Common Conversions

Time: 1 second = 1,000 milliseconds → 1ms = 1,000 nanoseconds
Data Transfer: 1 gigabit (Gb) = 1,000 megabits (Mb) → 1 megabit (Mb) = 1,000 kilobits (Kb) → 1 kilobit (Kb) = 1,000 bits (b)
Storage: 1 gigabyte(GB) = 1,024 megabytes (MB) → 1 megabyte (MB) = 1,024 kilobytes (KB) → 1 kilobyte = 1,024 bytes

Section 1: Transport Layer

User Datagram Protocol (UDP)

UDP: Loss tolerating, low-latency connections between applications
UDP Header (8 bytes): SRC PORT # | DEST PORT # | LEN (BYTES) | CHECKSUM | PAYLOAD
UDP Checksum: Binary addition of UDP segment content → complement bits to get checksum

Reliable Data Transfer (RDT) – Stop and Wait Protocols

RDT 1.0: Transfer over a perfectly reliable channel. No acknowledgement, no flow control.
RDT 2.0: Transfer over a channel with bit-errors. Use of **ARQ: Automatic Repeat Requests** i.e. ACK / NAK
RDT 2.1: 2.0 + Includes sequence numbers #0 and #1. If expected seq != seq, then duplicate packet.
RDT 2.2: 2.1 + Without NAKs. Sequence numbers can already detect duplicate, no need for NAKs.
RDT 3.0: Channel w/ bit-errors and packet loss. Use of **Time-based Retransmission**. Retransmit for both errors + loss.

Reliable Data Transfer (RDT) – Pipelined Protocols

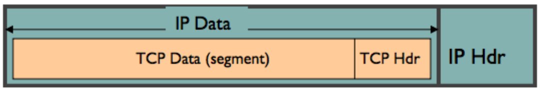
Allows for multiple "in-flight", un-ACK'd packets. Increases utilisation of a link.

Go-Back-N (GBN) Sender continues to send pkts specified by window-size N without receiving ACKs.	**Selective Repeat (SR)** Receiver individually ACKs all received pkts. Buffers packets for eventual in-order delivery to the upper layer.
• Sender window size N of consecutive un-ACK'd packets. • On timeout/loss of packet P: Receiver – discards P + resend ACK of last successful pkt. Sender - retransmit all pkts of higher seq# in window. • On success: Advance send_base	• Sender window size N consecutive seq #s. • On timeout/loss of packet P Receiver: buffer the out of order pkt. Sender: retransmit P only • On success: Send P + all following in-order packets Advance send_base

Main difference: SR ACKs every packet, buffers them and only re-transmits the timeout/lost ones vs. GBN retransmits all.

Transmission Control Protocol (TCP)

TCP Header (20 bytes): UDP fields + seq# + ack# + RWND #bytes + connection establishment + teardown + options.
IP Packet encapsulates a TCP packet.



IP Packet: No bigger than **Max Transmission Unit (MTU)**
TCP Data/Segment: No more than **Max Segment Size (MSS)**
MSS = MTU – IP Header – TCP header

TCP Sender: Sends packet of <u>SEQ = X</u> | <u>Data Len = B bytes</u> i.e. [X , X + 1 . . . X + B-1]
TCP Receiver: If data in previous packet has been received, send <u>ACK = X+B</u> (expected seq # of next packet)

TCP Features

Fast Retransmission: Receiver sends 3 dupACKs for lost packet to trigger early retransmission.
- No need to wait for timeout on sender side, as timeout periods are often long

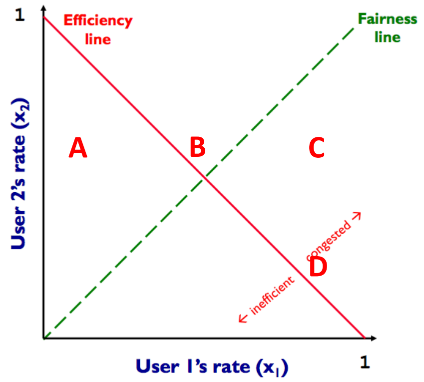
Flow Control (RWND): Receiver controls the sender, so the sender won't overflow the RWND buffer.
- <u>Receiver Advertised Window Size (RWND)</u>: Advertises the available receiver buffer space in the TCP header.
- <u>Result:</u> Sender limits the amount of un-ACK'd data to receiver's RWND buffer.

Connection Management:
- <u>Establishment</u>: SYN → SYN-ACK → ACK + DATA
- <u>Teardown</u>: FIN → ACK-FIN → ACK → WAIT / RETRANSMIT ACK → CLOSE CONNECTION

Congestion Control: Needed if a network node is taking in more data then it can output, leading to collapse.
- <u>Congestion Window (CWND)</u>: How many bytes can be sent without overflowing a router?
  Sender varies the CWND size to control the transmission rate
- <u>Sending Rate</u>: **S = W*MSS / RTT**
- <u>Average Sending Rate</u>: **Ave = 0.75W*MSS / RTT** → RTT is inverse to Transmission Rate
- <u>Operations</u>:
  - CWND < SSThresh | **Slow-Start**: CWND = CWND + MSS (exponential increase)
  - CWND >= SSThresh | **Congestion Avoidance / AIMD**: CWND = CWND + MSS/CWND (increase 1MSS per RTT)
  - DupACK / Loss occurs | **SSThresh & CWND** = CWND / 2
  - Timeout occurs | **SSThresh** = CWND/2 **CWND** = 1 MSS

TCP Flavours
TCP Reno: The standard one as described above
TCP Tahoe: CWND = 1 && Slow Start immediately for both DupACK and Timeout
TCP New Reno: Standard + improved fast recovery

TCP Fairness
Fairness Goal: **B/X** bandwidth for each TCP session | B = Bandwidth for bottleneck link. X = # TCP sessions.



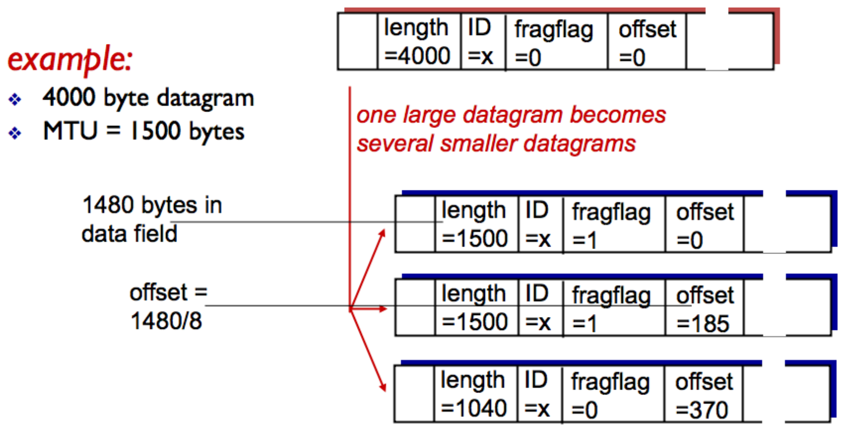
A = Inefficient / Not fair ($x_1 + x_2 = 0.7$)
B = Efficient / Fair ($x_1 + x_2 = 1$)
C = Congested / Not fair ($x_1 + x_2 = 1.2$)
D = Efficient / Not Fair ($x_1 + x_2 = 1$)

AIAD: Add increase $X_{1,2}$ | Add decrease $X_{1,2}$
- Does NOT converge to fairness

AIMD: Add increase $X_{1,2}$ | Mult decrease $X_{1,2}$,
- Converges to fairness (rates will equalise eventually)

Limitations of TCP Congestion Control
Different RTTs: Throughput is inverse to RTT. Different RTT's = lead to unfairness.
Loss NOT due to congestion: TCP may cut CWND for packet errors, NOT packet loss.
Disadvantage for short flows: Short flows may never leave Slow-Start phase = never attain fair share.
TCP fills up queues: If a single flow deliberately overshoots capacity, causing large delays for everyone.
Cheating / bypass CWND: Opening up many connections / use large initial CWND / increase CWND by more than 1MSS/ per RTT

Section 2: Network Layer
Internet Protocol (IP) Packets
IP Fragmentation Reassembly: A large IP datagram may be divided, as it can't exceed the **Maximum Transmission Unit (MTU)**
- MTU = Size of the largest network layer data that can be sent in a single network transaction.



20 byte header in each packet.

Original packet (4000 bytes)
= 3980 payload + 20 header

Frag pkt #1 (1500 bytes) | Offset = 0 (start of OG)
= 1480 payload + 20 header

Frag pkt #2 (1500 bytes) | Offset = 1480 / 8 = 185
= 1480 payload + 20 header

Frag pkt #3 (1040 bytes) | Offset = 2960 / 8 = 370
= 1020 payload + 20 header

Subnets + IPv4
IP Address	223.1.1.2	11111101 00000001 00000001 00000010
Subnet Mask	255.255.255.0	11111111 11111111 11111111 00000000
		IP & Subnet Mask
Network part	223.1.1.0	11111101 00000001 00000001 00000000
		Remainder Bits
Host part	0.0.0.2	00000000 00000000 00000000 00000010

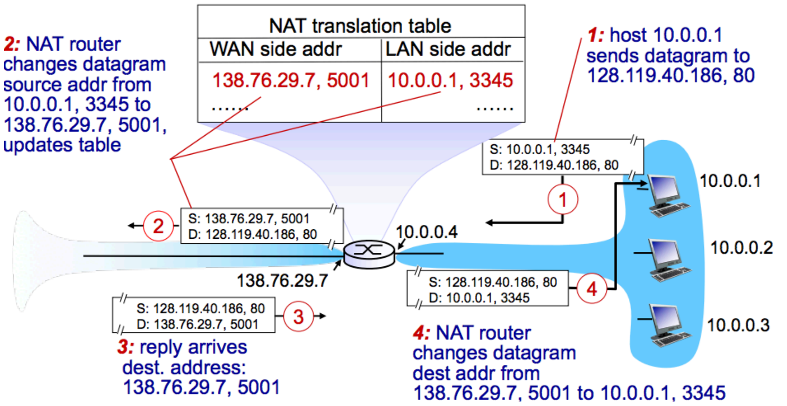
IP Address = Network Part + Host Part
Network Part = Bottom of the range
Range of Subnet = Network Part (bottom) + SIZE of Subnet Mask
Size = Complement of Subnet Mask

Dynamic Host Configuration Protocol (DHCP)

DHCP: Allows a host to dynamically obtain its IP from a network server when it joins a network.

- Allows IP reuse. DHCP server usually on a router.
- <u>Steps</u>:
  (1) Client broadcasts DHCP request (2) DHCP Server responds with offer
  (3) Client requests: **[IP] [Address of 1st hop router] [Address of DNS server]**
  encapsulates request in UDP → encapsulates UDP in Ethernet Frame → broadcasts Frame on LAN
  (4) Server picks up frame: demux to DHCP request → formulates ACK with requested items → encapsulates + sends
  (5) Client demux to DHCP and gets the information
- <u>DHCP Loophole</u>: DoS attack by exhausting available pool of IP addresses in LAN | Spoof as DHCP server.

Network Address Translation (NAT) and Private Addresses



NAT allows a device such as a Router, to **act as an agent** between the Internet (public network) and the local (private) network.
Only a single IP address is required to represent an entire group of computers.

Advantages:
- Single IP address for multiple devices
- Change addresses without notifying outside world vice versa.

Disadvantages:
- Violates <u>end-to-end agreement</u>
- Requires <u>recalculation of checksum</u>
- Some apps embed IP / Port #

IPv6 Addressing

IPv6: Helps speed up processing / forwarding + 32-bit addresses might run out soon.
IPv6 Header (40 bytes): No fragmentation allowed, identify priority packets, flow label, removed checksum.
Tunnelling: IPv6 packets carried as a payload in IPv4 packets through IPv4 routers.

Virtual Circuit Network

VC Network: provides a connection-based network layer service. Uses signalling protocol to setup/maintain circuits.
VC SETUP: (1) <u>Source</u>: Sends setup msg with dest address.
(2) <u>Intermediates</u>: Choose VC (from lowest) → Determine outgoing interface from routing table → create VC table entry → forward setup
(3) <u>Setup reaches dest</u>: Dest chooses incoming VC # → chosen VC = outgoing VC of all routers except last → Send ACK to source
(4) <u>Acknowledgement</u>: Intermediate routers along the reverse path complete their VC tables (filling in outgoing VC)

Forwarding: Within router (input link → output link) | **Routing**: Establishing a path/route from source → dest.

Routing Protocols #1: Intra-domain Routing (LINK STATE / DISTANCE VECTOR)

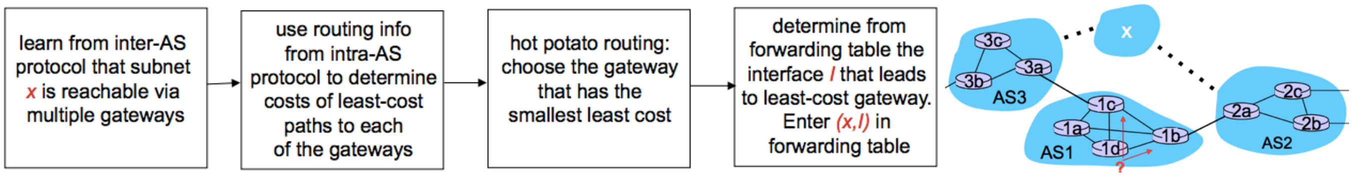
Link State Routing (Global): All routers maintain complete topology + know cost of each link in the network.
- <u>How it works</u>: **(1) Link State Advertisement Flooding (2) Path calculation with Djikstrasb**
- <u>Challenges</u>: Packet loss, scalability, infinite loops, oscillations (cost changing continuously)

Distance Vector Routing (Decentralised): Routers only know neighbours + link cost to neighbours.
- <u>How it works</u>:
  (1) Each router initialises its DV table based on link costs to immediate neighbours + sends its DV to the neighbours.
  (2) Neighbours process the DV and repeats STEP #1 until the iterative process converges to a set of shortest paths.
  (3) Each node then waits for changes in their local link cost or msg from neighbours.
  (4) If change occurs → recompute costs in DV and notify neighbours if anything changes.
- **Count to Infinity Problem**: Occurs when node/link is broken.
  Nodes incorrectly update their DV table + increase cost for the link until updates propagate through the network and the link cost = infinite.
- **Poisoned Reverse Rule**: Routers actively advertise certain links are unreachable. I.e. cost = infinite
  This will significantly increase number of routing advertisements made in the network

	Link State	**Distance Vector**
Message Complexity	N nodes / E edges = O(N*E) messages sent	Exchange between neighbours only.
Speed of Convergence	O(N^2) algorithm | relatively fast	Convergence time varies Count to Infinity / Routing Loops may occur
Robustness	LS node can advertise incorrect LINK cost. Each node computes only its own table.	DV node can advertise incorrect PATH cost. Each node's table is used by others, errors propagate through the network.

Routing Protocol #2: Inter-domain Routing Protocol

Inter-domain Routing: AS establishing routes with other domains + wants to control who can route through their network.
Gateway Routers: Routers that act as the "edge" of an AS which links to another AS in the internet.
<u>How it works</u>: Scenario: Router in AS1 needs to determine which gateway to forward packet to, so it can reach subnet X



Section 3: Link Layer

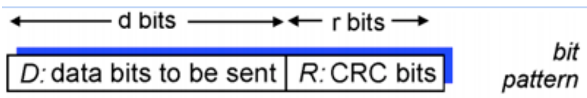
Data-Link-Layer: responsible for transferring a datagram from one node to a physically adjacent node over a link.
Framing, link access: Encapsulate datagram into frame, add header/trailer, providing channel access + MAC address to identify.
Reliable delivery between nodes: Low bit-error in some links i.e. Fiber. High error rates in wireless links.
Flow Control: Pacing between adjacent sending and receiving nodes
Error Detection: Errors caused by signal attenuation (reduction of signal strength during transmission) and noise.

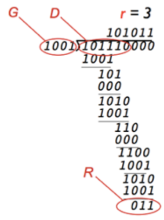
- Receiver signals for retransmission or drops the frame.

Error Correction: Receiver identifies and corrects bit-errors without needing retransmission.
Half-Duplex and Full-Duplex: DUPLEX = ability for two devices to communicate at the same time.

- Wireless WiFi = half-duplex | Wired LAN = full-duplex

Error Detection: Cyclic Redundancy Check (CRC)



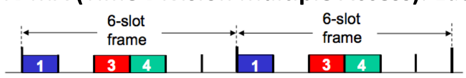


Multiple Access Links / Multiple Access Protocols

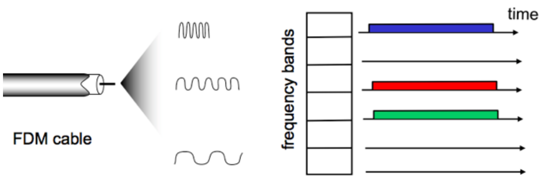
Multiple Access Problem: How to coordinate access from multiple sending/receiving nodes to a shared broadcast channel?
Collision occurs if nodes receive two or more signals at the same time, losing frames + wasted channel during collision interval.
Multiple Access Protocols help determine how nodes share a channel + when they can transmit

Protocols #1: CHANNEL PARTITIONING

TDMA (Time Division Multiple Access): Each station gets a fixed length slot. Unused slots go idle.



FDMA (Frequency Division Multiple Access): Channel is divided into frequency bands, each station is assigned a frequency.



Protocols #2: RANDOM ACCESS (channel not divided, allow collisions to occur + recover from collisions)

Slotted ALOHA: When a node obtains a frame, transmit it in the next slot.

- <u>Collision</u>: The node retransmits the frame in each subsequent slot with P probability.
- <u>Max Efficiency</u>: Useful **37%** of the time.

Pure un-slotted ALOHA: When the 1st frame arrives, transmit immediately. Simpler with no synchronisation.

- <u>Collision</u>: probability increases. **frames sent at t_0 collides with frames sent in [$t_0 - 1$, $t_0 + 1$]**
- <u>Max efficiency</u>: Useful **18%** of the time.

Carrier Sense Multiple Access (CSMA): Nodes sense / listen before they transmit

- <u>Channel IDLE</u>: Transmit frame
- <u>Channel BUSY</u>: Defer transmission
- <u>Collisions</u>: still occur as two nodes may not hear each other's transmissions due to propagation delay.
- Distance + Propagation delay affect collision probability. CSMA reduces but NOT eliminates collisions.

CSMA / Collision Detection (CSMA / CD): Nodes detect collisions by sensing transmissions while transmitting a frame

1. NIC receives datagram from network layer: creates a frame + encapsulates datagram
2. **IF** NIC senses channel is IDLE: starts frame transmission.
   ELSE wait until channel is IDLE.
3. **IF** NIC transmits frame without detecting another transmission, transmission is complete.
   ELSE abort transmission + send jam signal to ensure all receivers detect the collision.
4. After abortion, NIC enters **Exponential Back-off**:

- After m_{th} collision, the NIC chooses a random K from { 0 . . . 2^m-1 }.
- NIC waits K*512bit times then returns to STEP #2
- More collisions = longer backoff.

- For CSMA/CD to work, place restrictions on **Minimum Frame Size / Max Distance**.

Protocols #3: TAKING TURNS

Taking Turns: Nodes take turn, but nodes with more to send can take longer turns.
Polling Protocol: Base station controls entire channel + variable sized frame content.

- How it works: (1) Base sends poll packet to node to allow transmission (2) Control token is passed to each node.
- Concerns: Token overhead + single point of failure.

LAN + ARP Resolution

Address Resolution Protocol (ARP): Helps determine an interface's MAC address and IP address.
ARP Table: Entry = **<IP address, MAC address, TTL>**
SCENARIO #1: Send datagram within LAN

- SRC broadcasts ARP query containing DEST IP. → all nodes receive broadcast
- DEST replies to broadcast with DEST MAC. → frame sent to SRC MAC
- SRC caches IP-to-MAC pair in its ARP table until TTL has expired.

SCENARIO #2: Send datagram outside of LAN

- SRC must know: (1) DEST B's IP (2) 1st hop router R's IP (3) 1st hop router R's MAC
- SRC creates datagram **D[src=A, dest=B]** → encapsulates in frame with dest = R_MAC & R_IP
- SRC sends frame to R → R receives frame, detaches datagram from the frame.
- R creates link-layer frame [dest=B_MAC | data=A datagram] → forwards to DEST B → DEST B detaches datagram

Ethernet

Ethernet Topology BUS: Old topology, where all nodes can collide. Used CSMA/CD.
Ethernet Topology STAR: Active switch in the centre, each spoke runs a separate Ethernet protocol, so no nodes collide.

- No sharing, no CSMA/CD

Ethernet Characteristics: Connectionless, Unreliable (NIC's drop frames, unless upper-layer protocol provides recovery).

Ethernet Switches

Link-Layer-Devices: Stores + Forwards Ethernet frames.

Switches: Multiple simultaneous transmissions + forwarding table

- Switches/hosts have a dedicated link, so the link has no collisions.
- Each switch has a Switch Table: **<MAC address of host, interface of host, TTL>**

MAC addr	interface	TTL
A	*1*	*60*

Switches: **Self Learning**: Switch learns which hosts can be reached through which interfaces

1. Record the MAC address of sender + incoming link (interface)
2. Index the switch table using the MAC address.
   IF entry found { IF same port, drop | ELSE forward to interface indicated by entry }
   ELSE { floor / forward to all interfaces except arriving }

Switches: **Multiple interconnected switches**

- Self-Learning used again to know where to forward frames to through other switches.

Wireless

Frequency = C / λ , where C = speed of light | λ (lambda) = wavelength
WaveLength = **C / f** , where C = speed of light | f = frequency

Signal To Noise Ratio (SNR): Ratio between max signal strength + noise present in the connection. High SNR = better signal.
Bit Error Rate (BER): #bit errors per unit of time.
SNR vs BER trade-off: Given physical layer, aim is to increase SNR / decrease BER.
Given SNR, choose physical layer that meets BER requirements + giving highest throughput.

Hidden Terminal/Node Problem: Each node within communication range of AP, but nodes cannot communicate with each other as they do not have a physical link.
Exposed Terminals: Is when a node is prevented from sending packets to other nodes because a neighbouring one is making a transmission (due to carrier sense – verifying absence of transmissions before sending).

Code Division Multiple Access (CDMA): Simultaneous communication between multiple nodes

CDMA is a channel access method, using a unique code / chipping sequence assigned to each user.

- Each node uses the same frequency, but has a unique chipping sequence to encode data.

CDMA Encoding / Decoding:

- Assume signal #1 = (1, 0, 1, 1) | chipping sequence c_M = (1, -1)
  signal #2 = (0, 0, 1, 1) | chipping sequence c_M = (1, 1)
- **Encoded Signal = (original data) modulated by (chipping sequence)**
  If two signals at a point are in a phase, they add to give TWICE the amplitude of each signal, then subtract to give a signal that is the difference of the amplitudes. Assuming that both are transmitting simultaneously:

Step	Encode sender0	Encode sender1
0	code0 = (1, –1), data0 = (1, 0, 1, 1)	code1 = (1, 1), data1 = (0, 0, 1, 1)
1	encode0 = 2(1, 0, 1, 1) – (1, 1, 1, 1) = (1, –1, 1, 1)	encode1 = 2(0, 0, 1, 1) – (1, 1, 1, 1) = (–1, –1, 1, 1)
2	signal0 = encode0 ⊗ code0 = (1, –1, 1, 1) ⊗ (1, –1) = (1, –1, –1, 1, 1, –1, 1, –1)	signal1 = encode1 ⊗ code1 = (–1, –1, 1, 1) ⊗ (1, 1) = (–1, –1, –1, –1, 1, 1, 1, 1)

  Because signal #1 and #2 are transmitted at the same time into the air, they add to produce the raw signal:
  (1, -1, -1, 1, 1, -1, 1, -1) + (-1, -1, -1, -1, 1, 1, 1, 1) = **(0, -2, -2, 0, 2, 0, 2, 0)**
- **Decoding = inner product (summation of bit-by-bit product) of encoded signal and chipping sequence**
  The receiver extracts the raw signal for a sender by combining the sender's chipping sequence code with the raw signal.

Step	Decode sender0	Decode sender1
0	code0 = (1, –1), signal = (0, –2, –2, 0, 2, 0, 2, 0)	code1 = (1, 1), signal = (0, –2, –2, 0, 2, 0, 2, 0)
1	decode0 = pattern.vector0	decode1 = pattern.vector1
2	decode0 = ((0, –2), (–2, 0), (2, 0), (2, 0)) · (1, –1)	decode1 = ((0, –2), (–2, 0), (2, 0), (2, 0)) · (1, 1)
3	decode0 = ((0 + 2), (–2 + 0), (2 + 0), (2 + 0))	decode1 = ((0 – 2), (–2 + 0), (2 + 0), (2 + 0))
4	data0=(2, –2, 2, 2), meaning (1, 0, 1, 1)	data1=(–2, –2, 2, 2), meaning (0, 0, 1, 1)

- **Post-decoding:** All values > 0 are interpreted as 1, while all values < 0 are interpreted as 0.
  E.g. if the extracted signal was (2, -2, 2, 2) → translate to (1, 0, 1, 1)

IEEE 802.11 Wireless LAN

Steps to send data over Wireless LAN network (think of procedure connecting to WiFi):

1. <u>Probing</u>: sending out a probe request on multiple channels that specifies an SSID (searching for a network like in WiFi)
2. <u>Authentication</u>: WEP/WPA/WPA2 , public/shared key authentication
3. <u>Association</u>: Finalise security + bit rate options + establish data link between LAN client and Access Point.

Passive Scanning: Beacon frames are sent from Access Points → then association request from client occurs
Active Scanning: Probe request sent from client + response from AP → then association request from client occurs

Collision Avoidance: RTS-CTS: Request to Clear / Clear to Send

- Sender transmits small RTS packets to broadcasting signal. RTS's may collide but they're short collisions.
- AP broadcasts CTS in response to RTS
- CTS is heard by all nodes → sender transmits data frame + other nodes defer transmission.

Wireless Advanced Features

- <u>Rate Adaptation</u>: Dynamically change transmission rate to adapt to current wireless channel conditions.
- <u>Power Management</u>: Nodes telling AP that they are sleeping until next beacon frame.

802.15: Personal Area Network (e.g. IoT) evolved from Bluetooth

- Less than 10m in diameter + <u>Ad-Hoc</u> (no need for infrastructure)
- <u>Replacement</u> for cables (mouse, keyboard etc.).
- <u>Master/Slave</u>: Slaves request permission to send to master.