HTTP vs SMTP

HTTP	SMTP
Client pull	Client push
ASCII interaction	ASCII interaction
ASCII status codes	ASCII status codes
Objects encapsulated	Multipart messages
Persistent connections	Persistent connections

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

Type	Name	Value
A	Hostname	IP Address
NS	Domain	Host server
CNAME	Alias	Canonical Name
MX	Domain	Mail Server

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)	

Datagram

source port #	dest port #
sequence number	
acknowledgement number	
head len / unused / C / E / U / A / P / R / S / F	Receive Window
checksum	urg data pointer
options (var len) & max MSS	
application data (variable length)	

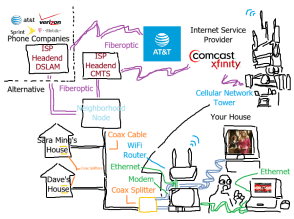
$$D_{client-server} \geq max\left\{N\frac{F}{u_s}, \frac{F}{d_{min}}\right\}, \quad D_{P2P} \geq max\left\{\frac{F}{u_s}, \frac{F}{d_{min}}, N\frac{F}{(u_s+\Sigma u_i)}\right\}, \quad packet\ \ delay = \frac{L \text{ bits}}{R \text{ bits/second}}$$

1.1. **Internet Layers**: Application (messages) → Transport (segments) → Network (datagram) → Link → Physical. Network layers provide separation of responsibility and modularity.

1.2. **The Internet** and **Internet Infrastructure** consist of computing devices → packet switches → communication links → networks. Internet **Protocols** are rules on how data packets are sent through different parts of the internet (Network Edge, Physical Access Networks, Network Core). **Internet Standards** include *Request for Comments* and the *Internet Engineering Task Force* (IETF)

1.3. **Digital Subscriber Lines (DSLs)** route data through a **DSLAM** (Access Multiplexer) and allow for internet and phone data to travel through the same coax cable. **Wi-Fi** networks provide wireless packet transfer via radio waves. **Enterprise Networks** connect devices in an institution over a localized network.

1.4. **Physical Media**: **bits** that propagate between transmitter and receiver pairs. **Physical links**: **Fiberoptic cables**: insulated glass tubes that propagate information via light (fastest). **Coaxial cables**: 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. **Twisted Pair**: 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.
Connection Types include **simplex**, which is a one way connection (think public radio), **half duplex**, which is a two way connection where only one signal is sent at a time (think HAM radio), and **full duplex**, which is a two way connection where signal travels both ways simultaneously (think phones).



1.5. The **Network Core** are the devices at the heart of the network and **Packet Switching** is what occurs to propagate the data found in the core to the **Network Edge**. **Circuit Switched** networks are reserved lines (unused today, but think circuit-switched phones), while **Packet Switched** networks are on demand.

Routers contain **buffers** that stored queued packets inside a buffer and forward them between networks. **Queuing Delays** occur due to network congestion. **Packet Loss** occurs when packets are **queued** into switches faster than they can be forwarded. **Routing** is akin to a city-per-city route while **forwarding** is the act of sending packets on the fastest local route. A packet's **Traceroute** consists of the hops it makes through different routers from its source to its destination.

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. Basic security: **Packet Interception** involves packet sniffing of exposed information in promiscuous networks when broadcast media (shared Ethernet, wireless) transmits all reads/records (e.g., including passwords!), in a format that can be collected by malicious actors. **Fake Identity** allows malicious actors to **spoof** an IP and become its digital doppelganger via packet injection, allowing them to request data from a destination that will then send back information assuming that it's the actual IP. **Denial of Service** renders resources (server, bandwidth) unavailable to legitimate traffic by overwhelming resource with bogus traffic. **Lines of Defense**: authentication (hardware assist), confidentiality (encryption), integrity checks (digital signatures), access restrictions (VPNs), and firewalls (middleboxes in access and core networks).

2.1. The **Client Server architecture**: *Servers* are an always-on host (called server) that services the requests from other hosts (clients), i.e. webapp. Servers have permanent/fixed IP addresses, and are often held in data centers or in cloud, for scaling. *Clients* contact and communicate with the server, may be intermittently connected, may have dynamic IP addresses, and don't communicate directly with each other.

P2P has minimal or no reliance on dedicated hosts. It has no always-on server or service provider, and is controlled by users. Arbitrary end systems directly communicate, made up of intermittently connecting peers that request service from each other and change IP. (P2P file sharing: BitTorrent). Disadvantages: security, performance, reliability.

2.2. **Sockets** are the means which a server provides access to its **Application Programming Interface** (API), which allows clients to make requests for certain information.

IP Addresses are unique, 32-bit values (IPv4) or 128-bit values (IPv6) that act as identifiers for clients, which have ports identified by **Port Numbers** that act as where the data flows through.

Applications with **no loss tolerance**: File transfer/download, email, web docs, smartphone messaging. Applications with **loss tolerance**: internet telephony/video conferencing, streaming stored video/audio, interactive games.

2.3. The **Transmission Control Protocol** (TCP) service is a connection-oriented, handshake-using protocol. Connection starts and ends with communication, making data transfer via TCP very reliable. TCP has flow control (preventing senders from overwhelming receivers) and congestion control (that throttles the sender when network when overloaded). It doesn't provide timing, minimum throughput guarantee, and security (it's not encrypted).

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

User Datagram (UDP) is a lightweight transport protocol, requiring no handshake. It's 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.

2.4. **Hyper Text Transfer Protocol** (HTTP) is the web's application-layer protocol that uses a client/server model and TCP on port 80. HTTP is stateless. **Status Codes**: 200 (OK), 301 (Moved Permanently), 400 (Bad Request), 404 (Not Found), 505 (HTTP Version Not Supported).

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: the cookie header line of HTTP response message, the cookie header line in next HTTP request message, the cookie file kept on user's host device (managed by the user's browser), and an ID in a back-end database at the website. When an 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.

Non-Persistent: TCP connection opened, at most one object is sent, 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 is one RTT to establish the TCP connection plus one RTT for the HTTP request and first few bytes of the response to return. Issues involve the fact that it requires 2 RTTs per object, has an OS overhead for each TCP connection, and browsers often open multiple parallel TCP connections to fetch referenced objects in parallel. Response time $= 2RTT +$ file transmission time

Persistent: TCP connection opened to a server, multiple objects can be sent, connection closed. Points: Server leaves connection open after sending response, subsequent HTTP messages between same client/server are sent over the open connection, client sends requests as soon as it encounters a referenced object. Response time: As little as one RTT for all the referenced objects (cutting response time in half).

Web caches (proxy servers) satisfy client requests without involving the origin and instead replying with cached data. This reduces client request response time and traffic at the origin.

Conditional Gets allow HTTP requests to save bandwidth by specifying whether a server should send new data or whether the local cache should by used in a field called **If-Modified-Since**.

HTTP 1: client requests 1 large object (e.g., video file) and 3 smaller objects, **HTTP 2**: decreased delay in multi-object HTTP requests. **HTTP 3**: adds security, per object error and congestion control (more pipelining) over UDP.

2.5. **Email and Mail servers** route mail traffic via Simple Main Transfer Protocol (SMTP) using ASCII on port 25. Transfer: SMTP handshake, transfer of messages, and closure. They contain a mailbox with the incoming user messages and a message queue of outgoing messages. The mail reader is called a user agent. **Pull** (HTTP) and **Push** (SMTP).

2.6. The **Domain Name System** cascades from Root servers, to Top Level Domain servers (.com, .org, .gov, .br), to Authoritative servers (pbs.org). DNS services provide hostname-to-ip address translation, host translation, mail server aliasing, and load distribution by replicating web server (many IPs correspond to one name). For efficiency, DNS servers employ **DNS Caching**, which involve local DNS servers cache data in maps from broader to more specific addresses. **DNS security** addresses DDoSing (Distributed Denial of Service) attacks. These are mostly unsuccessful, but if TLD servers are more at risk and **IP Spoofing** where malicious actors send bogus replies. DNS Cache poisoning caches bogus IP mappings in DNS servers, redirecting queries to a different IP.

2.7. **Content Distribution Networks** (CDNs) store and serve music, video, and movies from a distributed network of servers.

3.1. **Transport services** provide logical communication between application processes running on different hosts. Senders break application messages into segments to pass to the network layer while receivers reassemble segments into messages to pass to the application layer.

The act of **Multiplexing** involves encapsulating data into chunks with header information to be used during **Demultiplexing**, the act of delivering data from the transport-layer segment to the correct socket.

TCP uses connection setup, is reliable and provides in-order delivery, congestion control, and flow control. **UDP** is unreliable and uses unordered delivery, a no-frills extension of "best-effort" IP.

3.2. The goal of the **Internet Checksum** is to detect errors caused by flipped bits (copper wire creates a magnetic field that puts data transferred through at risk).

3.3. The **Principles of Reliable Data Transfer** (RDT): The complexity of the RDT strongly depends on the characterstics of the unreliable channel (does data get lost, corrupted, or reordered?), senders/receivers don't know each others' "state". RDT **1.0** assumes reliable transfer over a reliable channel. **2.0** (channel with bit errors: more realistic) uses ACKs and NAKs with checksum bits (in case of corruption), error detection, receiver feedback, retransmission, uses stop and wait (await response). **2.1** (sender/receiver handle garbled ACK/NAKs). **2.2** is NAK-free by including the sequence number of acknowledged packets in the ACK. **3** (channels with errors and loss) use a countdown timer to wait until it assumes a packet has been lost.

3.4. **Acknowledgments** (Acks) are packets that affirm that data has been received. **Negative Acknowledgments** (Naks) are packets that inform that the receiver is missing data because a packet didn't arrive or was mangled.

NAK free systems do not use negative acknowledgments while transferring data.

3.5. **Pipelined protocols** increase utilization by sending multiple, yet-to-be-acknowledged packets with unique sequence numbers:

Go-Back-N (GBN) corrects for errors by ensuring that packets within a certain window have been received. If one is missed, the window shifts back.

Selective repeat corrects for errors per-packet by waiting for individual acknowledgments from packets.

3.6. The TCP **3-way handshake**: "On belay?", "Belay on", "Climbing". Before exchanging data, the sender and receiver must agree on establishing a connection and on the connection parameters that they'll use.

TCP flags: Congestion Window Reduced, Echo (Explicit Congestion Notification) Urgent, Acknowledgment, Push, Reset, Synchronization, and Finish bits. **Flow control** involves the receiver throttling the sender so that the sender won't overflow the receiver's buffer too much by transmitting too fast. TCP connections are **closed** when the FIN bit is set to one.

3.7. **Congestion control**: Scenario one: one router, infinite buffers (no retransmission needed). Scenario two: one router, finite buffers (sender retransmits lost, timed-out packets). Scenario three: packets can be lost when dropped at the route due to full buffers. The sender knows when the packet has been dropped and only resends it if it's known to be lost.

Congestion control strategies: **End To End** provides no explicit support from the network layer. **Network-assisted** provides direct feedback to sending/receiving hosts and may indicate congsetion level or explicitly set sending rate. **Additive Increase, Multiplicative Decrease** (AIMD) increases the sending rate by one every RTT until loss is detected, in response to which it cuts the sending rate in half.