1 Communication Basics & Equations

The internet is a network of networks. It is **packet switched** - info transmitted in *packets* that *switches* / *routers* forward based on the info it contains. A **circuit switched** network would establish dedicated conns in the setup phase (expensive), but removes processing cost & space overhead.

Protocols are agreements between communicating perties on how comm will proceed. It consists of a **handshake**, the **conversation** & **closing**.

Bandwidth is the amount of info that can get into or out of the conn per time unit.

Throughput is the amount of info that does get into or out of the conn per time unit. At *steady state*, input throughput = output throughput. **Goodput** is useful throughput = throughput − overhead.

Latency or **propagation delay** is time for 1 bit to go through the conn.

- **Latency** $d = t_1 - t_0$ or $\frac{\text{dist}}{\text{wave prop. speed}}$
- **Throughput** $R = \frac{L}{t_2 - t_1} = \frac{\text{transferred bits}}{\text{duration}}$
- **Packetization** (*trans delay*) $= \frac{L}{R}$
- **Transfer Time** $\Delta = d + \frac{L}{R}$.
- **Utilisation** $U = \frac{L/R}{RTT + L/R}$.
- **Round Trip Time** $RTT = 2 \times d$.
- **Traffic Intensity** $= \frac{\text{Arrival Rate} \times L}{\text{Trans Rate}}$.

Every hop, there is **processing delay** (typically negligible), & **queueing delay** (time waiting for output link trans), which depends on router congestion.

1.1 Networks

WAN Wide Area Network (e.g., Internet). ***LAN*** Local Area Network (e.g., home Wi-Fi, office network). ***PAN*** Personal Area Network (e.g., Bluetooth devices)

2 Application Layer (L5)

A **client** initiates comm, a **server** waits to be contacted. In **Peer to Peer (P2P)** arch, an app is both. Each proc within a system is addred with a **port number**.

OS manages network interfaces with **sockets**. Client creates socket by connecting, which it uses to send & receive data, then disconnects. The server creates a socket by accepting a connection.

The **World Wide Web** is based on **hypertext** documents (may contain *objects*) & **hyperlinks**. **HTTP** is a (usually) *TCP* protocol. It is stateless, with **persistent** connections:

Line	Request Example
Req.	`GET /index.html HTTP/1.1`
Head	`Host: www.example.com`
Empty	
Body	`<optional body>`
Line	**Response Example**
Status	`HTTP/1.1 404 Not Found`
Head	`Content-Length: 1234`
⁞	`Content-Type: text/html`
Empty	
Body	`<html>...</html>`

Status code can be **1xx** (info), **2xx** (success), **3xx** redirect, **4xx** (client err), **5xx** (server err).

A **proxy** server may cache data sent to client - reduing as a server. Possibly store (cache) the object for some time, in case ces latency & traffic, enchances security. May implement a *firewall*. May also increase latency on cache miss.

HTTP is stateless, but **cookies** use extra headers to identify user sessions. Also, **common gateway interface (CGI)** allows identifying a program & parameters in the url, which a server generates a page for on-the-fly.

2.1 DNS

The **domain name system (DNS)** maps names to IPs - stored on 13 root-servers. Each top level domain has its own root server, which then has authoritative servers. **Caching** dramatically improves perforamnce, but introduces risk of **DNS cache poisoning**. Each record contains a `name`, `value`, `type` & `time to live` (*TTL*). It is connectionless, on UDP 53.

2.2 CDN

A **content distribution network** stores copies of content at geographically distrbuted servers. Either **enter deep** (store content on network edge) or **bring home** (store content at network core). Common CDN DNS are `8.8.8.8` & `1.1.1.1`.

2.3 Email

Async, one-way comm with multimedia content. No auth, confidentiality or delivery guarantee. **Simple Mail Transfer Protocol (SMTP)** sends emails. A msg has *headers (start with HELO)*, *body*, *. to end the msg*, *QUIT to end the session*. ESTMP also encrypts. **Multipurpose Internal Mail Extensions (MIME)** define extensions for encoding outside of 7-bit ASCII (html, images, etc).

Post Office Protocol (POP3) is a protocol to access incoming & outgoing msgs. However, it assumes retrieved mail is deleted at the server. Instead **Internet Message Access Protocol (IMAP)**, or secure **IMAPS**.

2.4 Common Protocols

Name	Port	Transport
FTP	20,21	TCP
SSH	22	TCP
TELNET	23	TCP
SMTP	25	TCP
DNS	53	TCP, UDP
DHCP	67, 68	UDP
HTTP(S)	80 (443)	TCP †
POP3(S)	110 (995)	TCP
IMAP(S)	143 (992)	TCP
BGP	179	TCP

† Can be UDP on QUIC.

3 Transport Layer (L4)

Now only two protocols, **trans Control (TCP)** & **User Datagram (UDP)**. Every host has unique `IP:port`, & no guarantee for *data integrity* or *order of delivery*. Allows (de)multiplexing.

3.1 TCP

- **CLOSED** No conn exists.
- **LISTEN** Waiting for incoming conn (server side).
- **SYN_SENT** SYN sent, waiting for SYN-ACK (client initiates).
- **SYN_RECEIVED** SYN received, SYN-ACK sent (server response).
- **ESTABLISHED** conn open, data can be exchanged.
- **FIN_WAIT_1** FIN sent, waiting for ACK or FIN.
- **FIN_WAIT_2** ACK received for FIN, waiting for peers FIN.
- **CLOSE_WAIT** FIN received, waiting to send own FIN.
- **CLOSING** Both sides sent FIN, waiting for ACK.
- **LAST_ACK** Sent FIN, waiting for final ACK.
- **TIME_WAIT** You closed conn, waiting (2ŒMSL) before cleanup.

Connection oriented service, **full-duplex**:

- **SOCKET** - create connection endpoint.
- **BIND** - attach local addr to socket.
- **LISTEN** - say "im accepting N conns".
- **ACCEPT** - wait until client wants conn.
- **CONNECT** - attempt to establish conn.
- **SEND** - send data over a conn.
- **RECEIVE** - receive data over a conn.
- **CLOSE** - release the connection.

Data is transmitted in **TCP segments**, **max segment size (MMS)** depends on *max trans unit (MTU)* of the conn: largest link-layer frame available to sender. Header is min 20B:

- **Source & Destination Ports**.
- **Sequence Number** for ordering.
- **Ack number** for reliable transfer.
- **ACK Flag** - valid ack number.
- **SYN Flag** - establish connection!
- **FIN Flag** - terminate connection!
- **Checksum** for validating integrity.
- *Other optional headers & flags.*

Header: *Seq num* is total bytes sent so far, *ack num* is first seq num not yet seen by receiver. Typically, ack every other packet. We make sure all data is received, in the correct order, & only once. To connect, we use a **3 way handshake**:

1. Client sends segment with `SYN` & initial sequence number.
2. Server responds segment with `SYN`, `ACK`, & its own initial seq num.
3. Client responds segment with `ACK`, & new sequence number.
4. *To disconnect, same thing with FIN.*

Max Segment Size (MSS) does *not include header*. Don't subtract it!

We can **multithread** TCP to improve performance - conn code is slow. Allow simul conns, as *ACCEPT is blocking*.

3.2 UDP

UDP provides only app identification & a CRC-checksum. *No flow control, error control or retransmission.* Max datagram is 64KiB (8B header), connectionless. Headers have src, dst, len & checksum. Provides *finer app level control* over what is sent & when, *no conn establishment* (faster), *no conn state to maintain* (simpler) & *small packet header overhead*. Besides real-time apps, UDP is also very useful for short client-server interactions.

3.3 Reliable Data Transfer

- **Error Detection** - use parity bit (XOR), checksum or Hamming code.
- **Receiver Feedback** - reciever notifies the sender on bit error.
- **Retransmission** - receiver retransmits.

We can try *stop-and-wait*, only continue transmission on ACK, but this doesn't deal with *invalid ACK/NACK*s. We can assume sender will always detect corruption, making ACKs redundant; or always assume NACK & retransmit. Instead of using an ACK & NACK, we keep sending ACKs for the seq num of the last good packet received.

On the receiver:

- If an *in-order* seg arrives with expected seq num, & all data so far has arrived, send a **delayed ACK**. Wait Xms for another in-order seg, & if that does not arrive send ACK.
- If an *in-order* seg arrives with expected seq num, & another in-order seg is waiting for ACK, send a **cumulative ACK** immediately for both segs.
- If an *out-of-order* seg arrives with a higher than expected seq num, immediately send a **duplicate ACK**.
- If a seg arrives that fills a gap, send an **immediate ACK** if the seg fills the lower end of the gap.

To account for unreliable channels, we also introduce **timeouts**.

3.4 Congestion

If all traffic is ACKed, there is no **congestion** - a full queue in a router in a packet's route. Segment loss (timeout or duplicate ack) mean congestion. The sender maintains **congestion window** W - the max bytes to send without receiving ACK, W = min(CongestionWindow, ReceiverWindow), & max throughput $\lambda = \frac{W}{RoundTripTime}$.

1. Initially W =. In **slow staft phase**, double W for every good ACK until $W > W_{\text{thresh}}$, or congestion event occurs.
2. In **congestion avoidance phase** $W' = W + \frac{MSS^2}{W}$ every round trip, until congestion event occurs. At every packet loss, half W. (*additive increase / multiplicative decrease*)
3. If a **timeout** $T = SmoothRT + 4 \times \sigma^2_{RTT}$, reset $W' = MSS$ & do slow start.
4. If a **NACK** (3 dup ACKs), halve $W' = \frac{W}{2}$ & do congestion avoidance. This is called **fast recovery**.

Sender transmits many segs without waiting for ACKs using a **sliding window**. **Flow control** allows the receiver to control the sender with a **receiver window**.

4 Network Security

Rootkits *secretly* control a computer. **Keyloggers** record all keypresses. **Trojans** control a computer remotely. **Evil Twins** lure their victims into networks attackers control / own.

4.1 Firewalls

A **guard** controls which *principals* can access a *resource*. A **firewall** controls access to a network - a security gateway between the *internal* & *external* networks. It can be a **stateless packet filter** (check IP & ports), **stateful packet filter** (remembers conns & checks curr & prev packets), **circuit level gateway** (fully takes over connection), **proxy server** (runs on network, protects entire LAN, does caching), **application level gateway** (only protects host), or **hybrid**. A *proxy* has 3 modes:

1. **Normal** - client aware of proxy & needs to be set up for it.
2. **Transparent** - client unaware of proxy, router takes care of everything.
3. **Reverse** - runs on receiving side, impersonating senders.

A **bastion host** expects to be attacked: auditing, logging, monitored & isolated. It may act as a *proxy firewall*.

Stateful inspection knows active conns:

- Relays conns & maintains conn state.
- Auths users.
- Drops conns based on dest, time, etc.
- Useful for logs / audits / monitoring.

Intrusion Detection System (detects); **Intrusion Prevention System** (detects & stops); **Next Gen Firewall** (IDS, IFS & ACL); **Unified Thread Man** (NGFW, antivirus, antispam, content filter, etc). **Access Ctrl List** (ACL) always picks the best match.

A **demilitarized zone (DMZ)** is a neutral zone between ext & int networks. **Port forwarding** tells router packets for certain ports should be forwarded directly to an internal host / port.

4.2 Securing a Network

- Create **backup**s.
- Keep OS & critical software **updated**.
- Apply **principle of least privelage**.
- Use a **proxy** server or **VPN**.
- **Application level gateway** on all hosts.

4.3 Encryption

Ciphertext $M_C = E(K,M)$ & **plaintext** $M = D(K^{-1},M_C)$. In **symmetric encryption** $K = K^{-1}$, & must be distributed carefully. When $K \neq K^{-1}$ **assymmetric encryption**. Encrupt M with pubkey of dst, only dst can decrypt. Encrypt M with src prikey, so that dst can verify that src sent it. **Diffie-Hellman** key exchange:

1. Agree on public p and g.
2. Generate private keys a and b.
3. Calc pubkeys $A = g^a \mod p$, $B = g^b \mod p$, and exchange them.
4. Secret $S = B^a \mod p = A^b \mod p$.

Alternatively, use *trusted third party* for key exchange. Potentially insecure. A **hash function** one way func to map data of any size to data of fixed size. Used to store passwords: `hash(pass + salt)`.

5 Network Layer (L3)

The **Internet Protocol (IP)** header (20*B*) contains *src & dst IPs*, *protocol*, *TTL*,*checksum*, *options*, *frag info*, etc. Datagrams are **fragmented** when they exceed the maximum trans unit (MTU). The header contains a frag flag, & frag offset (8 ***bytes*** unit). IP addr (*32B*) identify **interfaces**, managed by ICANN. 3 level heirarchy:

1. **External Routers** only addres & forward to correct network.
2. **Subnet Routers** apply mask & look up dst in their subnet. If found, router knows interface to forward packet to.

Network masks use either **prefix** `128.138.207.160/27` or **subnet mask** `128.138.207.160/255.255.255.224`. This is called **classless interdomain routing (CIDR)**. Routers try to match longest possible prefix by *&* IP & SM.

Dynamic Host Config Protocol (DHCP) assigns IPs in a subnets. Each machine broadcasts **DHCP Discover**, server replies with IP - can maintain static mapping. We require periodic refresh (*leasing*) to prevent hosts from keeping IPs forever. **Network Addr Translation (NAT)** partially solves IP shortage by translating a public IP into private ones. A *NAT router* does this automatically, but it breaks the end-to-end principle. Private addrs:

- `10.0.0.0`-`10.255.255.255`/*8*.
- `172.16.0.0`-`172.31.255.255`/*12*.
- `192.168.0.0`-`192.168.255.255`/*16*.

Special IPs include:

- `0.0.0.0`/*0*: **default** (*unknown*) route.
- `0.0.0.0`/*32 (or 8)*: **localhost**. Must *not* be sent.
- `127.0.0.0`/*8*: **loopback** to send a localhost addr.
- `169.254.0.0`/*24*: **Link Local** - something went wrong when acquiring IP.

Internet Control Message Protocol (ICMP) sends errors & operational info - used by router comm by *encapsulating control msgs* in IP packets.

IPv6 has **expanded addressing** (128 bits), supports reserving bandwidth for packet **flow**, supports **anycast**, no fragmentation (done by sender), no checksum, fixed length options.

5.1 Routers

Routers have IO **interfaces** / **physical ports**. Moving data requires **hops** at intermediate routers, for which network **topology** must be known. **Load balancing** required, **network heterogeneity** accounted for. **Datagram networks** are:

- **Packet Switched** - information transmitted in discrete units - datagrams.
- **Connectionless** - self contained msgs.
- **Best-Effort** - no delivery guarantees.

Routers cooperate to find best routes between nodes, together building a **sink tree** that maps hosts to *optimal routes*.

1. **Flood Routing** - forward everywhere but source. To prevent *drowning*, use a **hop counter**, avoid directed cycles, or **flood selectively**. Always produces shortest path, but high overhead.
2. **Distance Vector Routing** - select neighbour who's advertised cost, added with its own cost to get to the neighbour is lowest. Then, advertise new cost to other neighbours. However, we can *count to infinity*, if a link goes down & cost to dst increases by 1 each hop. (If you don't know, ask neighbour, but they don't know, etc).
3. **Link State Routing** - calculates a **sink tree**. Each router: discovers addrs of direct neighbours; calculates cost for each; constructs **Link State Advert (LSA)** packet; sends / collects LSA from *ALL* other routers; runs Dijkstras. Identification with `HELLO` packet & cost measurement with `ECHO` packet. LSA is flooded.
4. **Heirarchical Routing** - LSR cannot scale, so we have **regions**.
5. **Broadcast Routing** - in **reverse path forwarding (RPF)** every router forwards to adjacent routers, but accepted iff it came from a *direct* (unicast) path between them & the src.
6. **Multicast Routing** - **Core Based Tree**: single spanning tree per node group with core near middle. To send a multicast msg, send to core.

The **broadcast domain** ends at the router.

5.2 The Network Model

Internet is organised into **autonomous systems** & **gateway routers**. Within an AS, we use **interior gateway protocols** e.g. **Open Shortest Path First (OSPF)**: an *open* link state routing protocol that supports different *distance metrics*, *adaptability*, *load balancing* & *heirarchical routing*. It abstracts networks into a dir graph, each edge has cost. It computes shortest paths, allowing dividing AS into areas with **area border routers** & a **backbone area**.

Border Gateway Protocol (BGP): an inter-AS routing protocol used by all AS on the internet. Neighbouring routers maintain a conn to transmit reachability information within its AS & provide it to neighbour AS. It is **path-vector** - router decides entire path. A **BGP Ad** denotes dsts with addr prefixes. It contains **AS Number** (unique ID), **BGP Attrs**, **BGP Import Policy** (decides whether to accept/reject route ad).

6 Data Link Layer (L2)

Ethernet transmits data in cables of three forms: **Straight Through (MDI)** (between different OSI layers, e.g. switch-router); **Crossover (MDIX)** (between same OSI layer, e.g. switch-switch); **Rollover** (tap *directly* into dev to debug). Connecting router to PC is a **straight through** cable.

Ethernet Frames header contains *src/dst addr* & *ether type (IPv4, ARP, etc)*. Footer contains **Cyclic Redundancy Checksum (CRC)**. Each NIC contains a **MAC Addr**, storing 24 bit organisation ID & 24 bit **device ID**. Can be changed / faked.

A **switch** allows connecting many devs to the same network. It maps ports to MACs using a **Forwarding Info Base**. It can:

- **Store & Forward** - whole frame received then forwarded.
- **Cut Through** - frame forwarded when MAC read. Faster, but can forward corrupted frames.

A **Wireless Access Point (WAP)** (`802.11`) is a wireless repeater, not a router.

6.1 Topologies

- **Bus**: main coaxial cable connects hosts. Data travels up & down *bus*, with BNC terminator at both ends. Simple, slow, unreliable.
- **Ring**: phys ring to connect hosts. Each hosth as two NICs, & data flows one way. Every host point of failure, slow, unreliable.
- **Token Ring**: logical ring with **Multistation Access Unit (MSAU)**. Data flows with a **SINGLETON** token in a ring, when passed to sender its released. Fast, reliable, expensive, complex. Supports **priority** & **reservation** schemes.
- **Fiber Distributed Data Interface (FDDI)** is a dual token ring. **Wrapback** allows temp running on a single ring to avoid short circuits. Used for high-speed transfer.
- **Star** has a central switch with attached devices. Switch is point of failure.

6.2 Medium Access Control

MAC resolves *conflicts* when two devs want to access a channel. Strategies are:

- **Frame Collision** - both frames at recipient lost, must be retransmitted.
- **No Control** - let stations retransmit after collision, inefficient for many devs.
- **Round Robin** - stations take turns with **token-based** MAC systems.
- **Reservations** - stations must onbtain slot in **slotted** MAC systems.

Static Channel Alloc uses *Time Division Multiplexing (TDM)*, where trans is divided into time slots, each user transmits in turn. The trans rate is limited by the formula $\frac{\text{channel rate}}{\text{number of channels}}$. `Frequency Division Multiplexing (FDM)` divides bandwidth into freq bands, limiting each channel's bandwidth.

In **Dynamic Channel Alloc**, transmit when the channel is available. If collision occurs, they wait for a *random* amount of time before retrying (**ALOHA**). However, this method has *low channel efficiency* when there is high *contention*. The maximum efficiency in ALOHA is 18% at 50% channel load.

Slotted ALOHA improves by restricting trans to specific time slots, reducing collisions. Transmission is only allowed when the channel is idle, using **Carrier Sense Multiple Access (CSMA)** with collision detection (*CSMA/CD*) or collision avoidance (*CSMA/CA*). The maximum efficiency here is 36% at full channel load.

CSMA checks idle before transmission. Collisions may still occur due to delay. Transmission stops when collision occurs. Adds a **jamming signal** to create agreement about collision. Host must transmit long enough to know that the frame is OK. Min lengths is 2η where η is *propogation delay*. CSMA can be:

- **1-persistent** - keeps checking if free & transmits immediately.
- **Non-persistent** - if busy waits for random time before checking again & transmit immediately.
- ***p*-persistent** - keeps checking if free & transmits with probability p.

To avoid poor performance on *1-persistent*, use **binary exp backoff**. After c collisions, wait for $2^c - 1$ slots. CSMA does not *guarantee* transfer - best effort.

If η is longer than packet length, instead of having dead **carrier** bits, we can send many frames together with **frame bursting**. We can also avoid collisions with a switched topology.

To map MACs to IPs, an **addr resolution protocol (ARP)** asks the router if it has the host IP addr, in which case the host sends the router its MAC.

7 Physical Layer (L1)

Wires can be:

- **Unshielded Twisted Pair (UTP)** - twisted wires reducing interference & cross-talk.
- **Patch Panel** - socket panel leading to network switch or **private branch exchange (PBX)**.
- **Coaxial Cable** places conductors concentrically. Shielded & supports more freqs → bandwidth. High cost/m.
- **Optical Fibre** refracts light, has no interference. Low **attenuation** (lost signal), high bandwidth. Expensive.

7.1 Signals & Waves

Signals can be *analogue* or *digital*. **Baud rate** number of symbol changes per second, **bit rate** bps. We use **modulation** to encode digital info to be transmitted:

- **Baseband**: unmodified signal.
- **Broadband**: physical carrier signal encodes info; modifying *amplitude*, *frequency* or *phase*.

A **digital subscriber line (DSL)** transmits data on telephone lines. Since voice is limited to < 3kHz, we can **split** voice & data. Data is modulated & sent to **DSL Access Multiplexer (DSLAM)**. An **ADSL** modem performs modulation. Extensions such as **ADSL2** & **VDSL** improve bandwidth & bitrate. DSL download rate depends on cable length. We find this using $c = \lambda f$ to find the freq, then using the **shannon hartley theorem** max bitrate $C = B\log_2(1 + S/N)$, where B is bandwidth, S is signal power, N is noise power.

8 Linux

- **ifconfig** Displays or configures network interfaces. *e.g., `ifconfig eth0` to show interface details*
- **ip** Advanced network man. *e.g., `ip addr show` to display IP addresses, `ip link set eth0 up` to bring interface up*
- **ping** Tests net conns to a remote host. *e.g., `ping 8.8.8.8` to test connectivity*. ICMP based.
- **traceroute** Traces the route to a destination. *e.g., `traceroute example.com` to trace path*
- **netstat** Displays net conns & routing tables. *e.g., `netstat -tuln` for listening ports*
- **nslookup** Queries DNS for domain or IP information. *e.g., `nslookup example.com`*
- **dig** Advanced DNS querying tool. *e.g., `dig example.com` to get DNS details*

9 » THE QUESTION CHECKLIST «

1. Make sure *headers* are accounted for at *every level* in a calc.
2. Make sure units are sound.
3. Consider edge cases & extra requests / segments that may be sent.
4. Consider **network** (identifies a subnet) & **broadcast** (broadcast messages to all devs within the subnet) addresses when counting.