket is a file descriptor that lets the application read/write from/to the network
    - Remember: All Unix I/O devices are modeled as files
      - including networks
- 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

Client-Server Exchange

Review

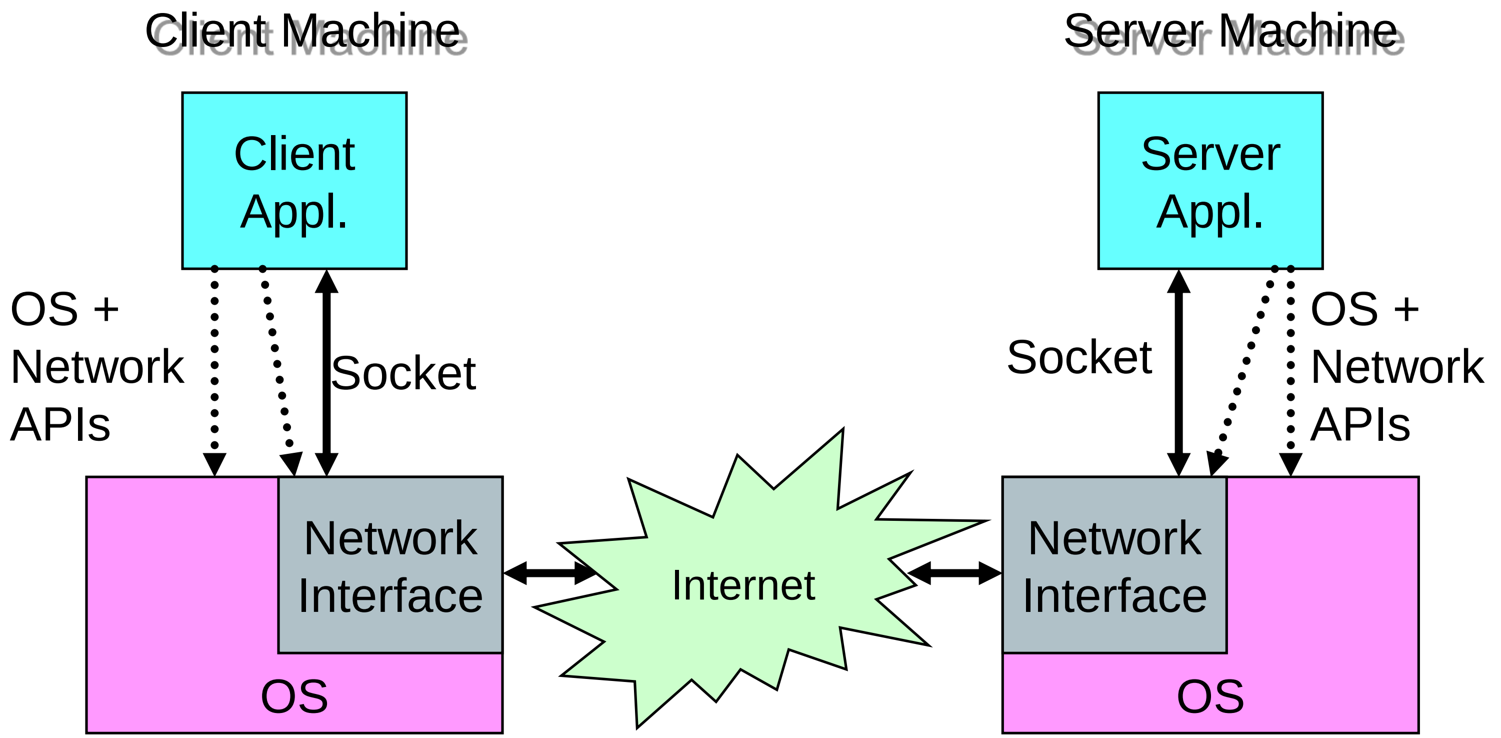
- A *server* process and one or more *client* processes
- Server manages some *resource*, e.g., information or storage
- Server provides *service* by manipulating resource for clients



Note: clients and servers are processes running on hosts (can be the same or different hosts)

Network Applications

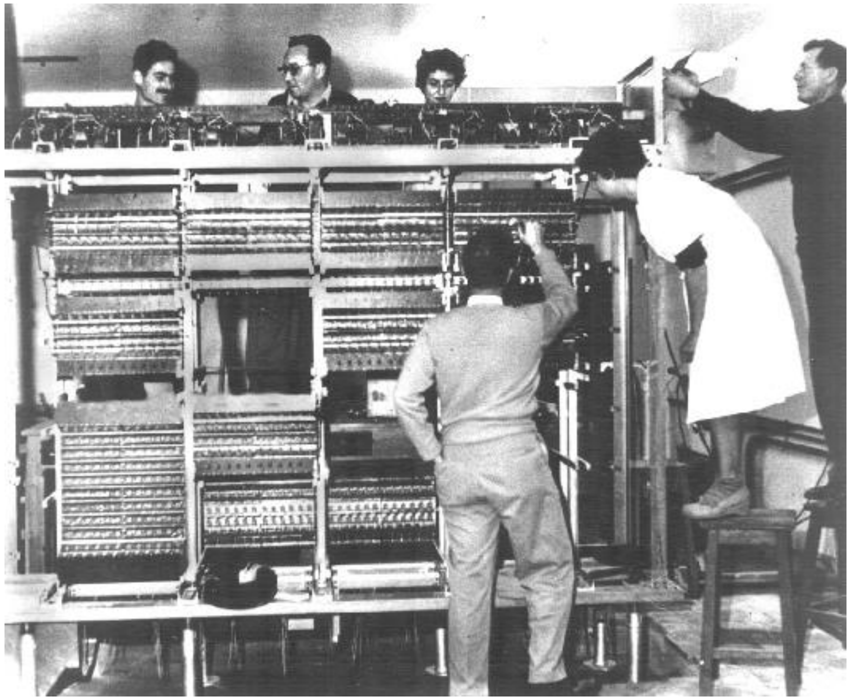
Review



- 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, segmentation

Summary

- We have reviewed client-server programing in C
- Now you are ready for the 1st programming lab



Lab 1: Host Resolution

- This is the first assignment, where you will resolve a host name into an IPv4 address.
- It serves as an short (and hopefully easy) introduction to network programming using the BSD socket API.
- The assignment includes some source code, which you will extend.
- If you want to write the program from scratch you may of course do so.

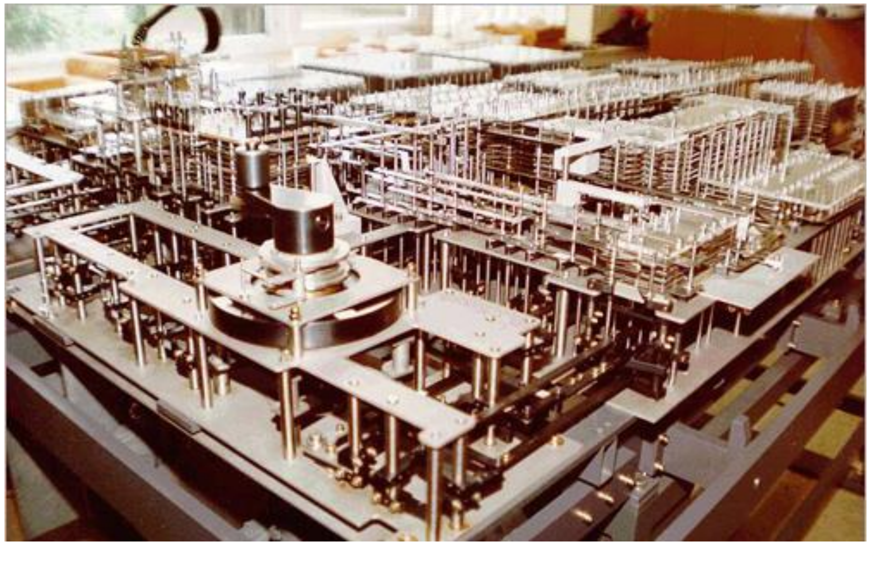
Lab 1: Host Resolution

- A reference solution is included.
  - No need to look at the solution before completing the lab.
  - If you get stuck somewhere, you can ask us during the lab
  - Or obviously refer to the solution – but avoid copying it!
- No hand in at the end of this lab.
  - But there is an quiz to solve online.

Next Lecture

We are going to look in to some code example.

We are going to look into the way that IP addresses are encoded and review our knowledge about byte ordering issues (*if time permits*).



Read more:

The Development of the C Language

http://cm.bell-labs.com/who/dmr/chist.html

Ken Thompson and Dennis Ritchie Explain UNIX

http://www.youtube.com/watch?v=JoVQTPbD6UY

http://www.youtube.com/watch?v=tc4ROCJYbm0

You are welcome to watch:

Computer Pioneers

http://www.youtube.com/watch?v=qundvme1Tik

ENIAC: The First Computer

http://www.youtube.com/watch?v=k4oGI_dNaPc

QUESTIONS?

BSD Networking History

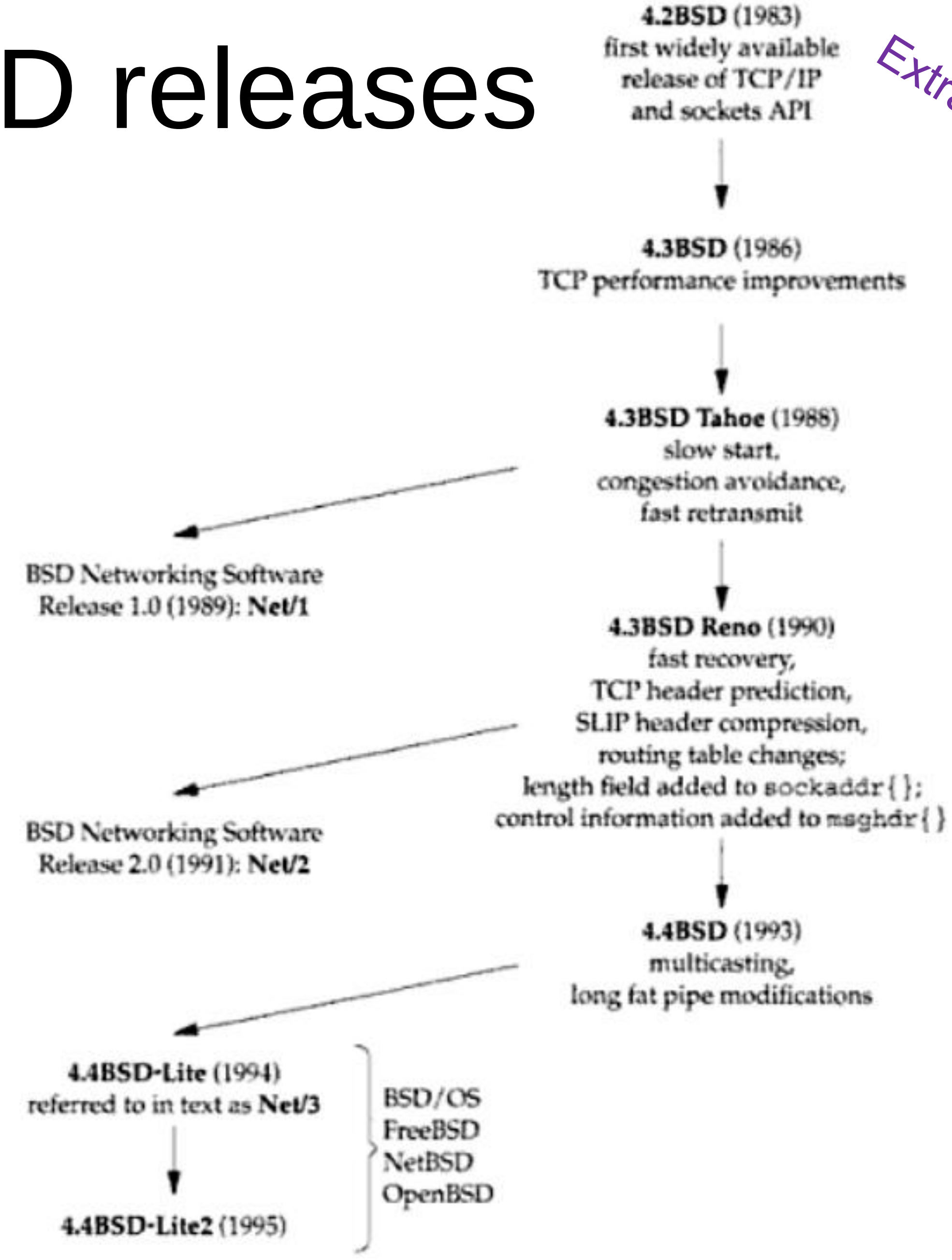
Extra

- The sockets API originated with the 4.2BSD system, released in 1983
- A few changes to the sockets API also took place in 1990 with the 4.3BSD Reno release, when the OSI protocols went into the BSD kernel
- Extra materiel: check out Kirk McKusick talk.
  - McKusick started with BSD by virtue of the fact that he shared an office at Berkeley with Bill Joy, who in essence spearheaded the beginnings of the BSD system

  http://www.youtube.com/watch?v=ds77e3aO9nA

History of BSD releases

Extra


4.2BSD (1983)
first widely available release of TCP/IP and sockets API
4.3BSD (1986)
TCP performance improvements
4.3BSD Tahoe (1988)
slow start, congestion avoidance, fast retransmit
BSD Networking Software Release 1.0 (1989): **Net/1**
4.3BSD Reno (1990)
fast recovery, TCP header prediction, SLIP header compression, routing table changes; length field added to sockaddr{}; control information added to msghdr{}
BSD Networking Software Release 2.0 (1991): **Net/2**
4.4BSD (1993)
multicasting, long fat pipe modifications
4.4BSD-Lite (1994)
referred to in text as **Net/3**
4.4BSD-Lite2 (1995)
BSD/OS
FreeBSD
NetBSD
OpenBSD

BSD Networking History

Extra

- The path down the figure from 4.2BSD through 4.4BSD shows the releases from the Computer Systems Research Group (CSRG) at Berkeley, which required the recipient to already have a source code license for Unix
- But all the networking code, both the kernel support along with the applications, were developed independently from the AT&T-derived Unix code

BSD Networking History

Extra

- Therefore, starting in 1989, Berkeley provided the first of the BSD networking releases
  - contained all the networking code and various other pieces of the BSD system that were not constrained by the Unix source code license requirement
- These releases were "publicly available" and eventually became available by anonymous FTP to anyone