**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)**

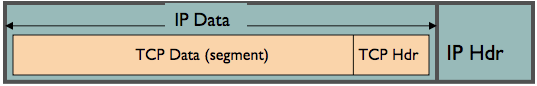
**TCP Header** **(20 bytes):** UDP fields + seq# + ack# + RWND #bytes + connection establishment + teardown + options.

**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 SEQ = X | Data Len = B bytes i.e. [ X , X + 1 . . . X + B-1 ]

**TCP Receiver**: If data in previous packet has been received, send ACK = X+B (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.

* Receiver Advertised Window Size (RWND): Advertises the available receiver buffer space in the TCP header.
* Result: Sender limits the amount of un-ACK’d data to receiver’s RWND buffer.

**Connection Management**:

* Establishment: SYN 🡪 SYN-ACK 🡪 ACK + DATA
* Teardown: 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.

* Congestion Window (CWND): How many bytes can be sent without overflowing a router?  
  Sender varies the CWND size to control the transmission rate
* Sending Rate: **S = W\*MSS / RTT**
* Average Sending Rate: **Ave = 0.75W\*MSS / RTT** 🡪 RTT is inverse to Transmission Rate
* Operations:
  + 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**

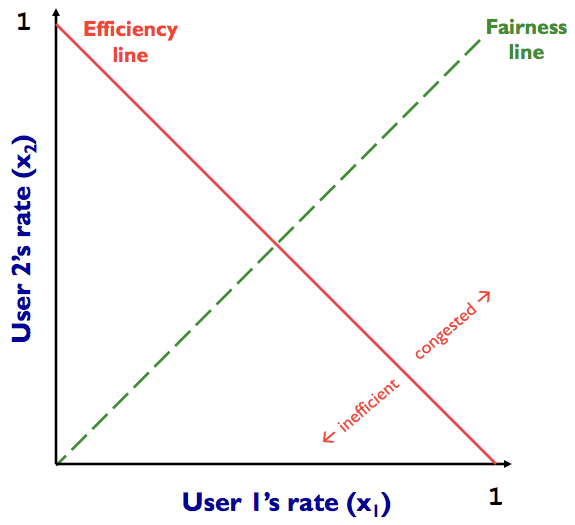
**TCP Reno:** The standard one as described above

**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 (x1 + x2 = 0.7)

**B** = Efficient / Fair (x1 + x2 = 1)

**C** = Congested / Not fair (x1 + x2 = 1.2)

**D** = Efficient / Not Fair (x1 + x2 = 1)

**AIAD**: Add increase X1,2 | Add decrease X1,2

* Does NOT converge to fairness

**AIMD:** Add increase X1,2 | Mult decrease X1,2,

* Converges to fairness (rates will equalise eventually)

**C**

**B**

**A**

**D**

**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

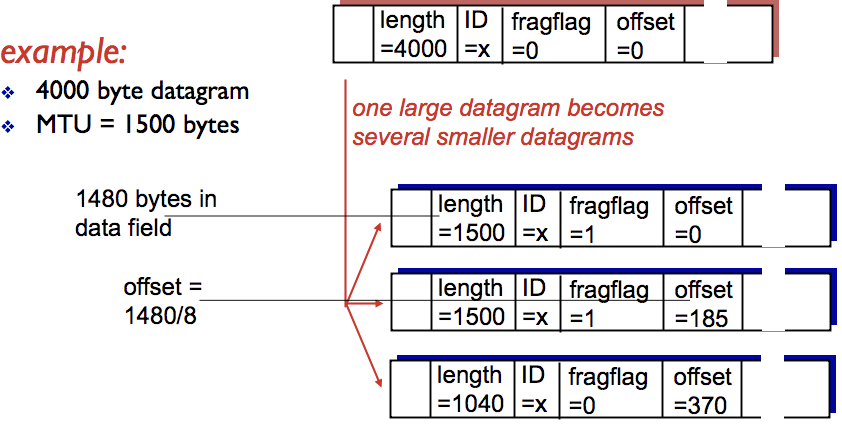
Frag pkt #2 (1500 bytes) | Offset = 1480 / 8 = 185

= 1480 payload + 20 header

Frag pkt #3 (1040 bytes) | Offset = 2960 / 8 = 370

= 1020 payload + 20 header

**Original 3980 bytes = 1480 + 1480 + 1020**



**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.
* Steps:  
  (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
* DHCP Loophole: DoS attack by exhausting available pool of IP addresses in LAN | Spoof as DHCP server.

NAT allows a device such as a Router, to **act as an agent** between the Internet (public network) and the local (private) network.

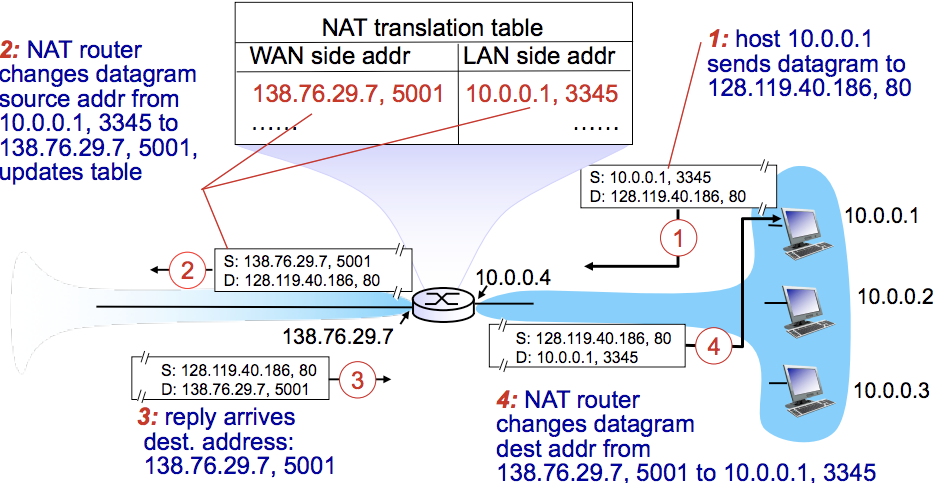
Only a single IP address is required to represent an entire group of computers.

Advantages:

* Single IP address for multiple devices
* Change addresses without notifying outside world vice versa.

Disadvantages:

* Violates end-to-end agreement
* Requires recalculation of checksum
* Some apps embed IP / Port #

**Network Address Translation (NAT) and Private Addresses**  


**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) Source: Sends setup msg with dest address.  
(2) Intermediates: Choose VC (from lowest) 🡪 Determine outgoing interface from routing table 🡪 create VC table entry 🡪 forward setup

(3) Setup reaches dest: Dest chooses incoming VC # 🡪 chosen VC = outgoing VC of all routers except last 🡪 Send ACK to source

(4) Acknowledgement: 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.

* How it works: **(1) Link State Advertisement Flooding (2) Path calculation with Djikstrasb**
* Challenges: Packet loss, scalability, infinite loops, oscillations (cost changing continuously)

**Distance Vector Routing (Decentralised)**: Routers only know neighbours + link cost to neighbours.

* How it works:  
  (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

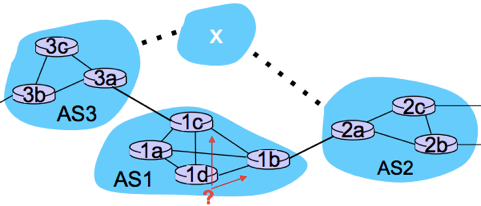
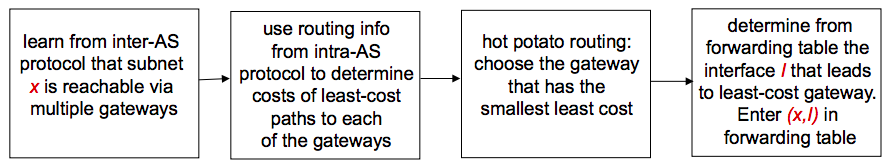
|  |  |  |
| --- | --- | --- |
|  | **Link State** | **Distance Vector** |
| **Message Complexity** | N nodes / E edges = O(N\*E) messages sent | Exchange between neighbours only. |
| **Speed of Convergence** | O(N2) algorithm | relatively fast | Convergence time varies  Count to Infinity / Routing Loops 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.

How it works: 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

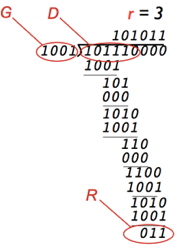
**Error Detection**: Errors caused by signal attenuation (reduction of signal strength during transmission) and noise.

* Receiver signals for retransmission or drops the frame.

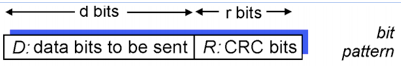
**Error Correction**: Receiver identifies and corrects bit-errors without needing retransmission.

**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**

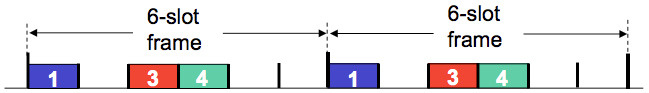
**Multiple Access Problem:** How to coordinate access from multiple sending/receiving nodes to a shared broadcast channel?

**Collision** occurs if nodes receive two or more signals at the same time, losing frames + wasted channel during collision interval.

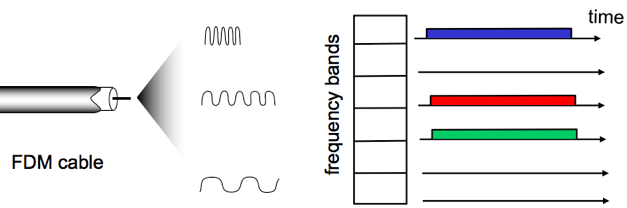
**Multiple Access Protocols** help determine how nodes share a channel + when they can transmit

**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.

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

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

* Collision: probability increases. **frames sent at t0 collides with frames sent in [ t0 – 1** **, t0 + 1 ]**
* Max efficiency: Useful **18%** of the time.

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

* Channel IDLE: Transmit frame
* Channel BUSY: Defer transmission
* Collisions: 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 mth collision, the NIC chooses a random K from { 0 . . . 2m-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.

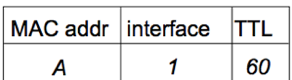
**Ethernet Topology STAR**: Active switch in the centre, each spoke runs a separate Ethernet protocol, so no nodes collide.

* 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>**

**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**

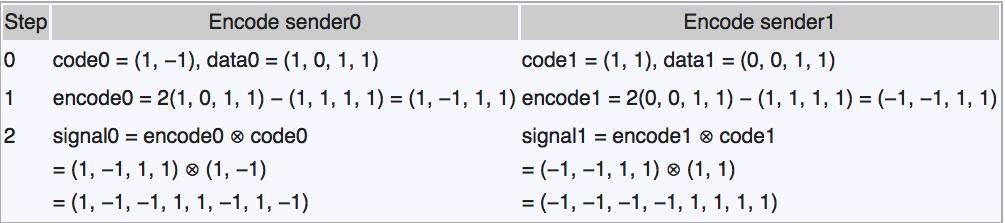
**CDMA** is a channel access method, using a unique code / chipping sequence assigned to each user.

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

CDMA Encoding / Decoding:

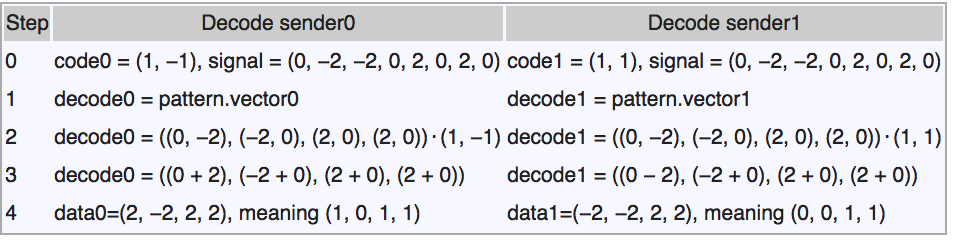
* Assume signal #1 = (1, 0, 1, 1) | chipping sequence cM = (1, -1)  
   signal #2 = (0, 0, 1, 1) | chipping sequence cM = (1, 1)
* **Encoded Signal = (original data) modulated by (chipping sequence)**

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:

  
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.  


* **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. Probing: sending out a probe request on multiple channels that specifies an SSID (searching for a network like in WiFi)
2. Authentication: WEP/WPA/WPA2 , public/shared key authentication
3. Association: 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**

* Rate Adaptation: Dynamically change transmission rate to adapt to current wireless channel conditions.
* Power Management: 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 + Ad-Hoc (no need for infrastructure)
* Replacement for cables (mouse, keyboard etc.).
* Master/Slave: Slaves request permission to send to master.

**Section 4: Network Security**

**Symmetric Encryption**

**XOR** allows easy encryption / decryption. Easily switch between plaintext <--> ciphertext.

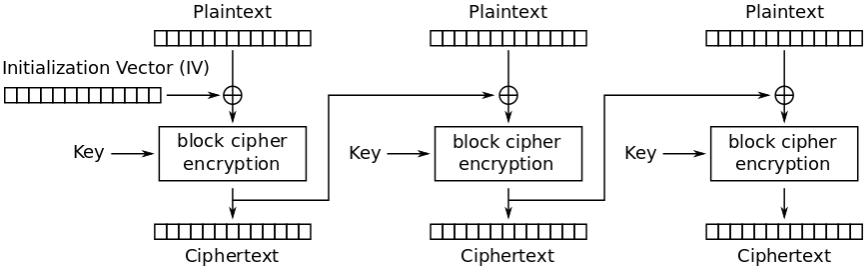
**Stream Cipher**: Example = One-Time-Pad

**Block Cipher – DES**: 56-bit key, 64-bit plaintext input. Uses CBC. Make it more secure with 3DES: encrypt 3 times w/ 3 diff keys.

* METHOD: (1) Initial Permutation (2) 16 rounds of encryption, each using 48-bits of key (3) Final Permutation

**Block Cipher – AES**: 128/192/256-bit keys, 128-bit input. Uses CBC.

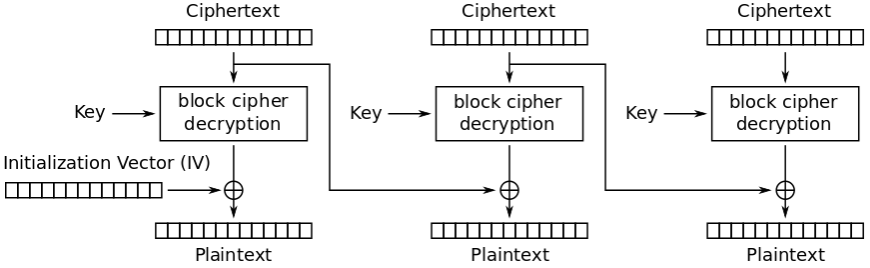
**Cipher Block Chaining (CBC)**

****

Solves the identical output problem

1. Initialisation Vector **XOR** Plaintext1  
2. Encrypt block1 = Ciphertext1  
3. Ciphertext1 **XOR** Plaintext2

4. Encrypt block2 = Ciphertext2  
5. Repeat #3 onwards for each block

****

Parallel Decryption

1. Apply the key to each ciphertext block.

2. For the first block:  
InitVector **XOR** FirstCipher = Plaintext  
3. For all else:  
CurrCipher **XOR** PrevCipher = Plaintext

**Asymmetric / Public Encryption**

**Asymmetric / Public Key Cryptography** is any crypto-system that generates a pair of keys:

1. **Public Key**: given out to the public and used to ENCRYPT a message.
2. **Private Key**: known only to the owner and used to DECRYPT & READ the message.

Everyone keeps their Private Key / Reading Key secret, but they all share a Public Key / Writing Key.

**Vulnerability of Public Encryption**: **Man-In-The-Middle**

* A spy can intercept the Shared Secret between Alice and Bob + send both parties their own Fake Shared Secret
* Any message now received can be decrypted by using the Fake Shared Secret.
* The main problem is that Diffie-Hellman doesn’t AUTHENTICATE the participants of the exchange.

**RSA** is a public-key cryptographic algorithm.

It is based off the **Factoring Problem** = difficulty of finding the factors of the product of two prime numbers.

1. Choose two distinct prime numbers **p = 3** and **q = 11**
2. Compute **n = pq** (n will be used as the mod for both private and public keys)  
   **n = 33 --> the mod**
3. Compute **phi = (p – 1) (q – 1)**  
   **phi = 2 \* 10 = 20**
4. Choose an **e** such that **1 < e < phi** ande / phiare coprime (gcd(e,phi) = 1) **e = 7**
5. Compute the secret exponent **d** such that **(d \* e) % phi(n) = 1** (as 7\*3 = 21 % 20 = 1) **d = 3**

**Public Key is (e, n) = (7, 33) Private Key is (d, n) = (3, 33)**

Suppose there is a plaintext m = 2.

* **USING PUBLIC KEY (e, n):** Encrypting plaintext m 🡪 cipher-text c  
  **c = 27 % 33 = 29**
* **USING PRIVATE KEY (d, n):** Decrypting cipher-text c 🡪 plaintext m **m = 293 % 33 = 2**

**Authentication**

**Nonces** are also non-secret, random value generated to be used only ONCE and protects against Replay attacks.

* Ensures that old communications can’t be re-used. Used in authentication protocols

**Session ID / Session Tokens** is a piece of data used to identify a session (series of message exchanges). Prevents Replay attacks.

* They expire after a short period of pre-set time of inactivity or become invalid after a goal is met.
* Usually in the form of a hash generated by a hash function sent from a server to a client to identify a current session.

**Digital Signatures**: digital signing of a document

* Verifiable, non-forgeable, non-repudiation (you can’t claim something didn’t happen when it did happen)

**Hash Functions**

**Hashing**: A function that takes a variable sized string of data and maps to a unique fixed size string (hash-value).

* Mostly for integrity + authentication

**MD5**: Computes 128-bit hashes (broken)

**SHA1**: Computes 160-bit hashes (broken)

**SHA2 – SHA3**: Best for security.

**Public Key Certification / Public Key Infrastructure (PKI)**

**PKI** is a suite of procedures for managing and revoking **Digital Certificates** (AUTHENTICATION) and managing Public Key Encryption. Used primarily to secure electronic transfer of information.

**Management of Digital Certificates: CERTIFICATE AUTHORITIES (CA)**

* They hold a “bank” of Public Keys
* They verify the identity of an individual through extensive vetting.
* If satisfied, they provide a **Signed Certificate** confirming the identity of the individual, encrypted with a private key held by the CA. So people communicating with this individual can use the CA’s Public Key to decrypt the certificate + authenticate the individual.
* Problems: Horton’s Principle + Single Point of Failure

Retrieving Bob’s public key from CA:

* Apply CA’s public key to Bob’s Cert = Bob’s public key