Contents

0 Preamble

This guide was adapted from a slide deck based on *Computer Networking: A Top-Down Approach*. Thank you James F. Kurose and Keith W. Ross for the slides, and thank you Igor for presenting this material!

1 Part One: The Anatomy of the Internet

The internet is composed of

1. Billions of connected computing devices, which are hosts which run apps found on the "edge" of the internet
2. Packet switches, which forward chunks of data called packets. There are routers and there are switches
3. Communication links, which include fiberoptic wire, copper wire, radio signal, satellite, all of which have their own transmission rate
4. Networks, which are collections of devices, routers, and links managed by an organization

Consider the word 'internet', which has it in the name: inter-network: a network of networks! Internet service providers (ISPs) are responsible for upkeep. Protocols define the rules that devices use to communicate with each other. They include Hypertext Transfer Protocol (HTTP), TCP, IP, WiFi, 4/5G, and Ethernet.

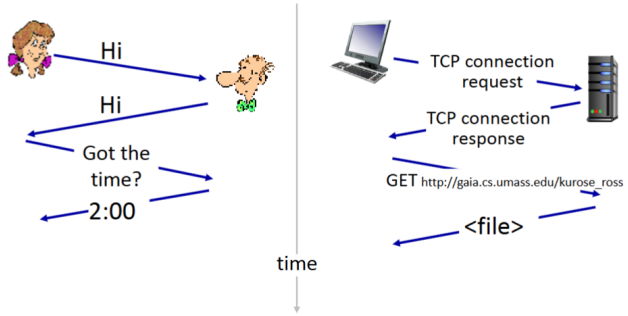
The internet also uses standards. Request for Comments is a document of internet standards and Internet Engineering Task Force is an organization responsible for up-keeping internet standards.

The internet works hand in hand with software, providing services to applications like web, streaming video, multimedia teleconferencing, email, games, e-commerce, social media, and appliance functionality. For software to communicate with these internet-based devices, features, and functionality, Application Programming Interfaces (APIs) are written to provide "hooks" to allow software to send and receive information from the internet.

1.1 Protocols

Internet protocols define how different parts of the internet interact.

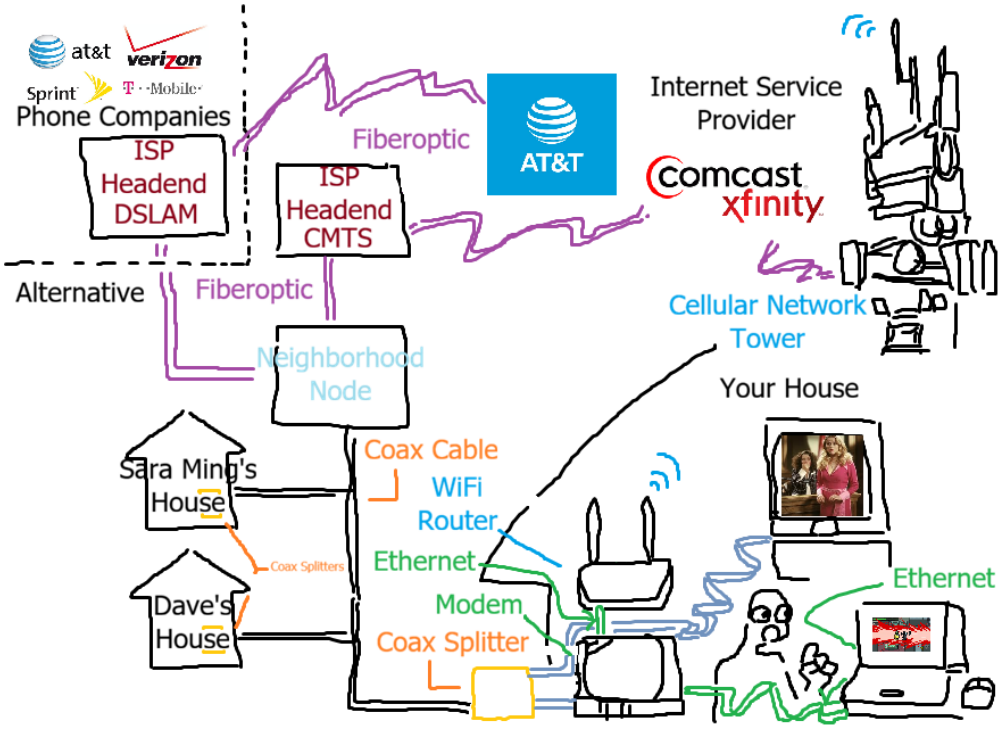
Replace with paint artwork



The Internet is composed of three distinct regions. At the edge of the network is the, well, **Network Edge**, where hosts, including clients and servers (often in data centers) lie. Think phones, cars, fridges, traffic lights, and

toasters that toast the weather forecast onto your toast. These communicate with **Physical Access Networks**, which include the wired and wireless communication links that transport the data to the hosts. Think network routers and cell towers. At the very center of it all is the **Network Core**. This is an network of networks composed of routers that include national, global, local, and regional ISPs.

1.2 Access Networks



1.2.1 Cable-Based Access

Individuals are provided precious internet access via a WiFi router, which is connected to a modem, which takes signal from a coax cable (potentially split so that the TV can get a coax line). This coax cable is connected (sometimes through a neighborhood node) to an ISP Headend. The headend is connected to the ISP via fiberoptic. By using Frequency Division Muliplexing (FDM), different channels can be transmitted via the same cable by using different frequency bands. Which is where the coax splitter comes in. In a household without a TV connection via coax (which is becoming increasingly more common due to internet TV), a splitter isn't required.

Important note: While the neighborhood fiber node and the ISP are connected via fiberoptic cable, homes that aren't specifically wired with fiberoptic are connected to the neighborhood fiber node via coaxial cable. These providers can technically claim that their internet service is fiberoptic, but the distinction between true fiberoptic and HFC is important to note.

Hybrid Fiber Coax (HFC) has an asymmetric transmission rate: up to 40 Mbps-1.2 Gbps downstream, and 30-100 Mbps upstream.

1.2.2 Digital Subscriber Line

Digital subscriber lines (DSLs) act similarly to HFC, but use existing telephone lines to a central office Digital Subscriber Line Access Multiplexer (DSLAM) to connect to the internet. These lines are able to transmit both telephone signal and internet signal, the redirection of which is handled by the DSLAM. These have 24-52 Mpbs dedicated downstream transmission rate and 3.5-16 Mbps dedicated transmission rate.

1.2.3 Wireless Networks

Wireless Networks include WiFi routers and cellular network towers. WiFi routers are wireless local area networks which are typically within 100 feet of a building and can range from 11 Mbps to 450 Mbps. Cellular network towers can reach up to 10km and have around tens of Mpbs.

1.2.4 Enterprise Networks

Enterprise networks are used by companies, universities, and other large institutions that benefit from having the people within them connected to a centralized internet. These utilize a mix of wired, wireless, routing, and switching technologies to provide Ethernet and WiFi. Etherneed speeds range from 100 Mbps to 10 Gbps. WiFi speeds range from 11 to 450 Mpbs.

1.2.5 Data Center Networks

Data center networks have high-bandwith links that range from 10-200 Gbps speeds to handle connecting hundreds to thousands of servers together and to the internet.

1.2.6 Data Transmission

To communicate with the internet, hosts send messages out as packets. Messages are first split into a number of packets of length L. Those packets are then transmitted into an access network at a transmission rate R, called a bandwidth.

$$\text{packet transmission delay} = \frac{L \text{ bits}}{R \text{ bits/second}}$$

Notice how the units cancel out: $\dfrac{L \text{ bits}}{R \text{ bits/second}} = L \text{ bits} * R^{-1} \dfrac{\text{seconds}}{\text{bit}} = \text{delay } (\textit{seconds})$

1.2.7 Vocabulary

Bits propagate between transmitter and receiver pairs

Physical links lie between transmitters and receivers

Guided media are signals propagated through solid mediums, i.e. copper, fiber, coax.

Fiberoptic cables have the fastest transmission rates out of all the physical internet connections that exist. These connect internet service providers to headends and what lie at the seafloor to communicate overseas.

The next physical media found in the Access Network sequence is the coaxial cable. These are composed of two concentric copper conductors that allow for bidirectional transmission that support broadband; multiple frequency channels on cable, with up to 100 Mbps per channel. These are the cables that connect the headends (or neighborhood nodes) to the houses, sending signal that will be received by a modem.

Lastly is the Twisted Pair (TP), which is composed of two insulated copper wires twisted together. Category 5 Twisted Pairs allow for 100 Mbps to 1 Gbps Ethernet while category 6 reaches 10 Gbps Ethernet. These are cables that connect devices to the modem for Ethernet.

Unguided media are signals that propagate freely.

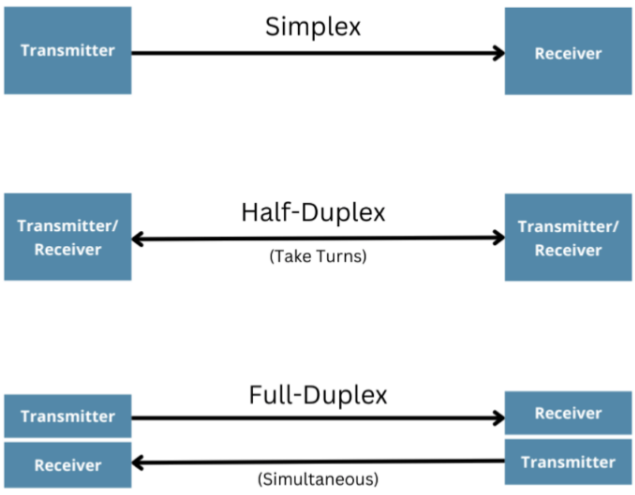
Radio waves carry signals in various bands of the electromagnetic spectrum. There is no physical wire and the broadcast is half-duplex. These are affected by the environment and get reflected, obstructed, and interfered with.

Radio link types include

1. Bluetooth: Replaces wired connections but only reaches around 10 to 20 meters
2. Wireless LAN (WiFi): 10-100s Mbps; 10s of meters
3. Wide-Area (i.e. 4G/5G cellular): 10s Mbps (4G) over ∼10km
4. Terrestrial Microwave: Point-to-point; 45 Mbps channels
5. Satellite: Up to 100 Mbps (Starlink) downlink; 270 millisecond end-to-end delay (geostationary)

1.2.8 Connection Types

Replace with paint artwork



1.3 The Network Core

The network core consists of a mesh of interconnected routers. The end systems exchange messages with each other by using packet-switching occurs via the routers and switches; the hosts break application-layer messages into

packets and then the network forwards packets from one router to the next, across links on path from source to destination.

Packets are transmitted over each communication link at a rate equal to the full transmission rate of the link using store-and-forward transmission. The store and forward system must receive the entire packet before it can begin to transmit the first bit of the packet to the next packet switch. The router utilizes store-and-forwarding, but first stores or buffers the packet's bits.

Each packet switch has multiple links attached to it For each link there is output buffer (output queue). There are queuing delays if the packet needs to wait before it is sent to the next link.

a: average packet arrival rate (packets/sec)
L: packet length (bits)
R: link bandwidth (transmission rate bits/sec)

Traffic Intensity: $intensity = \frac{L * a}{R} : \frac{\text{arrival rate of the bits}}{\text{service rate of the bits}}$

$intensity \sim 0$: avg. queuing delay small
$intensity <= 1$: avg. queuing delay large
$intensity > 1$: more work is arriving than can be serviced - average delay infinite!

1.3.1 Packet Loss

Since the link has finite capacity, if traffic intensity approaches 1, packet will arrive to find the queue full. If there's no place to store a packet router will drop that packet, losing it, therefore performance at a node is often measured not only in terms of delay, but also in terms of the probability of packet loss.

1.3.2 Store and Forward

Using the store and forward method, the switch must receive the entire packet before it can be transmitted on next link. While reliable, this is susceptible to packet transmission delay, as it takes $\frac{L}{R}$ seconds to transmit (push out) an L-bit packet into link at R bits/second.

1.3.3 Queuing

Loss occurs during packet queuing if the arrival rate (in bits/second) to link exceeds transmission rate (bits/second) of link for some period of time: packets will queue, waiting to be transmitted on output link, and can be dropped (lost) if memory (buffer) in router fills up.

1.3.4 Routing and Forwarding

Packets' general course are generally routed while forwarding occurs during transmission to avoid highly trafficked areas.

These general regions are the route. Each of these regions have their own network of routers in which the forwarding occurs. (Google Maps has all the cities and can route you through them but will alter your course within each city as you travel through so as to keep you on the fastest route).

Every system or host or link has a corresponding IP Address. The destination IP is recorded in the packet header, which every router reads, and forwards to an adjacent router. Each router has a forwarding table which it uses to map a destination portion to the IP header address to the router's outbound link. It's as if the packet is a letter. It has a destination address, and then also an address that it's currently on its way to next. When the packet arrives at a router based on the predefined forwarding table it finds the outbound link. Routing protocols are used to automatically set the forwarding tables.

1.4 Circuit Switching

In a circuit-switched network, the needed resources (buffers, link transmission rate) needed for communication between the end systems are reserved for the duration of the communication session. For example, phone networks are circuit switched: calls are made on demand.

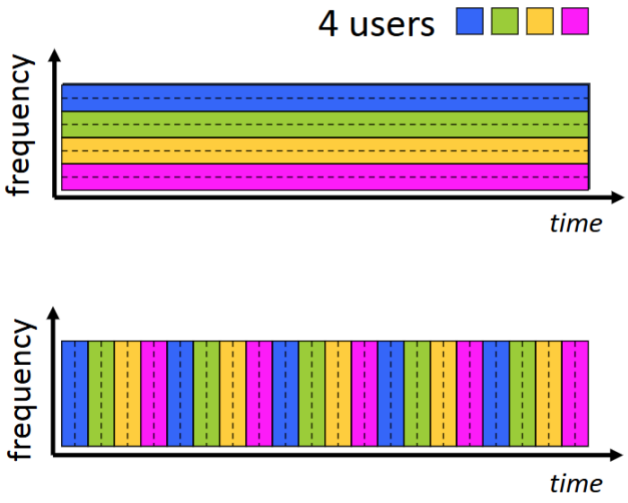
In packet switched networks, resources are not reserved, hence the delays. The sender and receiver establish a bona fide connection, switches and receiver maintain for the duration of the session.

If these were restaurants: circuit-switched networks would be restaurants that require a reservation while packet-switched networks would be ones you could walk in.

As an alternative to packet switching, circuit switching saves on end-end resources while allocating to and reserving for "calls" between the source and destination. This makes it so that the switch dedicates its resources, so it'll have circuit-like, guaranteed performance. The circuit segment stands idle if not on "call".

1.4.1 FDM and TDM

Replace with paint artwork



Frequency Division Multiplexing (FDM) is a method that uses optical, electromagnetic frequencies divided into (narrow) frequency bands (its bandwidth).

Time Division Multiplexing (TDM) is a method that divides the time into slots, allocating each call by its own band, and transmitting at that band's max rate. Each call is periodically allocated slot(s), and can only transmit at maximum rate of a wider frequency band elusively during one of its one or more allocated time slots.

FDMs transmit data all at once in parallel while TDMs transmit their data one at a time, utilizing their entire bandwidth for each distinct packet.

1.4.2 Is Packet Switching a Slam Dunk Winner?

Not necessarily. While packet switching is great for "bursty" data. If the system only needs to send data sometimes, it can allow for resource sharing and is simpler, as it has no call setup. It can, however, lead to excessive congestion, which will ultimately result in packet delay and loss due to buffer overflow.

This leads to the demand for a protocol with reliable data transfer and congestion control.

1.5 Network of Networks

Hosts connect to Internet via access Internet Service Providers (ISPs). These ISPs in turn must be interconnected so that any two hosts (anywhere!) can send packets to each other. This complex network of networks' construction and evolution is driven by economics and national policies.

Question: If given millions of access ISPs, how to connect them together? Connecting each access ISP to each other directly doesn't scale linearly, but exponentially. A network with n ISPs requires $n(n-1)$ connections. A network with 2 ISPs would require 1 connection. One with 100 would require 9900. One with 1000 would 999,000.

Why not go with the simplest option? Let's connect each access ISP to one global transit ISP (which is the current system).

A market for a global ISP to handle all interconnectedness leads to competition, which requires interconnection among all of the global ISPs, and then with private content provider networks (like Google, Microsoft, Netflix, etc...). These are all connected via internet exchange points (IXPs). When it's all said and done, there is a small number of well-connected large networks made up of Tier-1 commercial ISPs (e.g., Level 3, Sprint, AT&T, NTT) with national and international coverage. Content provider networks then connect their data centers to the Internet, often bypassing Tier-1 regional ISPs.

1.6 Bandwidth and Throughput

Bandwidth is the theoretical capacity of data transmission; the maximum amount of data that could travel from one point in network to another in a given time.

Throughput is the empirical amount of data actually transmitted and processed throughout the network.

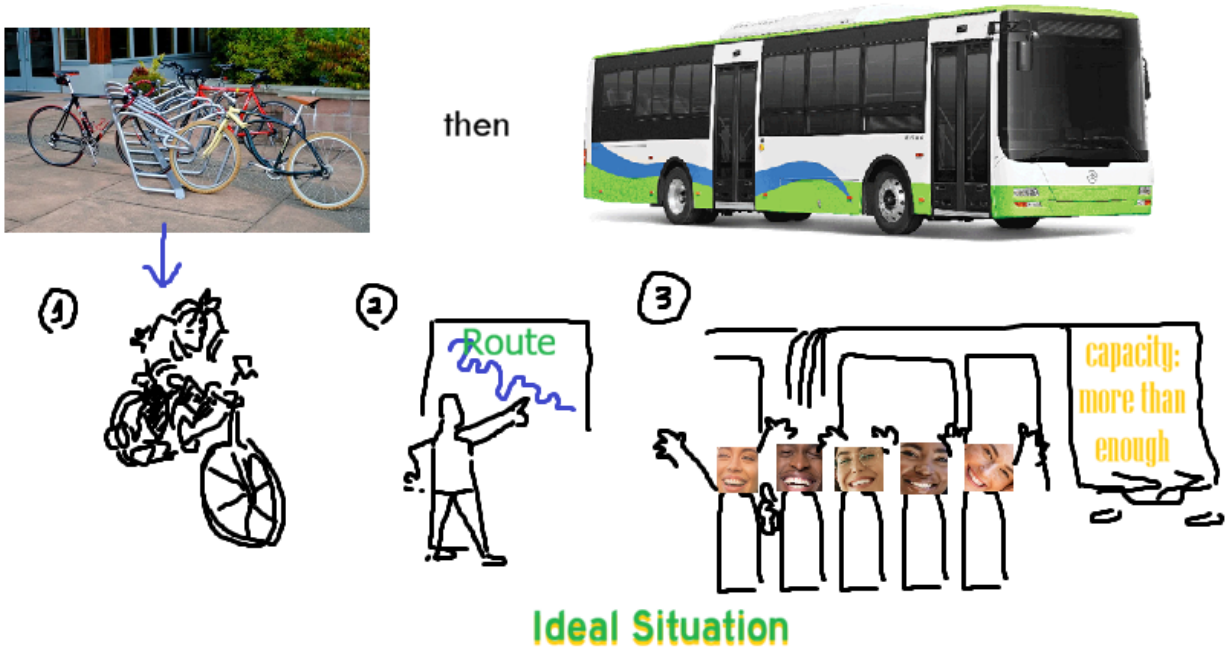
1.6.1 Throughput

Throughput: rate (bits/time unit) at which bits are being sent from sender to receiver. Instantaneous throughput is the rate at given point in time and average throughput is the rate over a longer period of time.

Imagine a pipe of fluid leading to a water routing station with rate of flow R_s and a pipe leading from the water routing station to an aquifer with rate of flow R_c.

When $R_s < R_c$, then average end-end throughput is R_c, and when $R_s > R_c$, the average end-end throughput is R_c. If there's a connection with a higher transmission rate than the cable it'll end up traveling through, it'll create a bottleneck.

Think of the same scenario, but within the context of a network instead. Here, ten connections (fairly) share the backbone bottleneck link that runs at R bits/sec. Per-connection end-end throughput $= min(R_c, R_s, \frac{R}{10})$. In practice: R_c or R_s will often be the causes of bottlenecks.



1.7 Packet Delay and Loss

Packets queue in router buffers, waiting for turn for transmission. Queue length grows when arrival rate to link (temporarily) exceeds output link capacity, and packet loss occurs when queued packets exceed memory capacity.

1.7.1 Packet delay: Four Sources

1. **Nodal Processing** is the time required to examine the packet's header, and determine the next route. Also, bit-level errors check.
2. **Queuing Delay** is the wait time before the packet is transmitted to another link
3. **Transmission Delay** is the amount of time required to transmit all of the packet's bits into the link (microseconds to milliseconds)
4. **Propagation** is the time required to propagate from the beginning of the link to router B

Transmission vs Propagation: The transmission delay is a function of the packet's length and the transmission rate of the link, but has nothing to do with the distance between the two routers. The propagation delay is a function of the distance between the two routers.

Factors impacting queuing delays:

1. Rate at which traffic arrives at the queue

2. Transmission rate of the link
3. Patterns of traffic (continuous or bursts)

1.8 Network Security

The internet was not originally designed with (much) security in mind. The original vision: "a group of mutually trusting users attached to a transparent network" :)

Today, internet protocol designers are playing "catch-up", making security considerations for all layers! Consider: how bad guys can attack computer networks, how we can defend networks against attacks, and how to design architectures that are immune to attacks.

1.8.1 Bad Guys: Packet Interception

Packet sniffing: broadcast media (shared Ethernet, wireless) exposes information. Promiscuous network interface reads/records all packets (e.g., including passwords!) passing by, which can be collected by malicious actors.

1.8.2 Bad Guys: Fake Identity

Malicious actors can spoof an IP and become its digital doppelganger. By injecting a packet with a false address, they can request data from a destination that will then send back information assuming that it's the actual IP.

1.8.3 Bad Guys: Denial of Service

Malicious actors can perform a denial of service (DoS) attack to render resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic. Guide:

1. Select target
2. Break into hosts around the network (see botnet)
3. Send packets to target from compromised hosts

Under the right conditions, DoS attacks don't require many resources from the malicious actor at all. *The call is coming from inside the house...* An attacker targeting a company (especially one that utilizes a lot of technology) will use their own resources against them.

1.8.4 Lines of Defense

1. **Authentication**: proving you are who you say you are
   Cellular networks provides hardware identity via SIM card (which is getting phased out in lieu of a digital SIM card...); no such hardware assist in traditional Internet
   Two factor authentication is one of the best security measures you can take
2. **Confidentiality**: Via encryption
3. **Integrity Checks**: Digital signatures prevent/detect tampering

4. **Access Restrictions**: Password-protected VPNs
5. **Firewalls**: Specialized "middleboxes" in access and core networks:
   Off-by-default: filter incoming packets to restrict senders, receivers, and applications
   Detecting/reacting to DOS attacks

1.9 Protocol Layers

Networks are complex, with many "pieces":

1. Hosts
2. Routers
3. Links of various media
4. Applications
5. Protocols
6. Hardware, software

1.9.1 Why layering?

Approach to designing/discussing complex systems: explicit structure allows identification, and establishes relationships of system's pieces. Modularization eases maintenance, and system updates.

1.9.2 Layered Internet protocol stack

1. Application: supporting network applications: HTTP, SMTP, DNS
2. Transport: process-process data transfer: TCP, UDP
3. Network: routing of datagrams from source to destination: IP, routing protocols
4. Link: data transfer between neighboring network elements
5. Physical: bits "on the wire"

2 Part Two: The Application Layer

2.1 Network Apps

Social networking, Web, text messaging, e-mail, multi-user network games, streaming stored video (YouTube, Hulu, Netflix), P2P file sharing, voice over IP (e.g., Skype, Whatsapp), real-time video conferencing (e.g., Zoom), Internet search, remote login

2.1.1 Creating a Network App

Communication for a network application takes place between end systems at the application layer to develop programs that run on (different) end systems, and communicates over the network. This way, the server software communicates with the browser software.

No need to write software for network-core devices (routers and switches) because network-core devices do not run user applications. Applications on end systems allow for rapid app development, propagation.

2.2 Application Architecture

From the developer's perspective, it's a five-layer static architecture designed by the application developer that dictates how the application is structured over the various end systems. These are either client-server or peer-to-peer (P2P) architectures.

Client-Server	P2P
Server: 1. Always-on host (called server) 2. Services the requests from other hosts (clients) 3. Example: Web application with always-on Web server services requests from browsers running on client hosts 4. Permanent/fixed IP address of the server 5. Often in data centers or in cloud, for scaling **Clients**: 1. Contact, communicate with server 2. May be intermittently connected 3. May have dynamic IP addresses 4. Do not communicate directly with each other 5. Think HTTP, FTP	1. Minimal or no reliance on dedicated hosts 2. No always-on server 3. No service provider, controlled by users 4. Arbitrary end systems directly communicate 5. Peers request service from other peers 6. Peers are intermittently connected and change IPs 7. Example: P2P file sharing [BitTorrent] 8. Disadvantages: security, performance, reliability

2.3 Process Communicating

Programs running in multiple end systems communicate with each other. Processes refer to the program running within the host. Within the same host, two processes communicate via inter-process communication defined by the operating system.

If two processes running on two operating systems, communication issues arise since these operating systems aren't necessarily the same (and if they are, they may be running different versions of the same operating systems). This is solved by sending messages across the computer network. The client process initiates communication while the server process waits to be contacted. For example, a web browser is a client process while a web server is a server process.

Note: Applications with P2P file systems transfer files from a process in one system to a process in another system. Both of the systems can act like clients and servers.

2.3.1 Sockets

Any message sent from one process to another must go through the underlying network. The process that sends and receives messages to and from the network interface is called a socket. If the process is a house, then the socket is its door. Sending process shoves the message out its door (socket), after which the message relies on transport infrastructure on other side of door to deliver message to the receiving process' socket. There are always two sockets involved in this sort of communication.

The socket is an interface between the application layer and the transport layer within a host. Sockets are also called Application Programming Interfaces (APIs). Developers writing APIs have control over application-layer side of the socket, but lose control once the message enters the transport layer of the socket. They do get to decide and the transport protocol and some parameters, such as maximum buffer and maximum segment sizes.

2.3.2 Addressing Processes

To receive messages, processes must have an identifier. The host device has unique 32-bit IP address.

Question: Does the host's IP address which process run on suffice for identifying the process?
No. Many processes can, and most likely be, running on the same host at any given time. To avoid this, an identifier includes both IP address and port numbers associated with the process on host. Examples of port numbers include

1. Web server: HTTPS protocol, port 80
2. Mail server: SMTP, port 25 to send HTTP message to
3. gaia.cs.umass.edu web server: IP address 128.119.245.12, port number: 80

2.3.3 Application-Layer Protocols Define

1. Types of messages exchanged, e.g., request, response
2. Message syntax: messages files and delineation (how new lines are defined)
3. Message semantics: what the message fields mean
4. Rules for when and how processes send and respond to messages
5. Open protocols: these are defined in RFCs, everyone has access to the protocol definition. This allows for interoperability e.g., HTTP, SMTP
6. Proprietary protocols: e.g., Skype, Zoom

2.4 Internet Transport Protocol Services

2.4.1 What transport service does an app need?

Data Reliability: Some apps (e.g., file transfer, web transactions, email) require 100% reliable data transfer: no packet loss while other apps (e.g., audio) can tolerate some data loss

Timing: Some apps (e.g., Internet telephony, interactive games) require low latency (packet delay) to be effective
Throughput: Some apps (e.g., multimedia) require minimum amount of throughput to be effective (bandwidth sensitive applications)
Security: Other apps ("elastic apps") make use of whatever throughput they get (file transfer)

1. Encryption (between sending host and receiving host)
2. Confidentiality
3. End point authentication
4. Data integrity

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/Video conferencing	Loss-tolerant	Audio: few kbps-1 Mbps Video: 10 kbps-5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss tolerant	Same as above	Yes: few seconds
Interactive games	Loss tolerant	Few kbps-10 kbps	Yes: 100s of msec
Smartphone messaging	No loss	Elastic	Yes and no

2.4.2 Transmission Control Protocol (TCP)

TCP is connection-oriented, requiring setup between client and server processes (handshake). Once an application finishes sending messages, the connection ends. This makes data transfer very reliable between sending and receiving processes.

TCP provides flow control, preventing the sender from overwhelming the receiver and congestion control that throttles the sender when network overloaded. It doesn't provide timing, minimum throughput guarantee, and security (it's not encrypted). Transport Layer Security (TLS) enhances TCP, providing encryption, data integrity, end-to-end authentication.

Securing TCP: Vanilla TCP and UDP sockets have no encryption. When cleartext passwords are sent into the socket, they traverse the Internet in cleartext! TLS provides data integrity and end to point authentication by encrypting messages sent through the socket.

2.4.3 Use Datagram Protocol (UDP)

UDP is a lightweight transport protocol, requiring no handshake. While lightweight, it becomes unreliable for data transfer between sending and receiving processes as it does not provide flow control, congestion control, timing, throughput guarantee, security, or connection setup.

2.5 Web and HTTP

First, a quick review of HTTP: web pages consists of objects, each of which can be stored on different Web servers. Object can be HTML files, JPEG images, Java applets, audio files, etc... Web pages consist of base HTML-files which include several referenced objects, each addressable by a URL, e.g., www.garpolytechinic.edu/CSE/staff_roster.gif where www.garpolytechnic.edu is the host name and /CSE/staff_roster.gif is the path name.

Hypertext transfer protocol (HTTP) is the web's application-layer protocol that uses a client/server model where the client is the browser that requests and receives data using HTTP protocol and displays Web objects. The server is, all the while, sending objects via HTTP in response to requests.

HTTP uses TCP: a client initiates a TCP connection by creating a socket connection to a server on port 80. The server then accepts the TCP connection from the client and HTTP messages are transferred through the sockets. The TCP connection is closed when the process is terminated. *HTTP is stateless: the server maintains no information about past client requests.*

There are two types of HTTP connections

Non-Persistent	Persistent
1. TCP connection opened 2. At most one object is sent over the TCP connection 3. TCP connection closed Downloading multiple objects requires multiple connections. The Round Trip Time (RTT) is the time for a small packet to travel from client to server and back. The HTTP response time for one object: one RTT to establish the TCP connection plus one RTT for the HTTP request and first few bytes of the response to return. Issues: 1. Requires 2 RTTs per object 2. OS overhead for each TCP connection 3. Browsers often open multiple parallel TCP connections to fetch referenced objects in parallel Response time $= 2RTT +$ file transmission time	1. TCP connection opened to a server 2. Multiple objects can be sent over a single TCP connection 3. TCP connection closed Points: 1. Server leaves connection open after sending response 2. Subsequent HTTP messages between same client/server sent over open connection 3. Client sends requests as soon as it encounters a referenced object 4. As little as one RTT for all the referenced objects (cutting response time in half)

HTTP messages are either requests or responses.

Replace with paint art

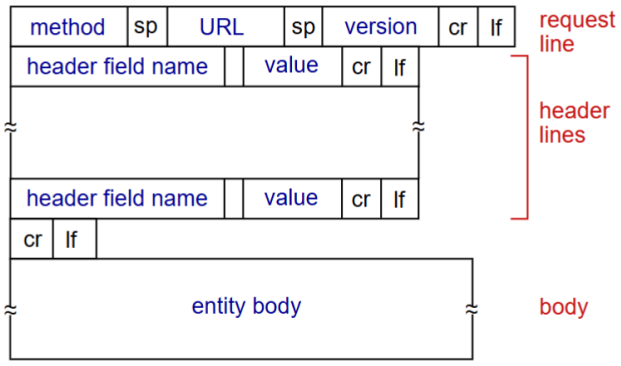


Figure 1: HTTP Request Message General Format

Other HTTP request messages include the

1. POST method, where the web page often includes form input and the user input is sent from the client to the server in the entity body of the HTTP POST request message
2. GET method (for sending data to the server), where the user data is included in the URL field of an HTTP GET request message following a '?', i.e. https://www.garpolytechnic.edu/coursesearch?Design&Patterns
3. HEAD method that only requests headers that would be returned if the specified URL were requested with an HTTP GET method
4. PUT method that uploads a new file object to the server, completely replacing the currently existing file that exists at a specified URL with the content in the entity body of the POST HTTP request message

2.5.1 HTTP Response Status Codes

200 OK	Request succeeded, requested object is found later in this message
301 Moved Permanently	Requested object moved, new location specified later in this message (in Location: field)
400 Bad Request	Request msg not understood by server
404 Not Found	Requested document not found on this server
505 HTTP Version Not Supported	HTTP Version Not Supported

2.5.2 Cookies

Since the HTTP Get/response interaction is stateless, it has no notion of what has happened in the past. To store data, websites and client browsers use cookies. Cookies are made up of four components:

1. Cookie header line of HTTP response message
2. Cookie header line in next HTTP request message
3. Cookie file kept on user's host, managed by user's browser

4. Back-end database at Web site

Let's run through an example:

1. A professor wants to apply to work at Gar Polytechnic, so they open their browser and navigate to https://www.garpolytechnic.edu/apply
2. When the initial HTTP arrives at the site, the site creates a unique ID (the cookie) and an entry in the backend database for the cookie it just created
3. Whenever that professor returns to the site, subsequent HTTP requests from that browser will contain the cookie ID, allowing the site to identify that professor

What cookies can be used for:

1. Authorization
2. Shopping carts
3. Recommendations
4. User session states (on top of the stateless HTTP)

Challenge: How do you keep the state?

The state can be kept at protocol endpoints and in messages.

2.5.3 Web Caches (Proxy Server)

The goal here is to satisfy client requests without involving an origin. The user configures their browser to point to a web cache, hosted locally on their computer. The browser then sends all HTTP requests to the cache.

1. The browser establishes a TCP connection to the Web cache and sends an HTTP request for the object to the Web cache.
2. The Web cache checks to see if it has a copy of the object stored locally. If it does, the Web cache returns the object within an HTTP response message to the client browser.
3. If the Web cache does not have the object, the Web cache opens a TCP connection to the origin server, that is, to www.garpolytechnic.edu. The Web cache then sends an HTTP request for the object into the cache-to-server TCP connection. After receiving this request, the origin server sends the object within an HTTP response to the Web cache
4. When the Web cache receives the object, it stores a copy in its local storage and sends a copy, within an HTTP response message, to the client browser (over the existing TCP connection between the client browser and the Web cache).

Web caches, otherwise known as proxy servers, act both as the client and the server. They reduce the response time for client requests as the cache is closer to the client, reduce traffic on an institution's access link, and allows more effective delivery of content.

Metric	Original Scenario	Faster Access Link	Web Cache
Scenario			
Access Link Rate	1.54 Mbps	154 Mbps	1.54 Mbps
RTT Router to Server	2 sec	2 sec	2 sec
Web object size	100K bits	100K bits	100K bits
Avg. Request Rate	15 bits/sec	15 bits/sec	15 bits/sec
Avg. Data Rate	1.5 Mbps	1.5 Mbps	1.5 Mbps
Performance			
Access Link Util	0.97	0.0097	?
LAN Utilization	0.0015	0.0015	?
End to End Delay*	2 sec + minutes + usec	2 sec + msec + usec	1.2 secs (see below)

*Internet delay + access link delay + LAN delay

Suppose the cache hit rate is 0.4: that means that 40% of requests are served by the cache with a low (msec) delay. 60% of requests are satisfied through the origin.
The rate to browser over access link would be $= 0.6 * 1.5Mbps = 0.9Mbps$
And access link utilization would be $= \frac{0.9}{1.58} = 0.58$ would be a low (msec) queuing delay at the access link
The average end to end delay would be $0.6 *$ (delay from origin servers) $+ 0.4 *$ (delay when satisfied at cache) $= 0.6(2.01) + 0.5(\sim msecs) = \sim 1.2secs$

2.5.4 Browser Caching

When caching, there's an issue. What if the cache is out of date? HTTP resolves this issue by allowing a cache to verify the date an object was last modified. If it was modified since the last time it was cached, the cache re-caches the object and responds with the new object. Otherwise, it returns the currently cached object. This is achieved by using a conditional GET. The client sends an *If-modified-since: ¡date¿* to the server and the server responds with *304 Not Modified* if it hasn't been modified since.

2.5.5 Email

Email has three major components: user agents, mail servers, and simple mail transfer protocol (SMTP). The user agent is the "mail reader" that handles the composing, editing, and reading of mail messages (Gmail, Outlook, etc...). Mail servers have a mailbox containing incoming messages for the user and a queue of outgoing messages from the user. SMTP is responsible for sending and receiving the mail.

2.5.6 SMTP RFC (5321)

Simple Mail Transfer Protocol Request For Comments (SMTP RFC) uses TCP to reliably transfer email message from a client to a server on port 25. There are three phases of the transfer: SMTP handshake (greeting), SMTP transfer of messages, and SMTP closure. Similarly to HTTP, there is a command/response interaction that takes ASCII commands and responds with a status code and phrase.

HTTP	SMTP
Client pull	Client push
ASCII command/response interaction	ASCII command/response interaction
ASCII status codes	ASCII status codes
Objects are encapsulated in response	Multiple objects are sent in multipart message
Persistent connections	Persistent connections
	Requires message (header and body) to be in 7-bit ASCII
	Uses CRLF.CRLF to determine end of message

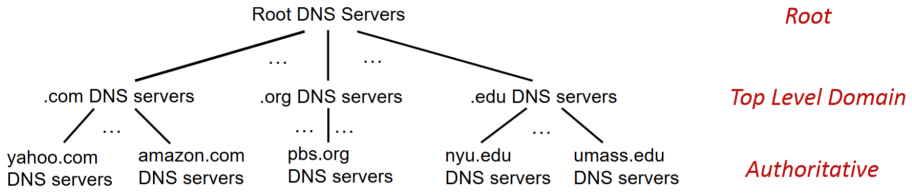
2.6 DNS: Domain Name System

People have many identifiers: their name, their SSN (Americans), their passport number. Just like people, internet hosts and routers have an identifier: the IP address. There are 32 bit values stored like 121.7.106.83. Each period separates one of the bytes expressed in decimal notation from 0 to 255. The name of the website, like garpolytechnic.edu, is for people.

The Domain Name System (DNS) is a distributed database implemented in the hierarchy of many DNS servers which use an application-layer protocol that allow hosts to query the distributed database. DNS servers are often UNIX machines running the Berkeley Internet Name Domain (BIND) software [BIND 2020]. DNS protocol runs over UDP and uses port 53.

DNS services are hostname-to-IP-address translation. It allows for host aliasing, mail server aliasing, and load distribution. DNS isn't centralized to avoid a single point of failure, high traffic volume, one point that defines all internet distances, and maintenance issues. It also doesn't scale, as Comcast DNS servers receive 600 billion DNS queries/day alone, with Akamai DNS servers receiving 2.2 trillion.

Replace with paint art



Say a client wants the IP address for www.garpolytechnic.edu.
The client queries the root server to find the .edu DNS server
The client queries the .edu DNS server to get the garpolytechnic.edu server
The client queries garpolytechnic.edu to get the IP address for www.garpolytechnic.edu

There are more than 1000 root server instances across the globe, representing 13 different root servers managed by 12 different organizations coordinated through the Internet Assigned Numbers Authority (IANA 2020).

Top level domain (TLD) servers are responsible for .com, .org, .net, .edu, .aero, .jobs, .museums, and all top-level country domains like .cn, .uk, .fr, .ca, and .jp.

Authoritative DNS servers are the organization's own DNS server(s), providing the authoritative hostname to IP mappings for the organization's named hosts. It These can be maintained by an organization or service provider.

2.6.1 Local DNS Name Servers

When a host makes a DNS query, it is sent to its local DNS server. When the local DNS server returns rely, they answer forwarding the request into the DNS hierarchy for resolution. Each ISP has a local DNS name server. (Find yours with MacOS: `% scutil --dns` and Windows: `ipconfig /all`).

DNS name resolution can occur in multiple ways.
Via an iterated query: connected servers reply with the name of a server to contact if it isn't the destination itself.
Via a recursive query: putting the burden of the name resolution on the contacted name server. This leads to heavy loads at upper levels of hierarchy.

2.6.2 Caching DNS Information

Once any name server learns a mapping, it caches that mapping and immediately returns a cached mapping in response to a query. Caching improves response time, and caches disappear after a timeout called time to live (TTL).

2.6.3 DNS Records

DNS databases store resource records (RRs), formatted as follows: (name, value, type, ttl).

Type	Name	Value
A	Hostname	IP Address
NS	Domain (e.g. garpolytechnic.edu)	Hostname of authoritative server for this domain
CNAME	Alias for some canonical name, i.e. www.ibm.com is really servereast.backup2.ibm.com	Canonical Name
MX	Domain	Name of SMTP mail server associated with its name

2.6.4 DNS Protocol Messages

DNS query and replay messages have the same format:

2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable #)	
answers (variable # RRs)	
authority (variable # RRs)	
additional info (variable # RRs)	

2.6.5 DNS Security

DDoS attacks bombard root servers with traffic. They haven't been successful to date due to traffic filtering and local DNS servers caching IPs of top level domain (TLD) servers, allowing for a root server bypass. TLD servers are more at risk.

Spoofing attacks intercept DNS queries, returning bogus replies. See DNS cache poisoning.

2.7 Peer-to-peer (P2P) Architecture

P2P architectures have *no* always on server. In a P2P system, arbitrary end systems directly communicate with each other, with peers requesting service from other peers, and providing service in return to other peers. This offers self-scalability, where new peers bring new service capability and new service demands. Peers are intermittently connected and change IP addresses. This is a lot to manage. Examples of P2P usage in the wild are seen in file sharing with BitTorrent, streaming with KanKan, and VoIP with Skype.

2.8 File Distribution

It's time to clock the distribution of a file of size F to N peers in a client-server network vs a P2P network.

2.8.1 Client-Server

A client-server network must sequentially upload and send N file copies, with the time to send one copy being $\frac{F}{u_s}$. Time to send N copies $= N\frac{F}{u_s}$. While this is happening, clients are downloading the file copies with d_{min} being the min client download rate and the min client download time being $\frac{F}{d_{min}}$. The time to distribute a file of size F to N clients is $D_{client-server} \geq max\left\{N\frac{F}{u_s}, \frac{F}{d_{min}}\right\}$. Note how the time increases linearly thanks to N.

2.8.2 P2P

A P2P network must upload at least one copy with time to send one copy being $\frac{F}{u_s}$. After the initial upload, clients must download the file copy with the min client download time being $\frac{F}{d_{min}}$. Clients as an aggregate must download

NF bits where the max upload rate (limiting max download rate) is $u_s + \Sigma u_i$. The time to distribute a file of size F to N clients is $D_{P2P} \geq max \left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, N\frac{F}{(u_s + \Sigma u_i)} \right\}$

2.8.3 P2P File Distribution: BitTorrent

When using the BitTorrent protocol, the distributed file's is divided into 256Kb chunks. Peers in the torrent send/receive file chunks while a tracker tracks peers in the torrent. If a peer joins the torrent while it's in progress distributing a file, it will accumulate them over time. While downloading the file, peers upload chunks to their peers. Peers may come and go as they please in a process called churning. Once a peer has the entire file, it may selfishly leave or altruistically remain in the torrent to help with distribution.

At any given time, different peers have a different set of time chunks. Periodically, peers ask each other what chunks they have, requesting the rarest chunks that they don't have first. A peer will send chunks to the four peers sending them chunks at the highest rate. Peers other than those four are choked by the peer sending the file chunks. The top four are re-evaluated every ten sections, and every thirty seconds the sender will randomly select another peer to start sending chunks to optimistically unchoke that peer.

2.9 Video Streaming and Content Distribution Networks (CDNs)

Streaming video traffic is a major consumer of internet bandwidth, trafficking Netflix, YouTube, Amazon Prime, which have 80% of residential ISP traffic (2020). The challenge is reaching circa one billion users. Users request to stream on demand and are not heterogeneous, as they have different devices with different capabilities. The solution is a distributed, application level infrastructure.

2.9.1 Video

Constant bit rate (CBR) encoded video has fixed rate video encoding while variable bit rate (VBR) encoded video changes its encoding rate as the spatial and temporal coding change.

When streaming stored video, a video server will send packets of encoded video data through the internet to the client. Server to client bandwidth varies over time due to congestion levels and lead to packet loss due to congestion. This either delays playback or results in poor video quality. The client also can interact with the video, pausing, skipping forward and back, and jumping through the video. To compensate, the client buffers the video stream.

2.9.2 Streaming Multimedia: DASH

Dynamic Adaptive Streaming over HTTP is a multimedia streaming method where the server will divide the video file into multiple chinks, encode each chunk at a different rate, store the different encoding rates in different files, and then replicate the files at various CDN (content distribution network) nodes. A manifest then provides URLs for different chunks. The client then periodically estimates server-to-client bandwidth and requests one chunk at a time while consulting the manifest, choosing the maximum encoding rate sustainable given the current bandwidth. Over time, it will pull differently encoded video files from potentially different servers for the smoothest playback experience.

2.9.3 Content Distribution Networks

The first option for a CDN to reach millions of simultaneous users is for them to have a single, large mega-server, but this is a single point of failure and will lead to congestion when users attempt to stream popular content. Their second option is store and serve multiple copies of videos at multiple geographically distributed sites.