Network

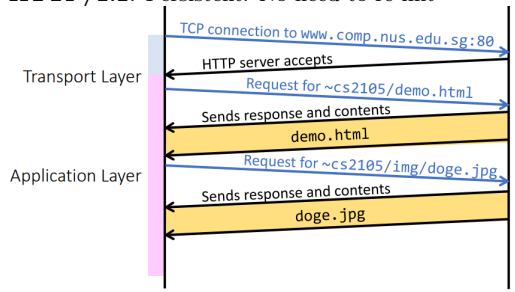
- **Protocols**: Define the structure and format of messages, and actions taken upon receiving and send messages. Eg. HTTP, FTP, SMTP, TCP, RTP
- **Network Core**: A mesh of interconnected devices.
- **Circuit Switching**: A pipe dedicated for the two parties. End to end resources dedicated to and reserved for the "call".
- **Packet Switching**: Message is broken in chunks and forwarded via available resources.
  Store-and-forward: entire packet must arrive before can be forwarded. Routing determines source-destination route.
- **Internet**: Uses packet switching.
- **Comparison: Circuit vs Packet**:

Circuit	Packet
Setup Required	Setup Not Required
Guaranteed	Best effort
Resource idle if unused	Shared
Straightforward	Requires routing, addressing, forwarding

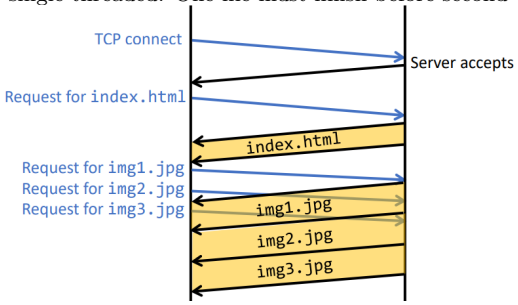
- **Network of Networks**: The Internet is a network of networks. Tier 3 ("Access ISP") connect to Tier 2 ("Regional ISP") that acts as middleman to Tier 1 ("Global ISP") which are interconnected / peered.
- **Packet loss**: Occurs when routers become overloaded. They have finite buffer. When it is exceeded, packets get discarded.
- **4 Sources of Delay**:
  End-to-end delay consist of these 4 sources of delay:
  1. Processing Delay: Check packet integrity, typically msec.
  2. Queuing Delay: Time waiting in router congestion queue.
  3. Transmission Delay: Time taken for packet to be placed onto the link. = Packet Length / Link Rate
  4. Propagation Delay: Time taken for packet to reach recipient (at speed of light). = Distance / Speed of Light.
- **Throughput**: How many bits can be transmitted from end to end.
- **5 Layer Simplified OSI**: Application, Transport, Network, Link, Physical

Application Layer

- **Application Requirements**: Different apps have different requirements of:
  1. Data integrity - transactions require full integrity, video/audio streaming no need. 2. Timing - games need data fast, normal file transfers no need so fast. 3. Throughput - video call requires some data to be effective, file transfer can afford lower throughput. 4. Security.
- **Socket**: On application ↔ transport layer. App sends/receives messages to/from its socket. Socket ≠ port! A port (transport layer) can have multiple sockets, much alike how a mailbox can be accessed by multiple people in the same household.
  Can have TCP vs UDP sockets.
- **Transfer Control Protocol TCP vs User Datagram Protocol UDP**: TCP is connection oriented, reliable, has flow control (don't overwhelm recipient), and congestion control (throttle when network is slow). UDP is unreliable / has none of the above, but has less overhead and can send quickly. Use TCP or UDP depending on what services you need.
- **HTTP**: Transfers via TCP. Client-server model.
- **HTTP/1.0**: TCP init connection, server OK, app request file, server SEND, close.
- **HTTP/1.1**: Persistent. No need to re-init



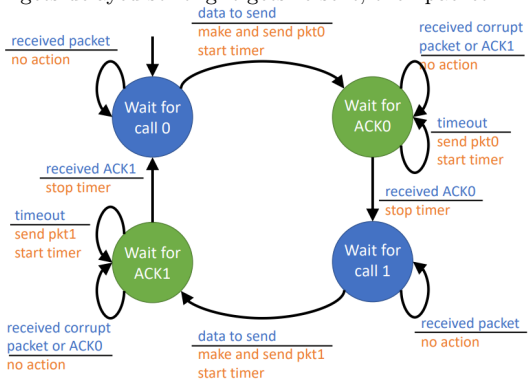
- **HTTP/1.1 Pipelining**: Adds pipelining. Allows multiple requests simultaneously over same connection. However server response is still single-threaded. One file must finish before second



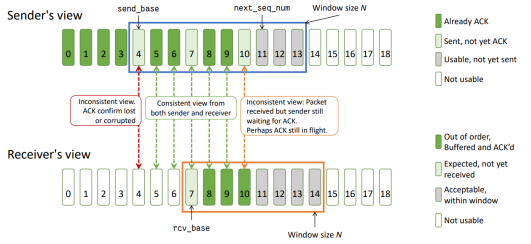
- **HTTP/2 Multiplexing**: Requests can come back in any order, even partially.
- **Domain Name System**: Translates hostname to IP address. A: Address/IP, NS: NameServer, CNAME: canonical name / alias, MX: mail exchange. DNS Runs over UDP 53.
- **NSLookup/Dig**: Shows all DNS configuration/mapping between hostname and IP.
- **Hierarchical DNS**: DNS is hierarchical. Root servers know about the .com., .org. etc. .com. knows about facebook.com. etc
- **Local DNS**: Each ISP has a DNS "default name server" that acts as cache/proxy to speedup DNS query.
- **DNS: Iterative**: DNS is iterative. Your local DNS server makes call to root, get IP addresses of .com, then you make another call to .com etc. Not recursive.
- **TCP socket**: is connection oriented, like a constant stream from socket.
- **UDP socket**: not connection oriented, messages are just independent packets that arrive.

Reliable Communication

- **Sources of data loss**: Corrupt packets, Dropped packets, Re-ordered packets, Delivery after arbitrary long delay. Need to ensure delivery and correctness, and in the same order.
- **RDT 1.0**: Perfectly reliable. Sender: wait for application data → send data. Recipient: wait for data → give data to application.
- **RDT 2.0 (Bit Errors)**: One bit can be corrupt. Recipient can check with checksum, and reply with ACK or NAK (Negative acknowledgement). However, if recipient sends corrupted NAK, sender won't know what to do. Re-sending packet is wrong as now, recipient cannot tell whether it's a new or re-sent packet.
- **RDT 2.1 (With Sequence No)**: Alternate waiting for packet 0 or packet 1. Now, client can tell whether it's resent or new packet.
- **RDT 2.2 (No more NAK)**: Replace with ACK of last correct packet instead.
- **RDT 3.0 (Lost packets)**: No re-order, but packets can be lost or arbitrarily delayed. ACK can be lost too. Sender should wait for timeout, then re-send. If receive duplicate packets, discard. However, this is still not perfect. What if packet 0 gets delayed so long it gets re-sent, then packet 1

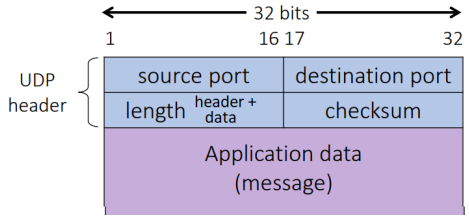


- **Stop-and-wait**: RDT uses stop and wait. Inefficient as lots of time spent waiting.
- **Alternating-bit Protocol**: Only 1 bit: 0 or 1 as packet ID.
- **Pipelined protocols**: Go-back-N, and Selective Repeat.
- **Go-Back-N**: Cumulative ACK. ACK n means all ≤ n have been received. Sender maintain sliding window and timeout for oldest unACK packet. On timeout, resend all packets in window. Recipient discard all out-of-order packets, only ACK packets that come in order.
- **Selective Repeat**: Sends ACK for each packet. Sender and recipient both maintain timer for each unACKed packet, and retransmit on timeout. Recipient needs buffer to cache out-of-order packets.



UDP

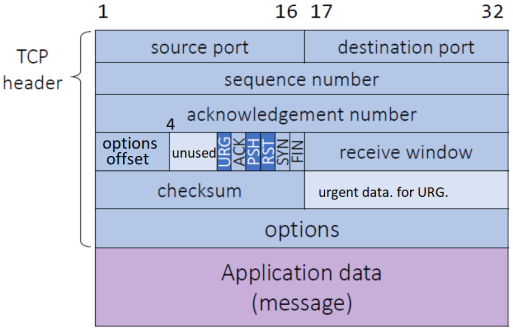
- **Connectionless De-multiplexing**: Sender creates socket with local port number. Specify destination host and port number. Recipient directs UDP segment to the socket with the specific port number.



- **Checksum**: Split data segment into 16-bit length. Sum all bits, wrap around carry, then compute 1's complement. Recipient just adds provided checksum with calculated checksum. If all 1s, then OK.

TCP

- **Connectionl-oriented De-multiplexing**: Each connection is specified with (srcIP, srcPort, destIP, destPort).

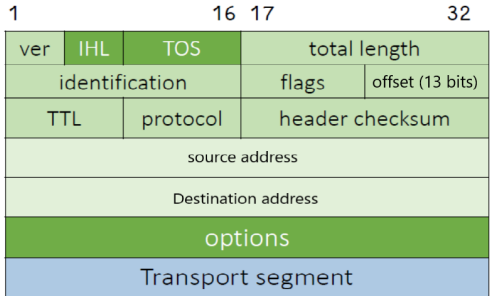


- **TCP Offset**: Size of header in number of 32-bit word. If offset ¿ 5, then options field is populated.
- **Sequence No.**: TCP Packet sequence number, in **bytes**, **not** segment.
- **TCP ACK**: Cumulative ACK, for next **byte** of data expected. Piggybacked on data. If in-order segment, waits 500ms for next segment (for performance reasons). Send ACK after time elapses.
- **Re-send**: TCP keeps only 1 timer for oldest packet, retransmits only the oldest unACK segment.
- **RTT Estimate**:
  $(1-\alpha) \cdot RTT_{hist} + \alpha \cdot RTT_{sample}$. Typical $\alpha = \frac{1}{8}$
- **RTT Std Dev**:
  $(1-\beta) \cdot RTTDev_{hist} + \beta \cdot |RTT_{sample} - RTT_{est}|$
- **Timeout**: $RTT_{est} + 4 \cdot RTTDev$. Estimation + "safety margin"
- **TCP Fast Re-transfer**: If receive 3x duplicate ACK, re-transmit without waiting for timeout.
- **TCP Syn**: During initial handshake, agree on connection parameters eg initial sequence number (need not be 0). 3-way handshake: SYN → SYN/ACK → ACK/Data.
- **TCP Fin**: Indicates that sender will not send any more data. But can still receive.

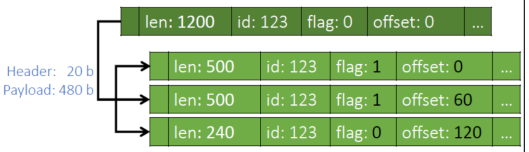
- **Receive Window**: Indicates amount of buffer space of recipient, so sender does not waste resources/flood.

Network Layer

- **Role**: Determines forwarding and routing.
- **IPv4**: 32-bit integer in dot-decimal notation. Hierarchical addressing.
- **Routing**: "Send me packets with address a.b.0.0/16". Longest (correct) prefix match. Compare target IP to advertised prefix.
- **DHCP**: Dynamic Host Configuration Protocol. Automatically assigns IP address from DHCP server. 1. Client (port 68) broadcast DISCOVER message. 2. Server (port 67) responds with OFFER. 3. Client REQUESTS for the IP address. 4. Server ACKs the assignment.
- **Subnet**: Hosts in the same subnet can access each other with just a switch. Subnet mask used to determine which subnet the IP belongs to. Subnet mask: all '1' for prefix bits, '0' for host bits. eg /23 = 11111111 11111111 11111110 00000000 = 255.255.254.0
- **NAT**: Internal private network, translated by the router into a single public IP. Maps port A internal to port B external. Hosts within NAT cannot be explicitly addressed from outside world. Needs to initiate connection from within NAT first, or port forward.



- **Datagram Fragmentation (IP)**: Each link determines its own Max Transmission Unit (MTU). Too large $\implies$ router splices it. Split original payload into multiple parts each with new IP header. Same id for all parts, length = how much data incl header, flag = 1 if have more parts, offset = bytes/8 from start of data.



- **Internet Control Message Protocol (ICMP)**: $\neq$ IP. Datagram similar to IP, protocol = 0001, additional type/code like 8/0 = echo request, 3/1 = dest host unreachable
- **Intra-AS routing**: Link State Algorithm - every router has global view, can run Dijkstra's. Distance Vector Algorithm: Only know connected neighbours, constantly update each other. Decentralised, self-terminating, iterative, async.
- **Distance Vector Algorithm (Poisoned Reverse)**: Uses Bellman Ford's. Each node stores cost to every other node. Sends neighbour x all the correct distance except those for which the neighbour is the next hop. Sends ∞ instead for those vertices. Solves counting to infinity for only 2-node cycle if a node goes down. Used by Routing Information Protocol (RIP).
- **Border Gateway Protocol (BGP)**: Inter-AS route to AS whose gateway has least cost.

Security & Cryptography

- **Integrity**: Calculate SHA1 and append to plaintext.
- **MAC**: Integrity + Authenticity. Use a key as part of digest. e.g. HmacSHA1. Affirms to receiver the message's origin. Message becomes Message + Hash(message + key).
- **Digital Signature**: MAC + Verifiable by third party. Use private/public key instead of shared symmetric key. Message become Message + K_A^-(Hash(message)). Note: not hash(key(msg)), because Public Key encryption is slow.

Link Layer

- **CRC**: With generator of r+1 bits (agreed beforehand), append r 0s to data. Take remainder of Data / generator. e.g. Data = 1101001, G = 1001. Take Data + "000" = 1101001000, then divide by G and take remainder. Division is XOR. Append remainder to data, and send. Receiver do same thing, 0 = no error.
- **Channel partitioning protocols**: 1. Time-Division Multiple Access (takes turns).
  2. Frequency-Division Multiple Access (different frequency).
  3. Code-Division Multiple Access (different orthogonal codes across all frequencies at the same time).
  4. Take turns with master-polling / pass the bucket
  5. Random: Slotted ALOHA. Send at any time and listen for collision. If collide, send again in next slot with fixed probability p.
  6. ALOHA: Send whenever, if collide, retry randomly with probability p.
  7. CSMA: Sense availability before transmitting. Can still collide due to propagation delay.
  8. CSMA/CD: Stop once collision is detected and re-transmit after random time. After n^{th} collision, wait $\{0, 1, \ldots, 2^{n-1}\} * 512$ bit-time. 512 because ethernet minimum frame = 64 bytes.
  9. CSMA/CA: Avoid collision due to hidden node problem by using request to send. Receives clear to send. Send. Receive ACK from recipient.
- **Ethernet Framing**:

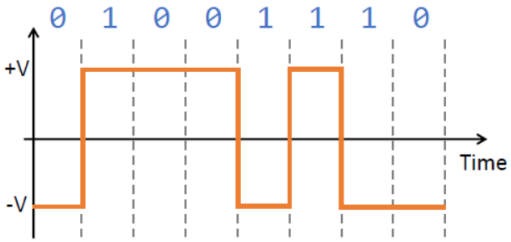
Preamble: 7 bytes of 10101010, + 10101011 to sync clock.
MAC: 6 bytes each.

Local Area Network

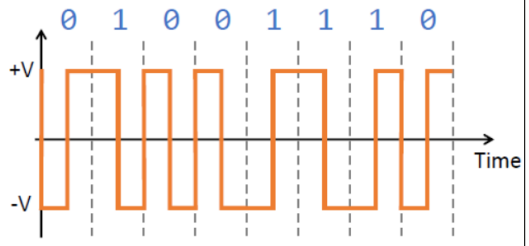
- **MAC Address**: Unique to adapter, when receive frame, check matches MAC address. Permanent address.
- **Address Resolution Protocol (ARP)**: Maps IP address to MAC address.
- **Sending message**: If A knows B's MAC address, just send frame to B's MAC address directly. If don't know, send a broadcast ARP query packet with B's IP address.
- **Sending message to different subnet**: If B is in different subnet (determined by IP), send packet to router, router replace MAC with its own, and forward to external network.
- **Collision**: Ethernet has no handshaking. If in Bus topology, can cause collision. Also no ACK/NAK, so lost data needs to be accounted for in higher layer RDT. Uses CSMA/CD to resolve collision.
- **Switch**: Network layer, has no IP address (transparent to hosts), provides star topology. Each host has dedicated link to switch, and switch selectively forwards frames to correct link.
- **Switch forwarding table**: has a similar MAC to interface mapping, learnt through broadcasting/forwarding messages.
- **Wireless LAN**: Difficult to detect collision. Uses CSMA/CA, uses link-layer ACK. Other devices, when detecting the CTS, defers sending until detecting the ACK from router. Then resume countdown from binary exponential backoff.

Physical Layer

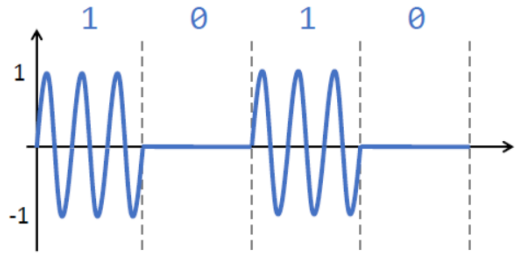
- **Non-return to Zero**: Unipolar scheme. +V = 1, 0V = 0. (Doesn't return to 0 between signals)
- **Non-return to Zero Level**: Polar scheme. Same as NRZ. But uses -V for 0 and +V for 1.
- **Non-return to Zero Invert**: Polar scheme. Same as NRZL. 0 = no inversion, 1 = inversion.



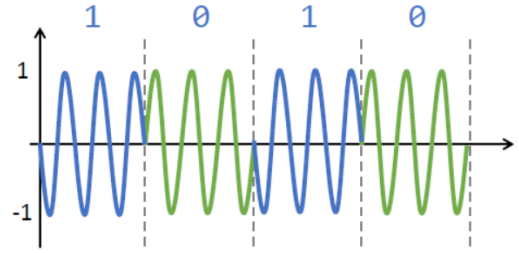
- **Bit slip**: The effect where clock not synced, multiple 1s later, the clock is off. Don't know n or n+1 bits.
- **Return to zero**: Always return to zero before changing new signal. [+V, 0], [-V, 0]. [-V, 0], ...
- **Manchester**: 0 = +v $\implies$ -v. 1 = -v $\implies$ +v.
- **Differential Manchester**: 0 = no inversion. 1 = inversion.



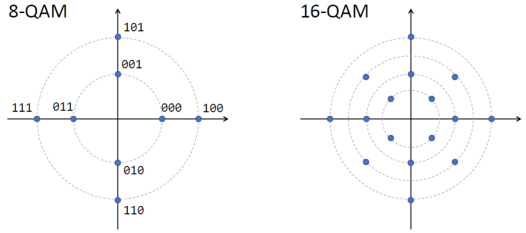
- **Fourier Transform**: Can decompose an analog signal into its constituent pure signals.
- **Nyquist Bitrate Formula**: Only for noise-less channel. Theoretical max bitrate: $2B * log_2 L$ where B is bandwidth (MHz) and L is number of signal levels. E.g. 2 signal levels in manchester. You can set any number of signal levels, only limited by how much noise you have.
- **Shannon Channel Capacity**: Noisy channel. Theoretical max bitrate: $B * log_2(1 + SNR)$ where B is bandwidth (MHz) and SNR is signal to noise ratio. E.g. 2 signal levels in manchester. You can set any number of signal levels, only limited by how much noise you have.
- **Analog Encoding**: Wave is $A \sin(2\pi ft + \phi)$. Can change A: amplitude, f: frequency, ϕ phase.
- **Amplitude Shift Keying**:



- **Frequency Shift Keying**: High freq = 1, low fre
- **Phase Shift Keying**:



- **PSK on Constellation Diagram**: Can be visualised as a circle, angle = phase change, distance = amplitude (constant for *PSK, varying levels with QAM = ASK+PSK)



- **Baud rate**: number of signal units per sec.
- **Bit rate**: number of bits per sec = baud rate * bits per signal. e.g. 2^3-QAM = 3 bits.