UNIX Networking 1

Benjamin Brewster

Adapted from Jon Herlocker, OSU

Network Communication : IPC

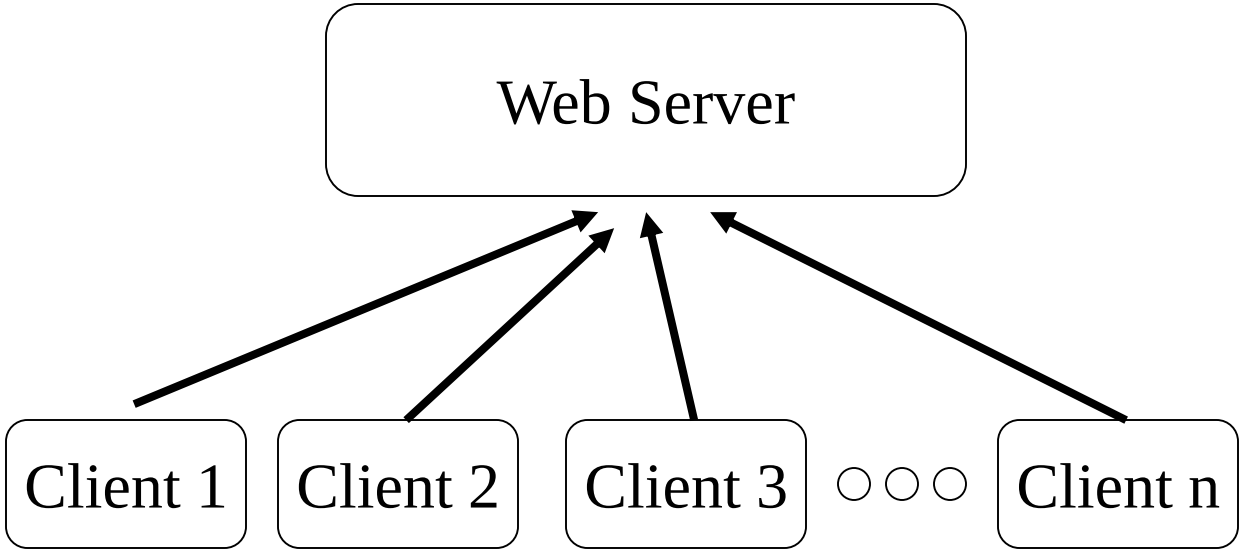
- Terminology
  - Client/Server architecture - a networking arrangement such that one process (server) is continuously waiting for new connections from other processes (clients)
  - Client - process that initiates the connection
  - Server - process that is always running, waiting for new connection initiations from client processes

UNIX Daemons – can be server

- A daemon is a process running in the background – a backgroud process
  - `syslogd` maintains the system log

- In UNIX, daemons all have the `init` process (pid == 1) as their parent

Client/Server Example: Web Server



- The web server is always running and looking for new connections
- Potentially unknown number of clients who may connect at any time
  - Netscape, IE, Lynx, etc.

TCP/IP

- TCP/IP provides communication between two processes, potentially separated by a network
- TCP/IP stands for Transmission Control Protocol / Internet Protocol
- TCP is what your application interacts with, and IP provides the addressing system for sending

What is TCP/IP

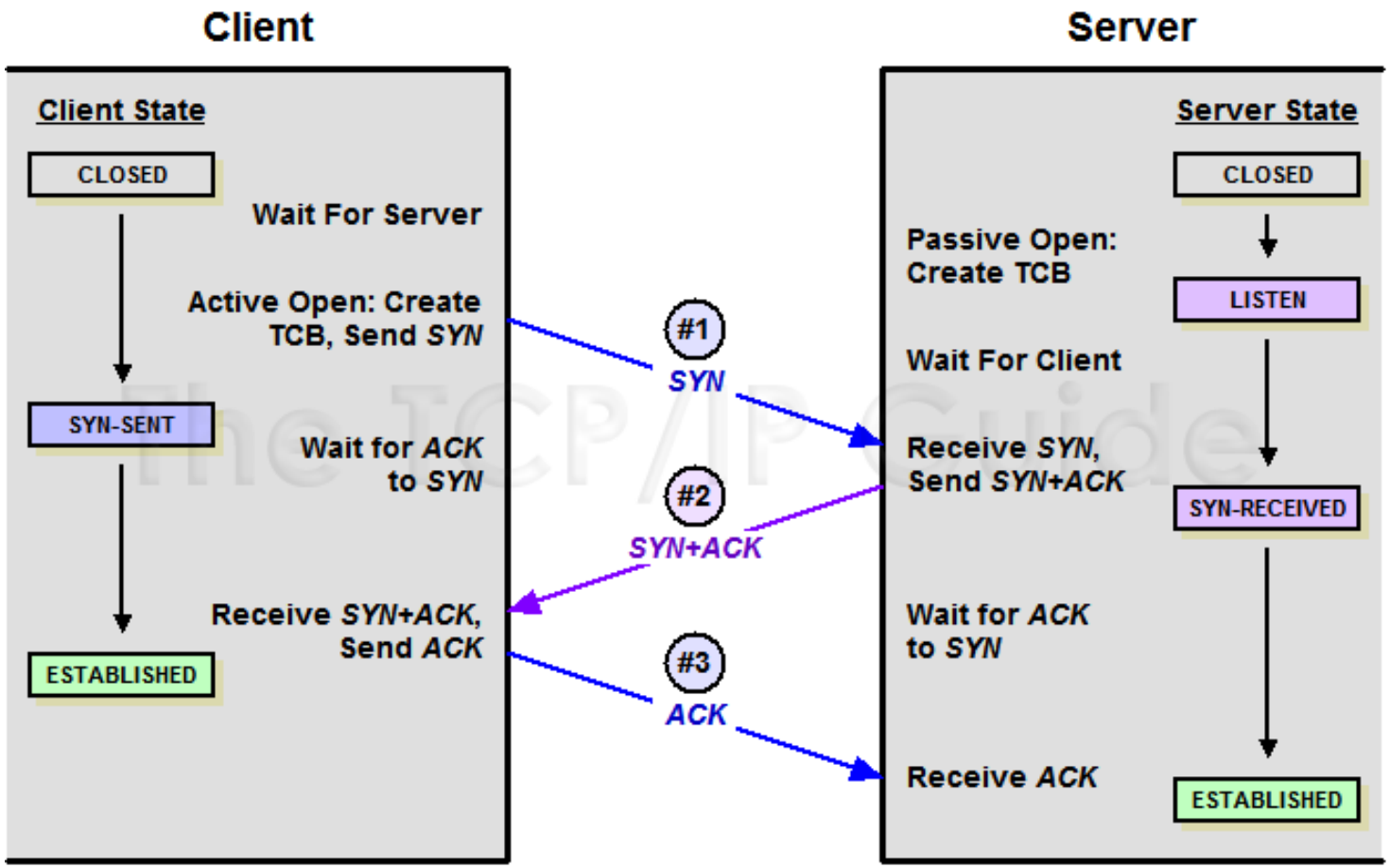
- A transmission protocol
- Provided by the OS to user processes
- Only passes bytes between processes
  - It does not interpret those bytes
- You need an "application protocol" on top of TCP/IP which defines what the bytes mean
  - This application protocol is what your program does with the bytes it receives

Application Protocols

- TCP/IP is like a phone connection
  - A phone transfers sound between two people
  - A phone does not interpret the meaning of the sound

- The telephone application protocol (A calling B)
  - B says "hello" (First step)
  - A responds "is so-and-so available" (Second Step)
  - B responds "yes", "no", or "speaking" (Third Step)
  - If B is the desired person, then begin talking
  - If B is not available, then leave message and hang up

TCP Handshaking



From tcpipguide.com, fetched 2/18/2015

Example: Web server application protocol

- HTTP - hyper text transfer protocol
- Standardized protocol defining:
  - how client requests are formatted
  - how server responses are formatted
- The protocol is text-based
  - All requests, responses, and errors
- Relatively simple protocol
  - Primarily used to support one feature: requesting a file to be downloaded

HTTP Protocol

- Simple request mode
  - Client connects to server then sends:
    - GET abc.html
  - Server receives and parses the get command and returns the file requested

- Enhanced request mode
  - Client connects to the server then sends:
    - GET index.html HTTP/1.0
  - Server responds with some header information, such as server type and version, then a blank line, then the data file

Text Protocol Debugging Tool

- `telnet` helps debug text-based protocols
- We're not using telnet here for shell access – OSU requires that we use only the SSH protocol
- You can pass telnet a second parameter that specifies the port you want to connect to
  - To connect to a web server, we typically use port 80
    - `telnet eecs.oregonstate.edu 80`

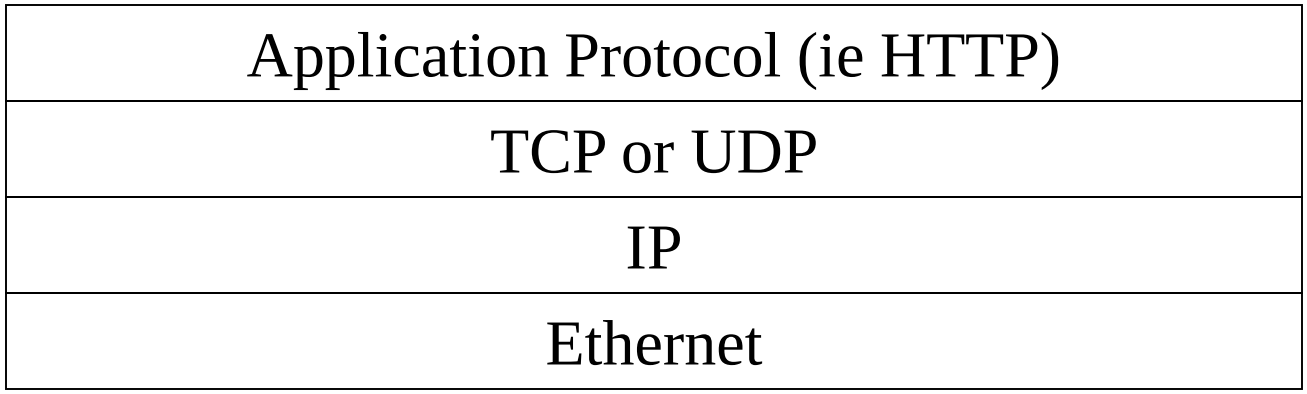
Demo of telnet and HTTP

1. `% telnet eecs.oregonstate.edu 80`
2. `<web server> GET / HTTP/1.1`
3. `<web server> Host: eecs.oregonstate.edu`
4. `<web server> (Enter twice)`

Non-text-based Application Protocols

- Not all application protocols are text-based
- TCP/IP has no problem transferring binary data
- Advantages of text-based protocols
  - Easy to debug
  - Easy to communicate and understand (and teach!)
- Disadvantages of text-based protocols
  - Not very compact or efficient
  - Server can spend a lot of time just parsing text
    - Very important for text protocols to be simple!

Internet network layers



- TCP - Transmission Control Protocol - connection-oriented, guaranteed data transport
- UDP - Universal Datagram Protocol - connectionless, not guaranteed
- IP - Internet Protocol - naming, addressing and routing, independent of physical connection type
- Ethernet - the addressing and signaling protocol used across a copper cable connecting a collection of machines

TCP versus IP

- Once the TCP layer sends out packets, its up to IP to get them where they are going
- Problem: IP does not guarantee
  - Data integrity
  - Packet order
  - Prevention of duplicates
  - Packet will actually arrive

TCP versus IP

- TCP, however, can detect if IP is having trouble, and can
  - Re-order packets
  - Request packet re-transmission
  - Drop duplicate packets

TCP

- Most commonly used protocol for transferring information
- Provides a byte stream interface (like stdio)
  - Data is guaranteed to not be lost, even over networks where packets can be dropped
  - Data arrives in the order it was sent
  - Connection oriented - each side of the connection maintains resources to keep the connection open until it is explicitly closed
  - A TCP connection is bi-directional - traffic can be sent across the connection in either direction

UDP

- Used less frequently than TCP
- Provides a very different interface
  - Connectionless (No handshaking, etc.)
  - Data is broken into packets called datagrams
  - Server does not remember clients between datagrams
  - Datagrams may be dropped by the network
  - Datagrams may arrive out of order

TCP Benefits

- Benefits of TCP (over UDP)
  - Error-free data transfer
  - Ordered-data transfer
  - Retransmission of lost packets
  - Discarding duplicate packets
  - Congestion throttling

UDP Benefits

- UDP has much less overhead than TCP
- When to use UDP
  - Streaming video/audio
  - Mass broadcasting
  - Asynchronous communication
    - GAMEZ

Internet Protocol (IP)

- IP specifies:
  - How we address machines on the network
  - If the machine we are addressing is not on the same local network: how the data is routed
- Each network interface (network card) has an IP address, which must be unique
- IP(v4) address are 32 bit numbers, but are usually *represented* as four one-byte numbers, separated by periods:
  - Ex: 128.101.34.200
  - Ex: 128.193.55.70

IP 4 vs. 6

- IP version 4 (IPv4) uses 32-bit addresses, and therefore has 2^{32} = 4,294,967,296 = $4*10^9$ unique addresses
  - This is not enough for the world - in my house alone:
    - Xbox One, Xbox 360, my laptop, wife's laptop, my phone, wife's phone, printer
  - A lot of this can be handled by routers, which create your own LAN, but this shows the problem

IP 4 vs. 6

- IPv6 uses 128-bit addresses
  - $2^{128} > 2^{32}$
  - $3.4 * 10^{38} > 4.2 * 10^{9}$
  - This is **50 octillion** addresses for **each** of 6.5 billion people on earth

50 octillion addys*:

50,000,000,000,000,000,000,000,000,000