

Network Fundamentals

Benjamin Brewster

Except as noted, all images copyrighted with Creative Commons licenses,
with attributions given whenever available

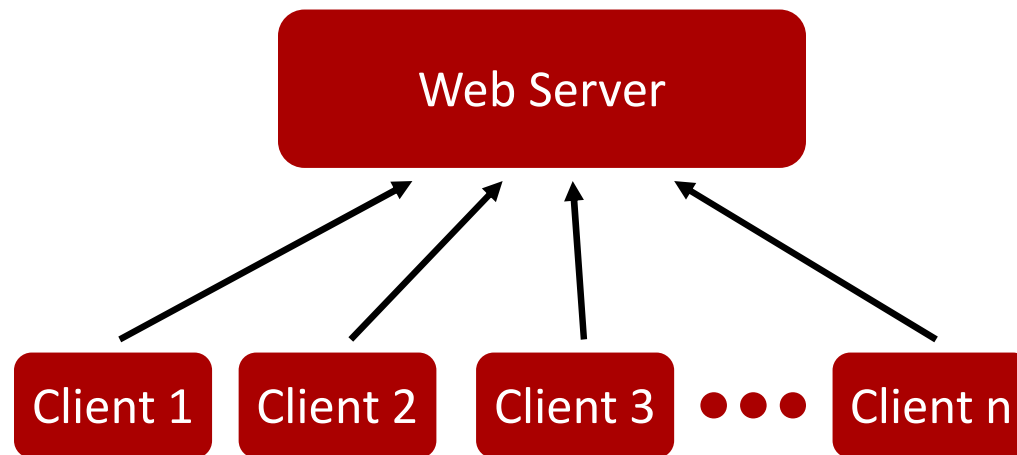
IPC via Network Communication



- 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, requesting a service
- **Server** :: process that is always running, waiting for new connections from client processes

Client/Server Example: Web Server



- The web server is always running and looking for new connections
- Potentially unknown number of clients may connect at any time
 - Chrome, Edge, Safari, IE, Firefox, Opera, Lynx, etc.

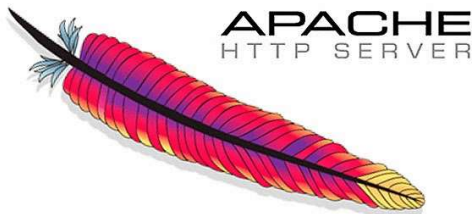
UNIX Daemons - a Type of Server

- In UNIX, a daemon is a process running in the background, ready to provide a service to the programs that need it
 - `syslogd` (system log daemon) maintains the system log
 - `lpd` (line printer daemon) manages print spooling
 - `ntpd` (network time protocol daemon) manages clock sync on a network
 - `dhcpd` (dynamic host control protocol daemon) assigns TCP/IP configuration data to network clients that request it
- In UNIX, daemons all have the `init` process (`pid == 1`) as their parent
 - As of 2016, many Linux distributions have replaced `init` with `systemd`



Example Protocol: HTTP

- **HTTP** - Hyper Text Transfer Protocol
- Relatively simple: primarily used to support one feature, the requesting of a file to be downloaded
- Standardized protocol defining:
 - How client requests are formatted
 - How server responses are formatted
- The protocol is text-based, including all requests, responses, and errors



The world's most widely used web server is Apache.
Don't confuse this software with the HTTP protocol
that the server primarily deals with!
Apache's UNIX daemon name is `httpd`

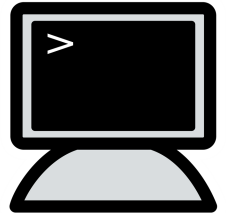
Example Protocol: HTTP

- Simple request mode
 - Client connects to server then sends:
 - GET abc.html
 - Server receives and parses the GET command, then returns the file requested
- Enhanced request mode
 - Client connects to the server then sends:
 - GET index.html HTTP/1.1
 - Server responds with some header information, such as server type and version, then a blank line, and *then* the data file



Text Protocol Debugging Tool

- `telnet` helps debug text-based protocols, but was originally designed for interacting with text-based protocol network sessions
- Almost without exception now, servers do not use telnet for shell access, as all text is passed in plain text - it can be read by every node in between the client and server
 - SSH is the current standard, which encrypts transmitted data
- You can pass telnet a second parameter that specifies the port you want to connect to



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 again)`

To connect to a web server,
we typically use port 80



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!



Network Layer Model - Application

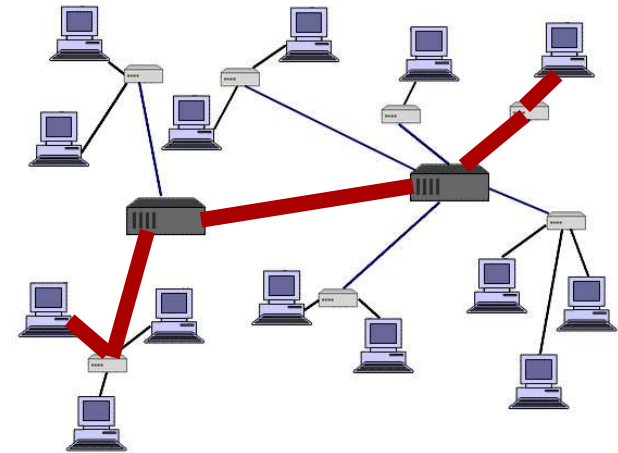
Layer	Common Protocols
Application	HTTP
Transport	TCP and UDP
Network	IP
Link	Ethernet, 802.11
Physical	Twisted pair copper, radio, fiber



- **Application:** Your software, utilizing the network
 - Agnostic to the underlying methods that make the data go from one host to the other
 - Web browsers, games, IM clients, video chat, email, etc.
 - Uses `send()` and `recv()`, for example

Network Layer Model - Transport

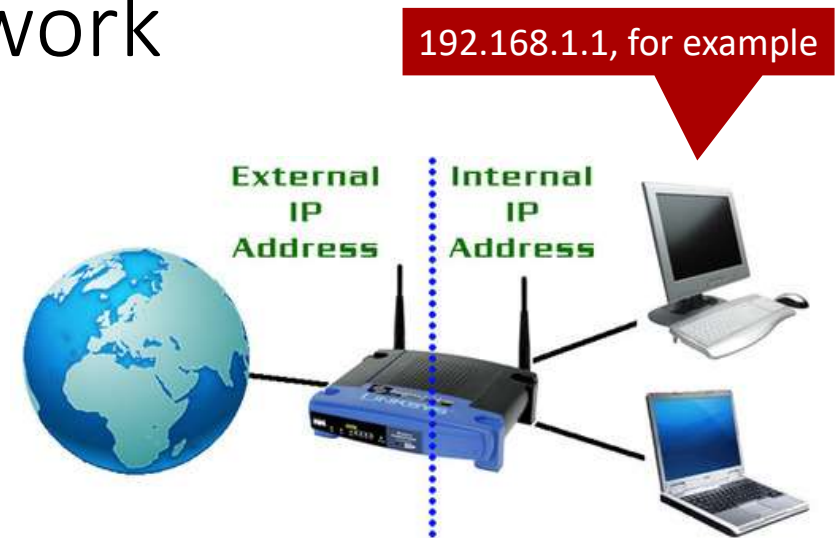
Layer	Common Protocols
Application	HTTP
Transport	TCP and UDP
Network	IP
Link	Ethernet, 802.11
Physical	Twisted pair copper, radio, fiber



- **Transport:** Protocols that control how data is sent from one host all the way to the other, irrespective of the number of nodes, hops, or networks the data passes through
- TCP: Transmission Control Protocol - connection-oriented, guaranteed, in-order data transport
- UDP: Universal Datagram Protocol - connectionless, not guaranteed

Network Layer Model - Network

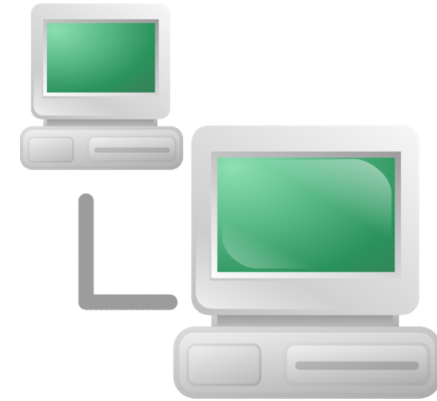
Layer	Common Protocols
Application	HTTP
Transport	TCP and UDP
Network	IP
Link	Ethernet, 802.11
Physical	Twisted pair copper, radio, fiber



- **Network:** Addressing and organization of a particular set of connected hosts (called a network), defines addressing between networks
- IP: Internet Protocol - naming, addressing and routing from host to host and across networks
- Still independent of physical connection type

Network Layer Model - Link

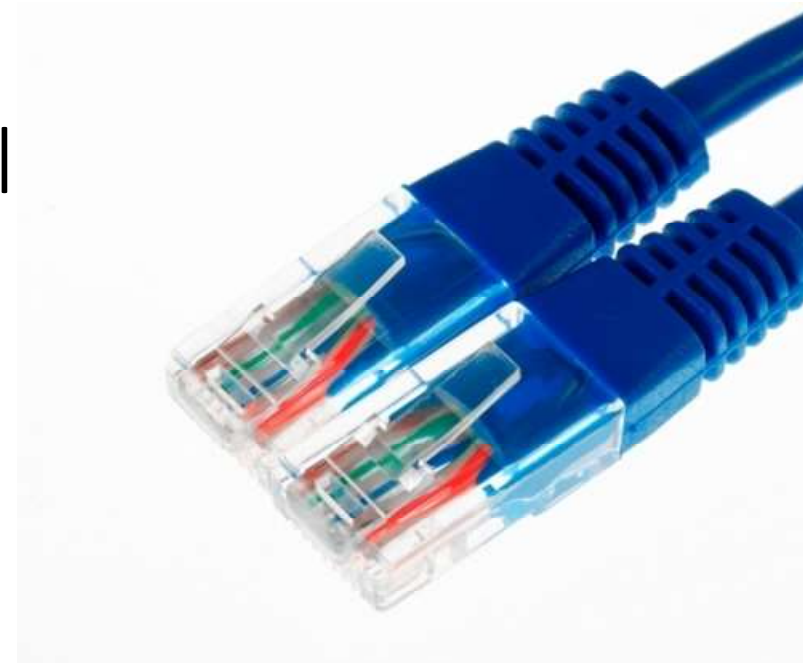
Layer	Common Protocols
Application	HTTP
Transport	TCP and UDP
Network	IP
Link	Ethernet, 802.11
Physical	Twisted pair copper, radio, fiber



- **Link:** Concerned with getting data from just one node to the next neighboring node; does no planning of routes
- Ethernet: The de facto addressing and signaling protocol currently in use in modern networks; uses Media Access Control addresses to communicate, together with IP
- 802.11: aka Wi-fi, the de facto protocol for wireless communication; controls sharing of the congested transmission space and connection speeds based on quality

Network Layer Model - Physical

Layer	Common Protocols
Application	HTTP
Transport	TCP and UDP
Network	IP
Link	Ethernet, 802.11
Physical	Twisted pair copper, radio, fiber



- **Physical:** The actual hardware used to enable two hosts to talk
- Copper: Standard category 5 and 6 network cables (4 pairs of twisted copper strands) connect most of the world
- Radio: Enables wireless devices to communicate



TCP/IP High-Level Functionality

- The combination of TCP and IP provides communication between two processes, potentially separated by a network
- TCP/IP stands for **T**ransmission **C**ontrol **P**rotocol / **I**nternet **P**rotocol
- TCP is the protocol that your application interacts with, while IP provides the addressing system for routing network packets



TCP Details

- TCP is the most commonly used protocol for transferring information across a network; second-most common Transport protocol is UDP
- Provides a byte stream interface (like stdio)
- 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
- Provides controls to slow down the sender if the nodes in the path to the receiver are burdened with traffic or otherwise lossy

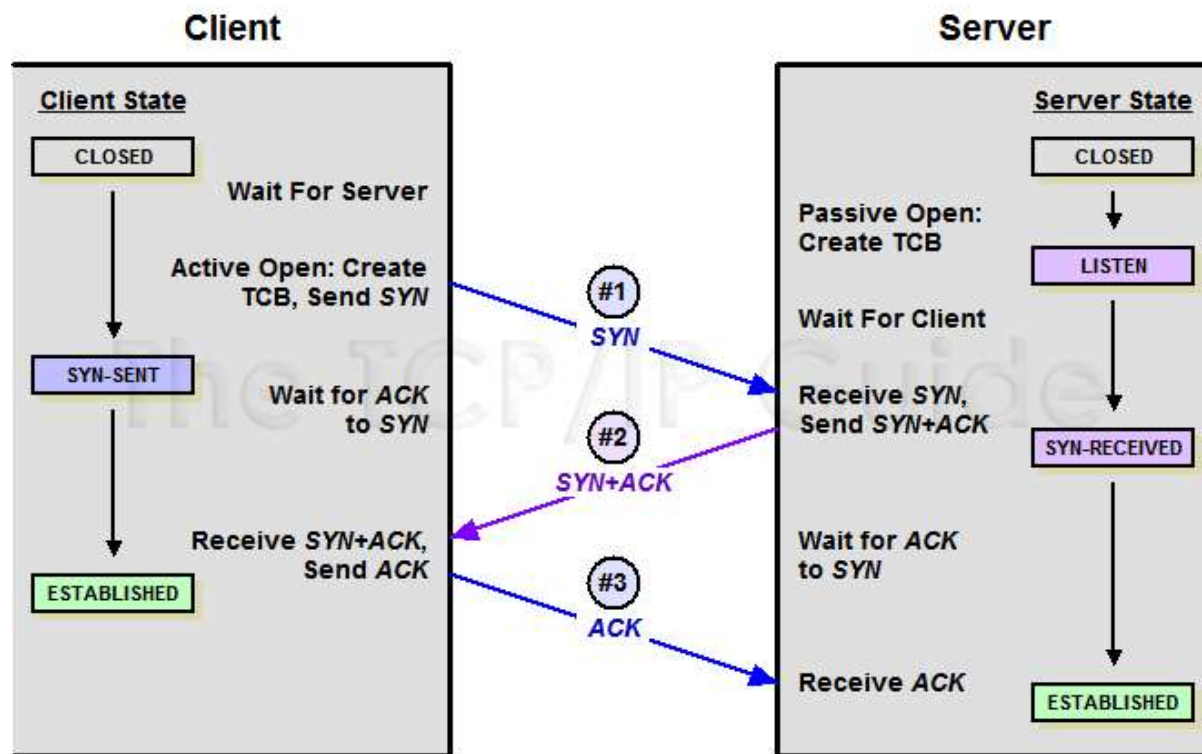


Applications On Top of TCP/IP

- TCP/IP only passes bytes between processes - it does not interpret those bytes
- You write an "application protocol" on top of TCP/IP which defines what the bytes mean: this is what your program does with the bytes it sends & receives
- 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 (Bob calling Alice):
 1. Bob calls Alice's phone
 2. Alice answers and says "Hello"
 3. Bob says "Hi" back, and then they both exchange information



TCP Handshaking Starts the Connection



From tcpipguide.com, fetched 2/18/2015

TCP Comparison with IP

- TCP sends out network traffic organized into bundles called packets, using the addresses specified and organized by IP
- Problem: IP does not guarantee
 - Data integrity
 - Packet order
 - Prevention of duplicates
 - Packet will actually arrive
- TCP can detect if the integrity of the stream of packets encounters issues and will take steps such as these to keep things running:
 - Re-order packets
 - Request packet re-transmission
 - Drop duplicate packets




UDP - User Datagram Protocol

- 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
- UDP has *much* less overhead than TCP, and is much faster to transfer data
- When to use UDP
 - Streaming video/audio
 - Mass broadcasting
 - Asynchronous communication like that used in games



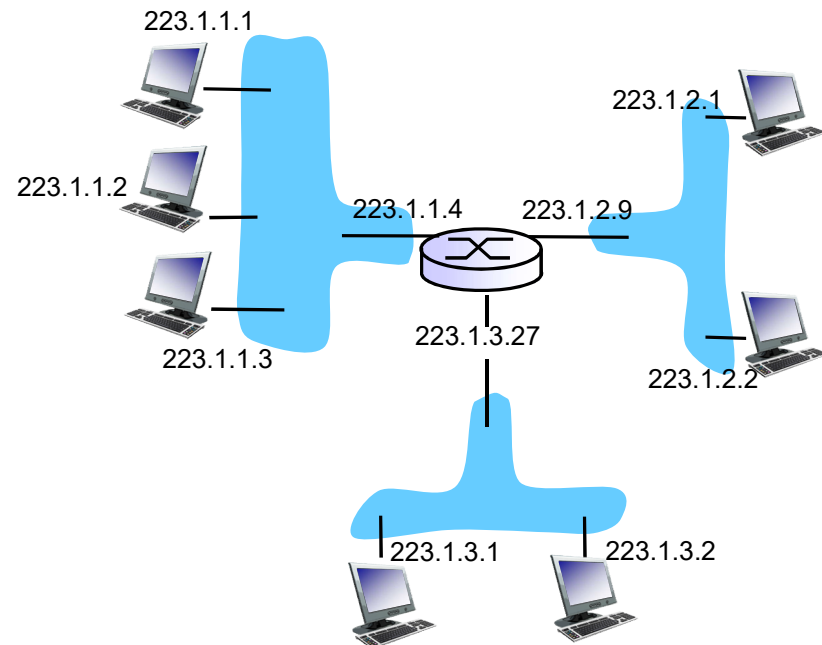
Internet Protocol Details

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


$$223.1.1.1 = \underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$$


IP Addressing: Introduction

- **Interface:** connection between host/router and physical link
 - Routers have multiple interfaces
 - A host typically has one or maybe two interfaces (e.g., wired Ethernet, wireless 802.11)
- *Each* interface has its own IP address; thus devices with more than one interface control multiple addresses



223.1.1.1 = $\underbrace{11011111}_{223} \underbrace{00000001}_1 \underbrace{00000001}_1 \underbrace{00000001}_1$



IP 4 vs. 6



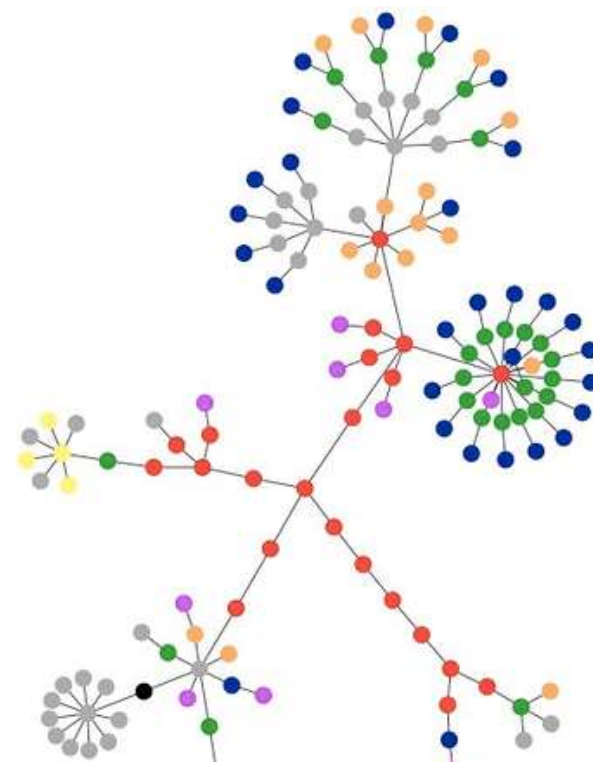
- IP version 4 (IPv4) uses 32-bit addresses, and therefore has $2^{32} = 4,294,967,296 = 4 \times 10^9$ unique addresses
- This is not enough for the world; in my house alone:
 - Xbox One, Xbox 360, five computers, 3 cell phones, amplifier, Blu-ray player, printer
- A lot of this can be handled by routers, which creates your own private LAN with addresses hidden from the outside networks, but this only shines light on the problem
- There are no more IPv4 addresses to give out for the Internet!
- Internet of Things (IoT), always-on connections, and inefficient address allocation all contributed to usage

Companies/orgs with IPv4 /8 blocks from Internet Assigned Numbers Authority (IANA, a dept. of ICANN)

Owner	/8 Blocks	~IP addresses
US Military (Department of Defense etc.)	12	201 million
Level 3 Communications, Inc.	2	33 million
Hewlett-Packard	2	33 million
AT&T Bell Laboratories (Alcatel-Lucent)	1	16 million
AT&T Global Network Services	1	16 million
Bell-Northern Research (Nortel Networks)	1	16 million
Amateur Radio Digital Communications	1	16 million
Apple Computer Inc.	1	16 million
Cap Debis CCS (Mercedes-Benz)	1	16 million
Computer Sciences Corporation	1	16 million
Department of Social Security of UK	1	16 million
E.I. duPont de Nemours and Co., Inc.	1	16 million
Eli Lilly and Company	1	16 million
Ford Motor Company	1	16 million
General Electric Company	1	16 million
Halliburton Company	1	16 million
IBM	1	16 million
Interop Show Network	1	16 million
Merck and Co., Inc.	1	16 million
MERIT Computer Network	1	16 million
Massachusetts Institute of Technology	1	16 million
Performance Systems International (Cogent)	1	16 million
Prudential Equity Group, LLC	1	16 million
Société Internationale De Telecommunications Aero.	1	16 million
U.S. Postal Service	1	16 million
UK Ministry of Defence	1	16 million
Xerox Corporation	1	16 million

<http://royal.pingdom.com/2008/02/13/where-did-all-the-ip-numbers-go-the-us-department-of-defense-has-them/>

Who's Got Those Addresses?



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*:

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

*Theoretically: a lot of them are reserved for special purposes, so the number is lower



Back to Just Our Process

- A process can use just one interface to communicate with itself and other processes on the same machine
- Address network transmissions to your interface's own IP address, or the special hostname "localhost"
- Indicate which process you're talking to by specifying the "port" (more on this next lecture)

