**Common Conversions**

**Time:** 1 second = 1,000 milliseconds

**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**

**Transport Layer:** Responsible for delivering data to applications on host computers.

**Multiplexing / De-multiplexing**

**Multiplexing**: Multiple data streams from different sources are combined and transmitted over a single shared medium.

* Handles data from multiple sockets, adding transport header which is used in de-multiplexing.

**De-multiplexing**: At the receiving end, the reverse occurs, separating data streams from the single channel and routing to corresponding receivers / destination.

* Use the transport header info to deliver received segments to the correct socket.

**Connectionless De-multiplexing (UDP)**

* IP datagrams with the same dest port but different source IP/port will be directed to the same socket at the dest.

**Connection-orientated De-multiplexing (TCP)**

* Receiver uses all four values of TCP 4-tuple to direct segments to the appropriate socket.
* A server host may support simultaneous TCP sockets.
* Web servers have different sockets for each client. Non-persistent HTTP will have a different socket for each request.

**UDP: User Datagram Protocol**

UDP is a communications protocol used primarily for establishing **low-latency** and **loss-tolerating** connections between apps.

It is **connection-less**, where each UDP segment is handled independently of others.

**UDP Header (8 bytes)**: SRC PORT # | DEST PORT # | LEN (BYTES) | CHECKSUM | PAYLOAD

**UDP Checksum**

* Treat segment content + header fields as a sequence of 16-bit integers.
* Checksum = binary addition of segment contents (sum) 🡪 invert the bits to get checksum (complement of sum)
* Sender puts checksum value into UDP checksum field.
* Receiver adds all segment content with the checksum. Result should = 1111 1111 1111 1111, else there are errors.

**Reliable Data Transfer (RDT) – STOP AND WAIT PROTOCOLS**

With RDT, transferred data is **NOT CORRUPTED | NOT LOST | DELIVERED IN ORDER**. TCP offers this service model to apps.

**RDT 1.0 – Transfer over a perfectly reliable channel (not a realistic model)**

* All packet flow is from sender 🡪 receiver, no need for receiver-side to provide feedback to sender.
* Assume the receiver is able to get data as fast as the sending of data, thus no need for flow/congestion control.

**RDT 2.0 – Transfer over a channel with bit errors (more realistic model)**

* In this model, we assume packets can be corrupted.
* Recover from errors through **ARQ: Automatic Repeat Requests Protocols**.

ARQ - Stop-and-Wait Protocol: Sender sends packet and waits for an ACK or NACK. ACK = not corrupted | NACK = corrupted.

Sender can’t receive more data from upper layer while waiting.

Flaw with Stop-and-Wait: ACK/NACK can be corrupted themselves.

Solution to flaw: Number packets with a sequence number #0 or #1

**RDT 2.1 – Protocol includes sequence numbers #0 #1 to track expected packets**

* Sender: Check ACK/NACK + Remember whether expected packet = seq #0 or #1
* Receiver: See if packet is a duplicate (duplicate if expected seq# != received seq#).

**RDT 2.2 – NAK-free protocol**

* Same as 2.1 but only using ACKs. Instead of sending NACK, send the same ACK for the last successfully received packet.
* E.g. Server PKT\_0 🡪 Client | Client ACK\_0 🡪 Server | Server PKT\_1 🡪 Client | **Client ACK\_0 🡪 Server [ NACK ]**

**RDT 3.0 – Transfer over a channel with bit errors and loss**

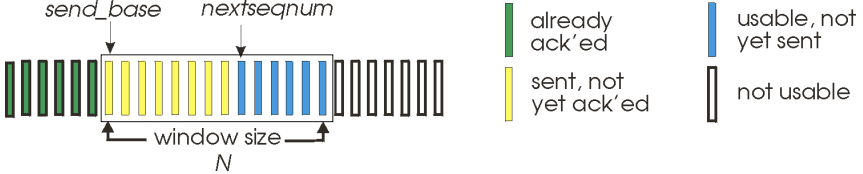
* New assumption: In addition to bit errors, the channel can also lose entire packets.
* New concerns: (1) How to detect packet loss? (2) What to do when packet loss occurs?
* **Time-Based Packet Retransmissions** that can interrupt the sender after an amount of time waiting for an ACK expires.  
  The sender will need to:  
  (1) Start the timer after each packet  
  (2) Respond to timer interrupt – take appropriate actions i.e. retransmit packet  
  (3) Stop the timer
* Retransmission is an all-in-one solution: doesn’t matter if packet is LOST or LARGE DELAY. Seq #s will handle duplicates.

**Reliable Data Transfer (RDT) – PIPELINED PROTOCOLS: Go-Back-N, Selective Repeat**

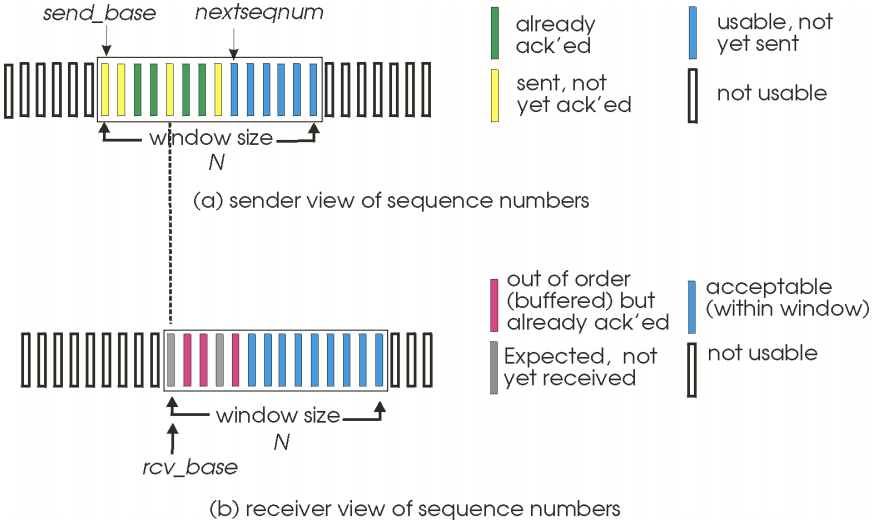
Pipelined protocols allow for multiple “in-flight”, un-acknowledged packets and increases utilisation.

|  |  |
| --- | --- |
| **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 |

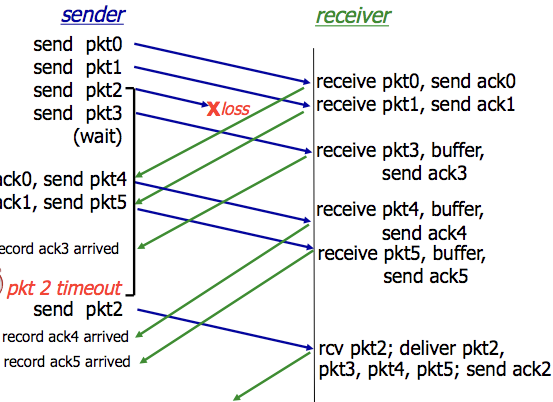
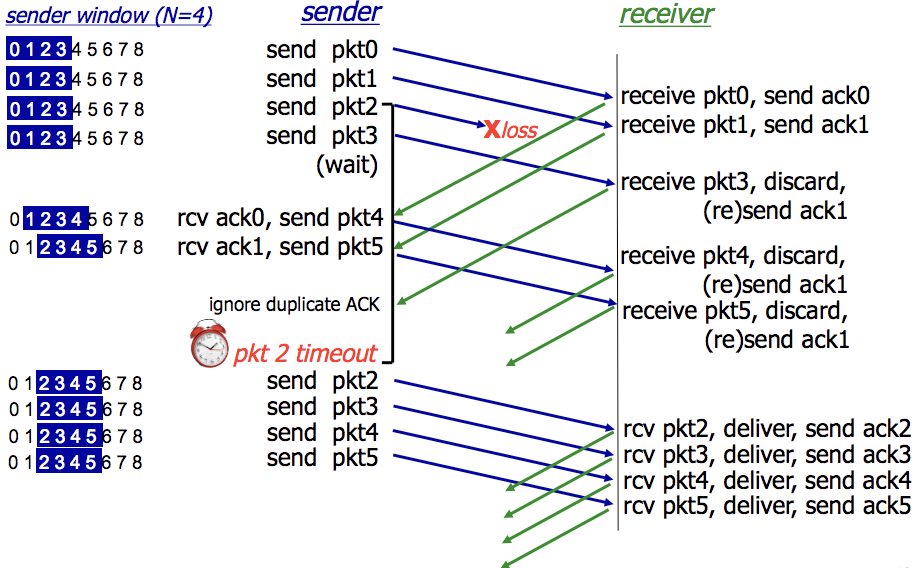
**GBN Sender Window:**



**Select Repeat Sender / Receiver Windows:**



**Go-Back-N Selective Repeat**



**Transmission Control Protocol (TCP) – Segment structure**

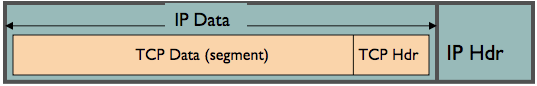
**TCP Header (20 bytes)**: UDP fields + seq#, ack#, receiver window #bytes, connection establishment + teardown, options.

**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 – Process, Timer + Retransmissions**

**TCP Sender / Receiver Process**:

* Sender: Sends packet of SEQ# = X. Packet len = B bytes [ X , X+1 , X+2 . . . X + B–1 ]
* Receiver: If data prior to X has already been received, send ACK# = X+B  
  X+B = next expected seq # from the next packet.

If highest-order byte received is Y, where Y+1 < X 🡪 resend ACK Y+1

* Next Seq# = ACK#

What else can TCP do?

* Receivers can buffer out-of-sequence packets like Selective Repeat / NOT drop out-of-seq packets.
* Senders can maintain a single retransmission timer like Go-Back-N and retransmit on timeout.

Set up TCP timeout by choosing a value > RTT.

* Choose value too short: premature timeout, unnecessary retransmission
* Choose value too long: slow reaction to segment loss and lower throughput for connection

**1. Measure EstimatedRTT:**

Exponential Weighted Moving Average

EstimatedRTTCURR = (1 – a) \* EstimatedRTTPREV + a \* SampleRTTRECENT

* SampleRTT
  + Time measured from segment transmission until ACK receipt (ignoring retransmissions)
  + Current value of RTT
* Typical value of a = 0.125

**2. Measure timeout interval: EstimatedRTT + “Safety Margin”**

RTT Deviation is calculated by

DevRTT = (1 – b) \* DevRTT + B \* |SampleRTT – EstimatedRTT|

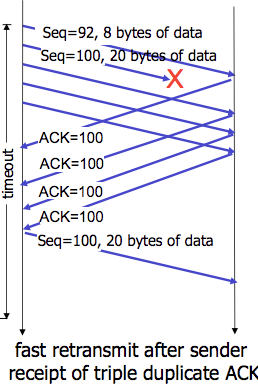
* Typical value of b = 0.25

Timeout Value is calculated by

Timeout Interval = EstimatedRTT + 4 \* DevRTT

* 4 \* DevRTT = The “Safety Margin”

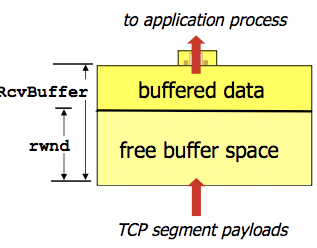
**TCP – Fast Retransmission**



TCP has a Fast Retransmissions feature that uses duplicate ACKs to trigger early retransmission.

* If sender receives 3 duplicate ACKs for the same data, resend the un-ACK’d data with the smallest sequence #.
* Timeout periods are often long, so there is a long delay before resending lost packets. No need to wait for timeout.

**TCP – Flow Control**



**TCP Flow Control** is where the receiver controls the sender, so the sender won’t overflow the receiver’s buffer by transmitting too much, too fast.

**Receiver Advertised Window (RWND)**: Advertises available recv buffer space in the RWND value in TCP header.

Sender limits the amount of un-ACK’d data to receiver’s RWND value.

**TCP – Connection Management**

**C Establishment**: (1) SYN 🡪 (2) SYN-ACK 🡪 (3) ACK + DATA 🡪 Data exchange

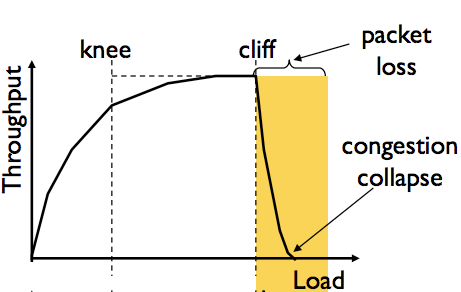
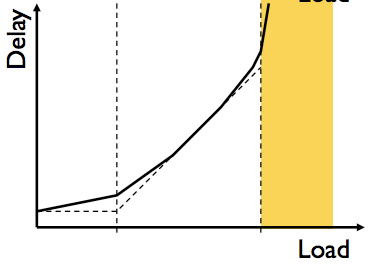
**C Teardown**: Data exchange 🡪 (1) FIN 🡪 (2) ACK-FIN 🡪 (3) ACK 🡪 (4) WAIT / Retransmit ACK 🡪 (4) CLOSE CONNECTION

**RST:** Reset Flag. Possibly because application has crashed on one end, or socket is closed, or there is a firewall.

**TCP – Congestion Control**

**Congestion Control** is needed if a network node/link/router is taking in more data than it can output, leading to collapse.

**Congestion Collapse**: Throughput starts to drop to zero, delays approach infinity.



**Knee Point**

Throughput increases slowly

Delay increases really fast

**Cliff Point**

Throughput begins to drop to zero  
(Congestion collapse)

Delay approaches infinity

**CWND**: Congestion Window i.e. how many bytes can be sent without overflowing routers?

* Sender varies the window size to control the sending rate.

**TCP sending rate =~ CWND / RTT** bytes per second.

**Sender-Side Window**: minimum { RWND , CWND }

Basics of sender rate adjustment:

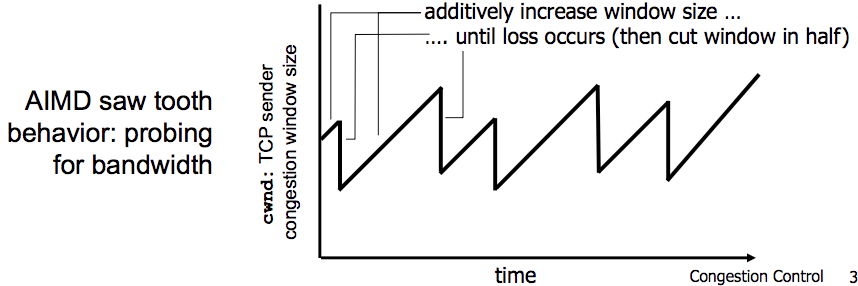
* Upon receiving ACK 🡪 increase rate
* Upon detection of loss 🡪 decrease rate

**(1) Bandwidth Discovery with Slow Start (SS)**

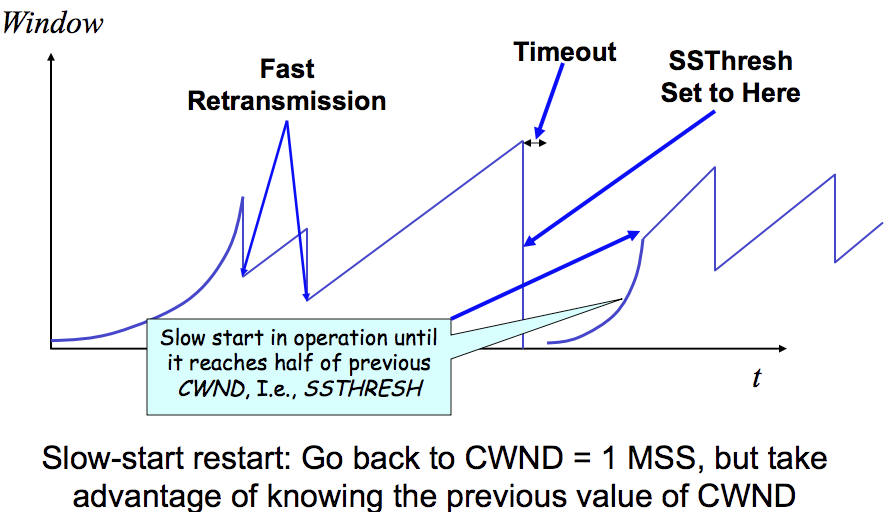
* When connection begins, initial rate is slow (for safety) then increase exponentially until the first packet loss event.
* Initial CWND = 1 MSS 🡪 Double CWND every RTT or alternate Increment CWND for every ACK received

**(2) Additive Increase Multiplicative Decrease** **(AIMD)**

* Slow start gave an estimate on available bandwidth. Now we want to track variations of this bandwidth via. probing.
* Additive Increase: Sender increase CWND / transmission rate, probing until a loss event occurs
* Multiplicative Decrease: Cut CWND in half after a loss occurs



**Slow-Start Threshold (SSThresh)** is used to determine when a sender should stop slow-start and start AIMD.

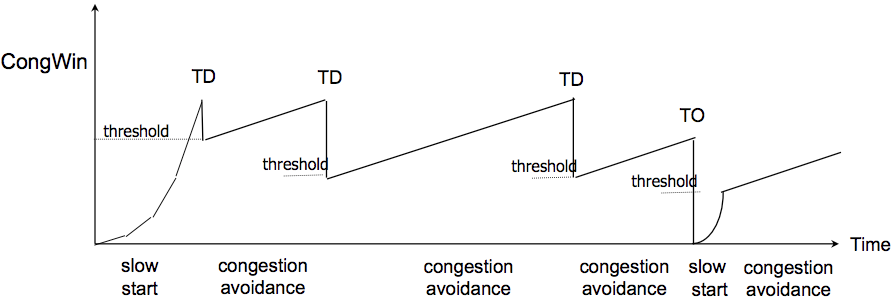
* SSThresh is initialised to a large value. On timeout, SSThresh = CWND/2.
* If CWND < SSThresh = Slow-Start
* If CWND > SSThresh = AIMD / Congestion Avoidance

Congestion Control Rate Increases:

* Slow-Start: CWND += MSS
* Congestion Avoidance/AIMD: CWND += MSS/CWND

Congestion Control Rate Decreases:

* DupACKs:  
  SSThresh = CWND/2  
  CWND = CWND/2
* Timeout/Loss Event:   
  SSThresh = CWND/2  
  CWND = 1 MSS

**TCP - Flavours**

**TCP Tahoe**: CWND = 1 on DupACK and Timeout

**TCP Reno**: Same as above.

**TCP New-Reno**: TCP Reno + improved fast recovery

TD = Triple Duplicate ACKs

TO = Timeout

**Section 2: Network Layer**

**The Network Layer** focuses on packet forwarding through intermediate routers / networks from source host to destination.

* Service Model: Guaranteed Delivery, Guaranteed Minimum Bandwidth, In-Order Deliveries

**Forwarding:** Move packets from router’s input to appropriate router output. (i.e. station platforms)

* Forwarding Table determines which output link to forward the packet to. **Entry =** **{ K=Header Value | V=Output Link }**
* Generalised Forwarding: Forward packets based off any set of header-field value, using **Longest Prefix Matching**.

For a given Destination IP Address, use the longest IP prefix that matches the address.  
-> STEP 1: Find the IP ranges / entries in forwarding table which match with the Dest. Address IP.  
-> STEP 2: Choose the IP range / entry with the longest matching prefix address.

**Routing**: Determine the route taken by packets from source to destination. (i.e. start 🡪 many stations 🡪 destination)

The **Data Plane** refers to the functions that determine how packets are forwarded from a router input to its output port.

The **Control Plane** refers to the functions that determine how a packet is routed among routers in the end-to-end path.

* Per-Router Control Plane: Individual routing algos in each router interact in the control plane.
* Logically Centralised Control Plane / Software-Defined Networking (SDN): A distinct controller interacts with local Control Agents (CA’s) / centralised servers.

**IP – Internet Protocol**

**IP Packet Structure:** 20 bytes of Standard Header, then Options

**Version Number** (4 bits)

* Indicates version of the IP protocol
* **“4” = IPv4** **| “6” = IPv6**

**Header Length** (4 bits)

* Number of 32-bit words in the header
* Typically “5” for **20-byte** IPv4 header
* Can be more with IP options

**Total Datagram Length** (16 bits)

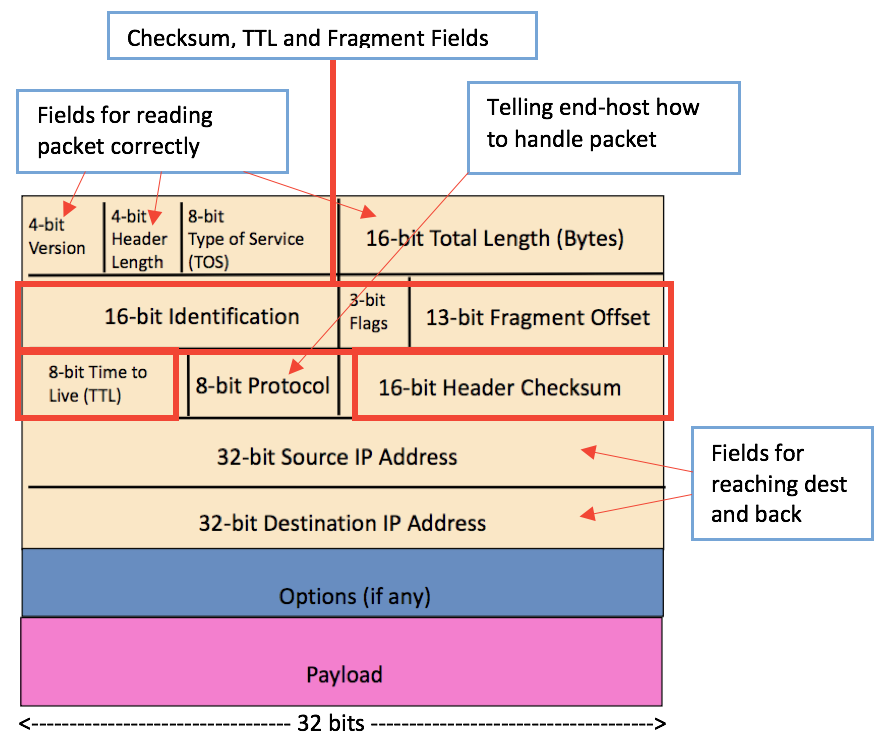
* # bytes in the packet
* **Max size = 65,535 bytes (216 – 1)**

**Protocol** (8 bits)

* Identifies the upper-layer protocols
* Important for de-multiplexing at receiving host

E.g. **Protocol=6 🡪 TCP** | **Protocol=17 🡪 UDP**

**Payload** (variable length)  
Typically a TCP or UDP segment.



**Time To Live (TTL)** (8 bits): Max number of remaining hops.

* Value decremented for each hop 🡪 packet discarded if value = 0 and a “time exceed” message is sent to the source.
* This mechanism will prevent packet forwarding loops.

**IP Fragmentation Reassembly** (Frag offset – 13 bits)

* A large IP datagram is divided / fragmented within the network, as datagram can’t exceed the **Max Transmission Unit**.
  + **MTU** = size of the largest network layer data that can be sent in a single network transaction.
* One datagram becomes several, which are reassembled at the final destination.
* Fragmentation Header Bits are used to identify the ORDER of the fragments.

*The offset is the address or the locator from where the data starts in the original payload. The system/router takes the payload and divides it into smaller parts, keeping track of this offset so that reassembly can be done later.*

20 byte header in each packet.

Original packet (4000 bytes)

= 3980 payload + 20 header

Frag pkt #1 (1500 bytes) | Offset = 0 (start of OG data)

= 1480 payload + 20 header

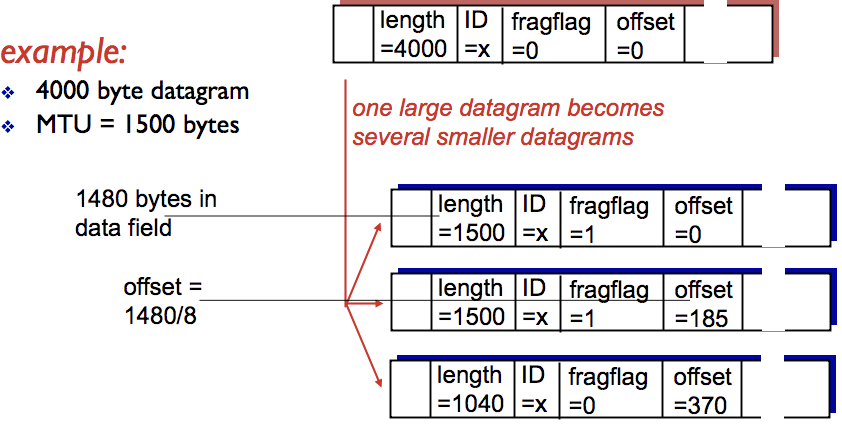
Frag pkt #2 (1500 bytes) | Offset = 1480 / 8 = 185

= 1480 payload + 20 header

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

= 1020 payload + 20 header

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



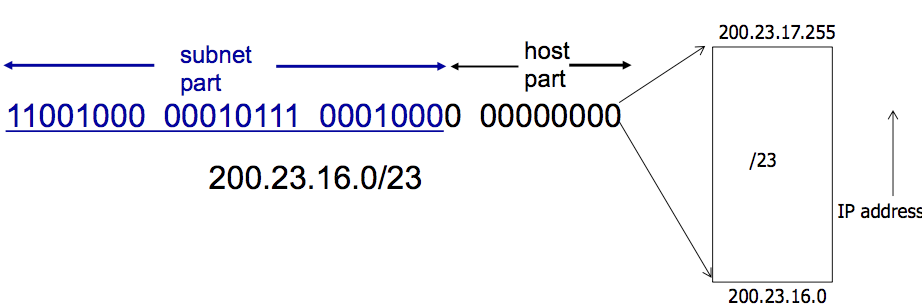
**IPv4 Addressing**

**Interface** is a connection between a host/router at the link-layer, typically with multiple interfaces per host/router.

**IP Addresses** are associated with each interface.

**Subnets** are groups of IP addresses that form multiple smaller divisions of a larger/major network. E.g. Companies use subnets.

**Classless Inter-Domain Routing (CIDR) – Today’s Internet Addressing**



Address format: **[ 255 . 255 . 255 . 255 / X ]**

where X = #bits in subnet portion of address.

**CIDR** uses hierarchical address allocation: addresses are allocated in continuous chunks / prefixes.

What happens when an organisation wants to switch from ISP#1 🡪 ISP#2?

* The organisation keeps its IP address block, ISP#1 and ISP#2 will continue to advertise their address blocks.
* ISP#2 will also advertise the org’s more specific address block.
* Routers from the internet will know which ISP to route packets towards, using **Longest-Prefix-Matching.**

A **subnet mask** separates the IP address into the NETWORK PART and HOST PART of the address. Example:

* IP: 200.23.16.2/24 | Subnet Mask: 11111111 11111111 11111111 00000000 (24 most-sig bits set to 1) & IP
* Network: 200.23.16.0 | Host: 0.0.0.2 (remainder bits after MASK & IP)

How many IP addresses belong to the subnet 128.119.254.0/25?

* IP = 10000000 01110111 11111110 00000000
* Subnet Mask = 11111111 11111111 11111111 10000000
* Host has **0b1111111 (127) addresses to use**. Therefore RANGE = 128.119.254.0 to 128.119.254.127

**Dynamic Host Configuration Protocol (DHCP)**

DHCP allows a host to dynamically obtain its IP from a network server when it joins the network.

* Host can receive same IP address each time it connects to the network or be assigned a temporary IP addr each time.
* Host can renew its lease on address | Allows reuse of an address (only holds addr when “connected” to network)

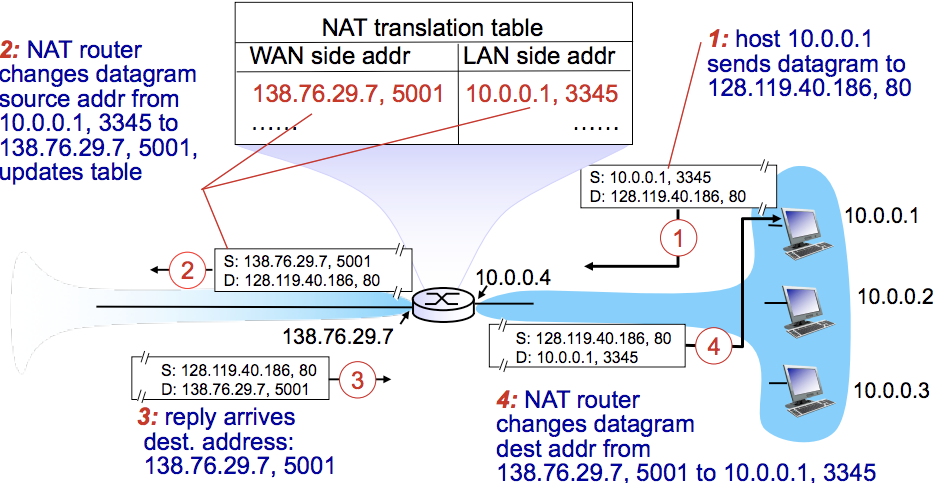
DHCP Steps:

1. Host broadcasts DHCP Discover message
2. DHCP server responds with DHCP Offer message
3. Host requests IP address with DHCP request message  
   🡪 Client requires: **[ IP ] [ Address of 1st hop router ] [ Address of DNS server ]  
   🡪** DHCP request encapsulated in UDP 🡪 encapsulated in IP 🡪 encapsulated in Ethernet frame  
   🡪 Broadcast Ethernet frame on LAN  
   🡪 Frame is received at the router running the DHCP server  
   🡪 Ethernet frame de-multiplex to IP 🡪 de-muxed to UDP 🡪 de-muxed to DHCP request.
4. DHCP server sends the address with DHCP ACK message  
   🡪 DHCP server formulates an ACK: **[ client IP ] [ Address of 1st hop router ] [ Address + Name of DNS server ]  
   🡪** Encapsulate ACK in frames and send to client. Client demuxes back to DHCP  
   🡪 Client will now know its **IP Address, Address of 1st hop router, Address + Name of DNS server**.

More about DHCP:

* The **MAC** address is used to identify clients. DHCP server can be configured to accept a list of specified MAC addresses.
* DHCP loopholes: DoS attack by exhausting pool of IP addresses in LAN | Masquerade as a 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 of NAT:

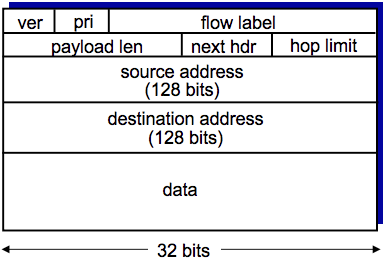
* A range of addresses are not needed from an ISP. Use a single IP address to represent all devices in the LAN.
* Can change addresses in the LAN without having to notify the outside world
* Can change ISP without changing addresses of devices in the LAN.

Disadvantages / Issues of NAT:

* Controversial as routers should only process up to the Network Layer, thus violating end-end agreement.
* NAT modifies port # and IP address 🡪 **requires recalculation of TCP and IP checksum**
* Some apps embed IP / port # in their message protocols. Some encrypt the IP / port# fields. How to decrypt on NAT?!
* NAT traversal problems:

**IPv6 Addressing**

**Initial Motivation**: 32-bit address space will run out soon.

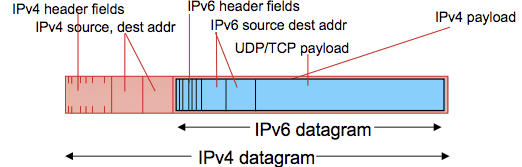
**Additional Motivation**: IPv6 header format helps speed up packet processing / forwarding.

IPv6 Datagram Format:

* Fixed-len 40-byte header, no fragmentation allowed
* **Priority**: identify priority among datagrams in flow (traffic class)
* **Flow Label**: identify datagrams in the same flow
* **Next Header**: Identify the upper layer protocol for data

Changes from IPv4

* **Checksum** removed entirely to reduce processing time at each hop
* **Options** allowed, but outside of the header, indicated by “next header” field
* **ICMPv6**: new version of ICMP



**Tunnelling**: IPv6 datagram is carried as a payload in IPv4 datagram among IPv4 routers.

🡨 See diagram

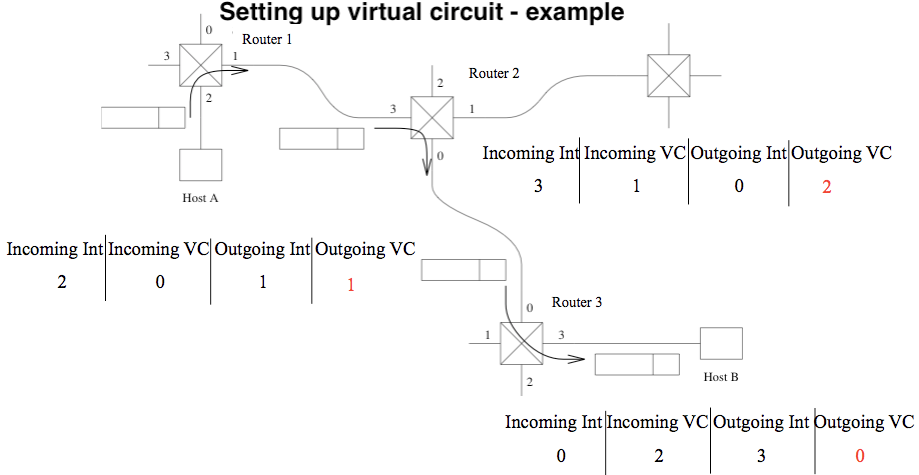
**Virtual Circuit Network**

**Datagram Network**: Provides a connection-less network layer service.

* NO SETUP | NO KEEPING STATE ABOUT CONNECTION | PKTS FORWARDED USING DEST HOST ADDRESS ONLY.

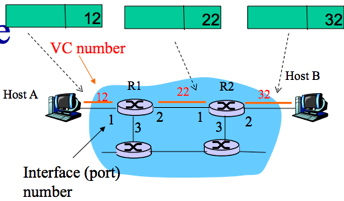
**Virtual Circuit Network**: Provides a connection-based network layer service.

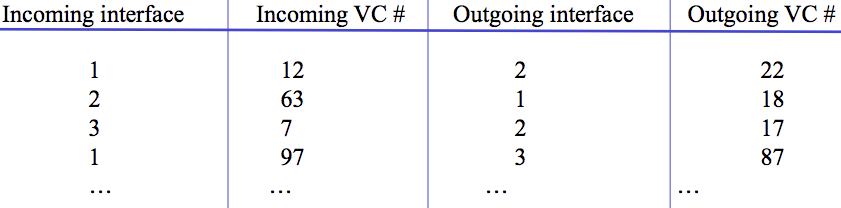
* Signalling protocols used to setup and maintain connection of virtual circuits. Not used in today’s internet.



**VC SETUP**:

1. Source  
   - Sends setup msg with dest addr.
2. Intermediate routers  
   - Choose VC # (from lowest = 0)  
   - Determine outgoing interface from routing table.  
   - Create entry in VC table.  
   - Forward setup to next hop.
3. Setup reaches dest:  
   - Dest chooses incoming VC #  
   - Chosen incoming VC # = outgoing VC # of all routers except last.  
   - Send ACK along the reverse path to SOURCE
4. Acknowledgement  
   - Intermediate routers complete their VC tables



**Forwarding Table in a Router:** switches/routers maintain connection state info

**FORWARDING =** Within Router (input 🡪 output link) **ROUTING** **=** Make sure hop leads to destination.

**Routing Protocols: Intra-Domain Routing (Link State / Distance Vector) and Inter-Domain Routing**

A routing protocol determines the end-to-end path of packets through the network.

The forwarding table determines the local forwarding at this router.

**Autonomous Systems (AS)** or **Domains** is a region of a network under a single admin authority. E.g. an ISP is an AS.

Internet routing works as two levels:

1. **Intra-Domain Routing Protocol:** AS Establishing routes within its own AS / domain.
   1. Single admin, so no policy decisions are needed. Performance > Policy.
   2. Examples of intra-domain routing:  
      Link State 🡪 **Open Shortest Path First (OSPF)**Distance Vector 🡪 **Routing Information Protocol (RIP)**
2. **Inter-Domain Routing Protocol**: AS Establishing routes between other AS / domains
   1. Admin wants control over routing in network + who routes through its network. Policy may > Performance.
   2. Examples of inter-domain routing:  
      Path Vector 🡪 **Border Gateway Protocol (BGP)**

**Link State Routing (Global)**: All routers have the complete topology and maintain / know the cost of each link in the network.

* How it works: (1) **Link State Advertisement (LSA) Flooding** (2) **Path calculation with Djikstra’s**
  + When receiving a new Link State msg, the router forwards it to all neighbours except one that sent the msg.
  + Routers keep a local copy so they don’t forward previously seen LSA’s.
  + Eventually, each node learns the entire network topology + can use Djikstra’s to compute shortest path.
* Eventually, each node learns entire network.
* Characteristics
  + Connectivity / cost changes are flooded to all routers in the network.
  + Converges quickly (less consistency, looping)
  + Limited network sizes, otherwise it will be too costly.
* Challenges:
  + Packet Loss / Out-of-order packets (solved with ACKs, Retransmissions, Seq Numbers, TTL for packets)
  + Scalability: # Messages to flood **O(N\*E)** where N = #nodes E = #edges | Djikstra’s **O(N2)**  
    # entries in topology database **O(E)** | # entries in forwarding table **O(N)**
  + Transient Disruptions / Infinite Loop problems: Inconsistent link-state database, as some routers know about failures before others. Shortest path is not always consistent, which can cause transient / infinite loops.
  + Oscillations: Costs can change around continuously. For given new costs 🡪 new route 🡪 new costs and so on.

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

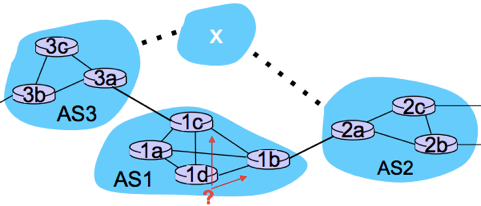
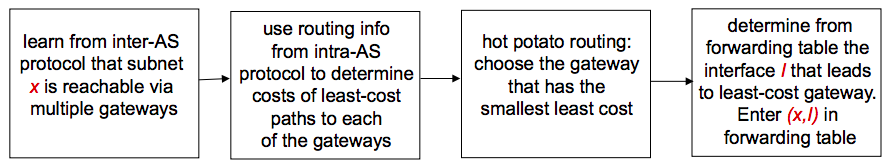
* How it works:  
  (1) Each router initialises its DV table based on link costs to immediate neighbours + sends its DV to the neighbours.  
  (2) Neighbours process the DV and repeats STEP #1 until the iterative process converges to a set of shortest paths.  
  (3) Each node then waits for changes in their local link cost or msg from neighbours.  
  (4) If change occurs 🡪 recompute costs in DV and notify neighbours if anything changes.
* Initial state: best 1-hop paths | one simultaneous round = best 2-hop | k simultaneous rounds = best (K+1)-hop paths
* Characteristics:
  + Cost changes are iterative, exchanges info from neighbour to neighbour.
  + Requires multiple rounds to converge
  + Scales to large networks.
* **Counting to Infinity Problem (“bad news travels slowly”)**: Usually occurs when a node becomes broken.
  + Because of a broken link, nodes keep incorrectly updating their DV table and increasing cost for the broken link until the updates slowly propagates through the network and eventually reaches infinity.
* **Poisoned Reverse Rule** is a method to avoid the Count to Infinity Problem.
  + Routers actively advertise certain links as unreachable (cost=infinity). However, this will significantly increase the number of routing announcements made in the network.

Comparison of Link State vs. Distance Vector

|  |  |  |
| --- | --- | --- |
|  | **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. |

**Inter-Domain Routing Protocol**

* Gateway Routers are the “edge” of an AS which links to another AS’s gateway in the internet.
* How forwarding works between different AS networks:  
  SCENARIO: Router in AS1 needs to determine which gateway to forward a packet to, so it can reach subnet X.



**Section 3: Link Layer**

The **Data-Link-Layer** has the responsibility of transferring a datagram from one node to a physically adjacent node over a link.

**Nodes:** Hosts and routers

**Links**: Communication channels connecting adjacent nodes i.e. Wireless, Wired, LAN

**Layer-2 Packet**: Frame encapsulating a packet/datagram

**Link Layer Services**

**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.

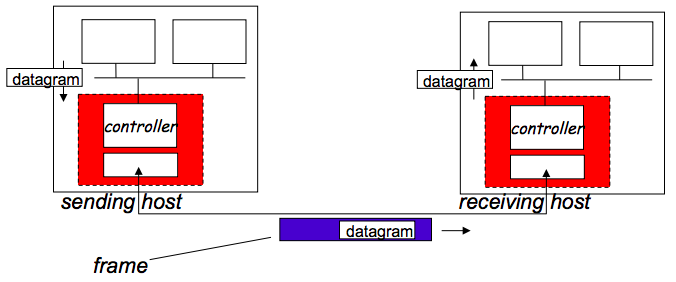
**Error Correction**: Receiver identifies and corrects bit-errors without needing retransmission.

**Half-Duplex and Full-Duplex**: DUPLEX = ability for two devices to communicate at the same time.

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

Where is the link layer implemented?

* In a **Network Interface Card (NIC)** embedded on each and every host. E.g. Ethernet card, 802.11 card.
* NIC is a combination of hardware, software, firmware.



**Adaptors Communicating:**

Sending side:

* Encapsulates datagram in frame
* Adds error checking bits, rdt, flow control etc.

Receiving side:

* Looks for errors, rdt, flow control
* Extracts datagram, passes to upper layer at receiving side.

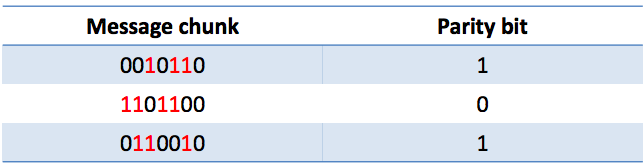
**Link Layer Error Detection: Parity Check Method**

An error detection method is using **Parity Check Method**. In practise, bit errors occur in bursts.

With Parity Checking, we are willing to trade computational complexity for space efficiency.

* We make detection checking more complex, but without the need to input lots of extra data for checking.

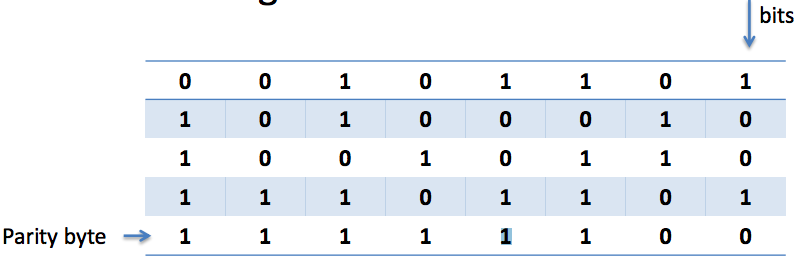
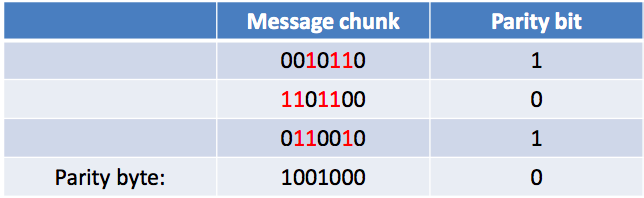
**Simple Parity – Sender**: For every d\_bits add a parity bit. Example: d = 7 | Result: 0010110**1**1101100**0**0110010**1**

* Parity Bit = 1 for an odd number of one’s.
* Parity Bit = 0 for an even number of one’s.

**Simple Parity – Receiver**: For each block of size d\_bits,  
calculate the parity bit and compare with the sent data.

**Simple Parity Cost**: 1 extra bit for each d\_bits.

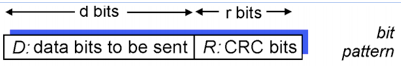
**Two Dimensional Parity:** Compute parity on columns as well as rows.



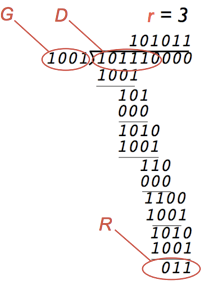
* Exactly ONE-BIT has been flipped in the example: which one is it?

**Link Layer Error Detection: Cyclic Redundancy Check (CRC) method**

CRC is another error-detection method, widely used in Ethernet, 802.11, WiFi, ATMs.



**Goal**: choose **R** number of CRC bits && choose G a generator  
such that:

1. **<D,R>** is exactly divisible by **G**
2. Receiver also knows the value of **G** and divides  
   <D,R> by G to check for errors.
3. If remainder = 0 🡪 NO ERRORS  
   If remainder != 0 🡪 ERROR DETECTED

**Multiple Access Links / Multiple Access Protocol**

There are two types of links:  
**(1)** **Point-to-Point**: Single wire e.g. Link for dial-up access | Link between Ethernet switch/host

**(2) Broadcast (shared wire or medium)**: e.g. 802.11 Wireless LAN

Broadcast links have a **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.
  + All frames in the collision are lost + broadcast channel is wasted during the collision interval.

**Multiple Access Protocol** is used to determine how nodes share a channel + determine when they can transmit.

* Communication about channel sharing must use the channel itself. i.e. no outside channels allowed for coordination.
* Used by both wired and wireless LAN and satellite networks.

An ideal Multiple Access Protocol: Given a broadcast rate of R bps

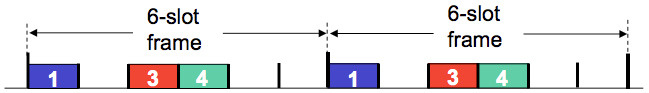
1. When one node wants to transmit, it can send at rate R
2. When N nodes want to transmit, it can send at rate R
3. Fully decentralised: no special node to coordinate transmissions
4. Simple

**MAC Protocols**

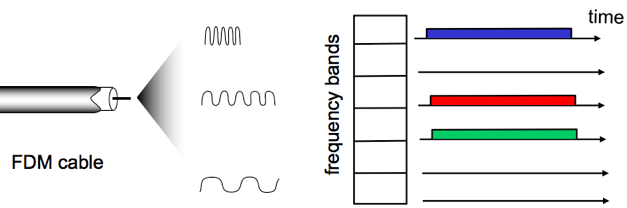
The **Medium Access Control** is the lower sub-layer of the data-link-layer, which provides addressing and channel access control mechanisms that make it possible for several nodes to communicate within a multiple access network over a shared medium.

Three classes:  
**(1) Channel Partitioning Protocols:** Divide channel into smaller “pieces” (time slots, frequency, codes)

**TDMA (Time Division Multiple Access)**

* Each station gets a fixed length slot (len = packet transmission time) in each round. Unused slots go idle.
* Example of TDMA: 6 station LAN, slots 1-3-4 have a packet, slots 2-5-6 are idle  
  

**FDMA (Frequency Division Multiple Access**

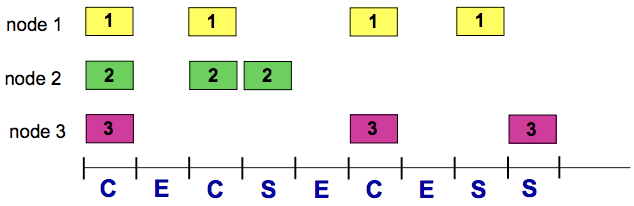
* The channel is divided into frequency bands, where each station has an assigned frequency.
* Unused transmission time in frequency goes idle.
* Example of FDMA: 6 station LAN, slots 1-3-4 have a packet, slots 2-5-6 are idle  
  

**(2) Random Access:** Channel not divided, allow collisions to occur and recover from collisions.

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

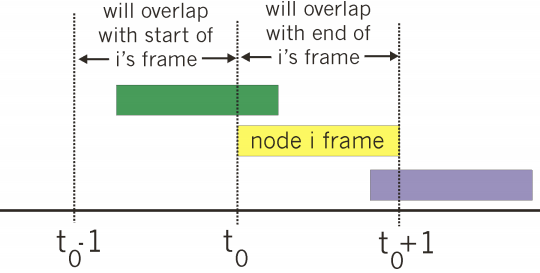
* NO COLLISION: The node can send the frame in the next slot

|  |  |
| --- | --- |
| PROS | CONS |
| * Single active node can continuously transmit at full rate of channel * Highly decentralised: only slots in nodes need to be in sync * Simple | * Collisions, wasting slots * Idle slots * Nodes maybe able to detect collisions in less than the time to transmit a packet * Clock synchronisation |

* COLLISION: The node retransmits the frame in each subsequent slot with P probability.

**Max Efficiency:** Channel is useful for transmissions **37%** of the time

**Pure Un-slotted ALOHA**: Simpler, with no synchronisation



* When the first frame arrives, transmit it immediately
* Collision probability will increase:  
  **frames sent at t0 collides with frames sent in [ t0 – 1** **, t0 + 1 ]**
* **Max Efficiency = 18%** of the time for useful transmissions.

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

* If channel is sensed to be IDLE: transmit entire frame.
* If channel is sensed to be BUSY: defer transmission.
* Collisions can still occur because propagation delay may cause two nodes to not hear each other’s transmissions.
  + Distance between the nodes + propagation delay affect collision probability.
* CSMA reduces but NOT eliminates collisions. This is a problem as collisions still take up an entire full slot.

**Carrier Sense Multiple Access + Collision Detection (CSMA / CD)**: Nodes detect collisions by sensing transmissions from other nodes while transmitting a frame.

* If collision is detected: node instantly terminates transmission of the frame + transmits a jam signal + waits for a random time interval before trying to resend the frame.
* Collisions are detected within a short-time frame
* Collisions are aborted, reducing channel waste
* CD is easy in wired LANs: measure signal strengths, compare transmitted, received signals
* Difficult in wireless LANs: received signal strength overwhelmed by local transmission strength

CSMA/CD Algorithm:

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 min frame size / max distance because transmission / propagation delay can affect collision probability.**

**(3) Taking Turns:** Nodes take turns, but nodes with more to send can take longer turns.

**Polling Protocol**

* The base station retains total control over the channel + Frame content is no longer fixed, allowing variable sized packets to be sent.
* The base station sends a specific packet (a poll packet) to trigger the transmission by a certain node.
* Nodes wait to receive a poll packet and upon receiving it, transmits the frame.
* How it works: A control token passed from one node to the next sequentially.
* Concerns: Token overhead, latency, single point of failure (token)

**MAC Protocols: Channel Partitioning vs. Random Access vs. Taking Turns**

* **Channel Partitioning Protocols (TDMA, FDMA)**
  + High load: Shares channels efficiently + fairly
  + Low load: Inefficient, 1/N bandwidth allocated even if only 1 active node in channel.
* **Random Access Protocols (Slotted ALOHA, Un-slotted ALOHA, CSMA, CSMA/CD)**
  + High load: High collision overhead
  + Low load: Single node can fully utilise channel
* **Taking Turns Protocols (Polling)**
  + Best of both Channel Partitioning + Random Access

**Local Area Network Addressing + ARP resolution protocol**

**MAC** is a 48-bit address burned in a NIC and is hex-based.

* It is used in a LAN to get a frame from one interface to another physically connected interface.
* Manufacturers of NICs buy a portion of MAC address spaces, administered by the IEEE.

**ARP (Address Resolution Protocol)** helps determine an interface’s MAC address and IP address.

* **ARP Table**: A table in each node in a LAN with entries of: **<IP Address, MAC address, TTL>**where TTL is time after which address is forgotten.
* Scenario #1: Send datagram from A 🡪 B within LAN (B’s MAC address is not in A’s ARP table)  
  1. A broadcasts ARP query packet containing B’s IP address 🡪 All nodes including B receives broadcast

2. B replies to A’s broadcast packet with B’s MAC address 🡪 Frame is sent to A’s MAC address

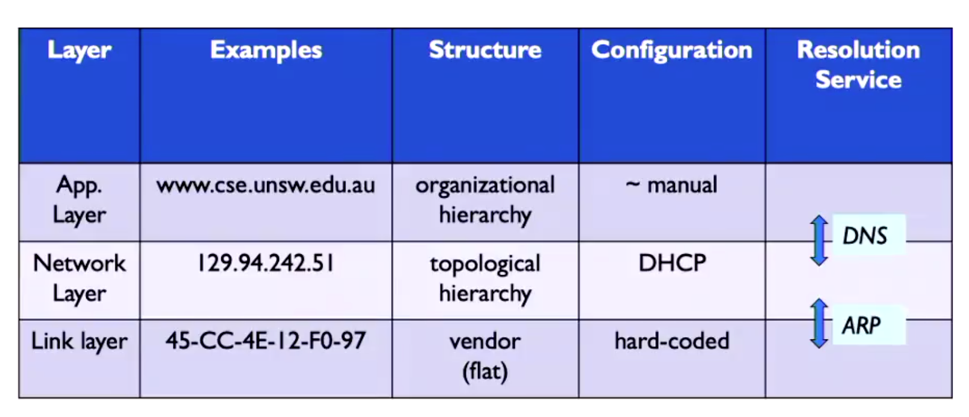
3. A caches/saves IP-to-MAC pair as an entry in its ARP table (until TTL is reached)

* Scenario #2: Send datagram from A🡪B via. R outside of LAN

1. For A to send a packet to B, A must know**: (1) *B’s IP address* (2) *1st hop router R’s IP address*: (3) *R’s MAC address***  
2. A creates datagram with SRC=(A) | DEST=(B) , then encapsulates in a frame with dest=(R’s MAC + datagram)

3. A sends frame to R: R receives frame, detaches datagram from the frame.  
4. R detaches datagram from the frame, creates link-layer frame with B’s MAC address as dest + datagram

5. R forwards frame to B, then B detaches the datagram from the frame. The data has reached the destination.

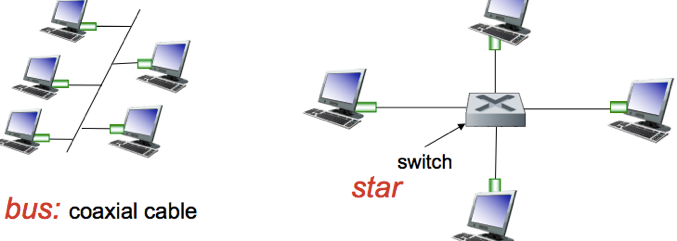
****

**Ethernet**

There are many different Ethernet standards for different physical media with different speeds: e.g. fibre, cable.

However, they all use a common MAC protocol and frame format.

**Ethernet Topology**



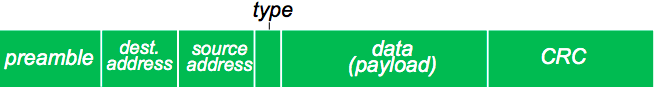
**BUS:** Popular through mid 90s, where all nodes can collide with each other. CSMA/CD for media access control.

**STAR**: Used today, where there is an active SWITCH in the centre.

* Each “spoke” runs a separate Ethernet protocol, so nodes do not collide with each other. No sharing, no CSMA/CD.

**Ethernet Frame Structure**

The sender encapsulates the IP datagram (or other network layer protocol packet) in an **Ethernet Frame**



Preamble (7 bytes): Used to sync receiver / sender clock rates. | Addresses (6-bytes): Source / Destination MAC address.

Type: Indicates if there is a higher-layer protocol | CRC: Cyclic Redundancy Check at the receiver.

**Ethernet: Unreliable + Connectionless**

Connectionless: No handshaking between NICs sending and NICs receiving.

Unreliable: A receiving NIC doesn’t respond with ACKs/NAKs to the sender. Data in dropped frames recovered only if the initial sender uses a higher-layer RDT method, otherwise the data is lost.

Ethernet’s MAC protocol: Un-slotted CSMA/CD with binary back-off.

**LAN: Ethernet Switches**

**Link-Layer-Devices**: Stores and forwards Ethernet frames. Examines incoming frame’s MAC 🡪 forwards frame to outgoing link.

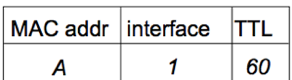
**Transparent**: Hosts are unaware of the presence of switches.

**Plug-and-play, Self-Learning**: Switches do not need to be configured.

**Switches: Multiple Simultaneous Transmissions + Forwarding table**

Hosts have a dedicated link / connection to the switch. Link has FULL-DUPLEX and no collisions, as each link is its own domain.

Each switch has a **Switch Table: <MAC address of host, interface of host, TTL>**



**Switches**: **Self-Learning**

A switch learns which hosts can be reached through which interfaces.

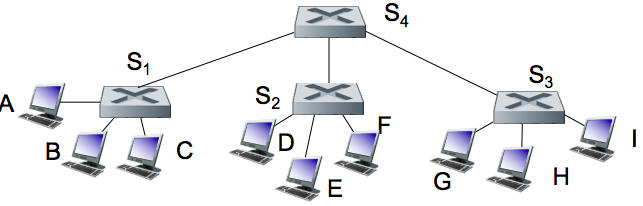
* When a frame is received, a switch learns the location of the sender + records sender/location pair in ^switch table.

When a frame is received at the switch:

1. Record the incoming link, MAC address of the sender.
2. Index the switch table using the MAC destination address.
3. **IF** entry is found **{**   
    **IF** dest MAC on segment exists in the switch’s table / comes from same port 🡪 drop/filter packet  
    **ELSE** forward frame on the interface indicated by the entry. **} ELSE** flood /\* forward frame to all the interfaces except the arriving interface \*/

**Switches: Multiple Interconnected switches**

Switches can be connected together i.e. three routers in one house.



**Q:** Sending from A-to-G, how does S1 know to forward frame to G via. S4 and S3?

**A: Self-learning again**! Works exactly the same way as in the single-switch case.

**Switch Poisoning (DoS)**: Attacker fills up a switch table with bogus entries by sending large # of frames with bogus source MAC addresses. Since the switch table is full, genuine packets frequently need to be broadcasted as previous real entries are wiped.

**Wireless Networks**

Two important challenges:

* + 1. **Communication** over a wireless link (2) **Mobility:** Handling mobile user who changes point of attachment to the network.

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

**WaveLength** = **C / f**  , where C = speed of light | f = frequency

**Elements of a wireless network**

* Wireless Hosts: Laptops, smartphones, running applications (either stationary or mobile)
* Base Station: Cell towers, 802.11 Access Points
  + Typically connected to the Wired Network
  + **Relay:** responsible for sending packets between wired / wireless hosts in its “area”
* Wireless Link: Typically used to connect mobiles to Base Stations
  + Multiple Access Protocol coordinates link access.
  + Various data rates + transmission distances.
* MODE #1: Infrastructure Mode: Base Station connects mobiles into the Wired Network
  + **Handoff**: mobile changes Base Stations that provide them with a connection into the wired network.
* MODE #2: Ad-Hoc Mode: No Base Stations.
  + Nodes can only transmit to others within link coverage. Route among themselves.

**Wireless link characteristics**

Characteristics make wireless communication difficult:

* Decreased signal strength / path loss: radio signals lose affect as it propagates through matter.
* Interference from others sources: wireless network frequencies shared by other devices e.g. phones can interfere.
* Multipath Propagation: Radio signals reflect off objects, arriving at the dest at slightly different times.

Path loss / Attenuation:

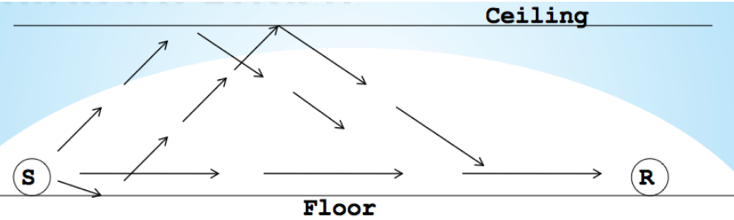
* Things that can affect attenuation:  
  Reflection, diffraction, absorption | terrain contours (urban/rural) | humidity

****

Multipath Effects:

Signals bounce off surface and interfere with one another.

Self-interference.

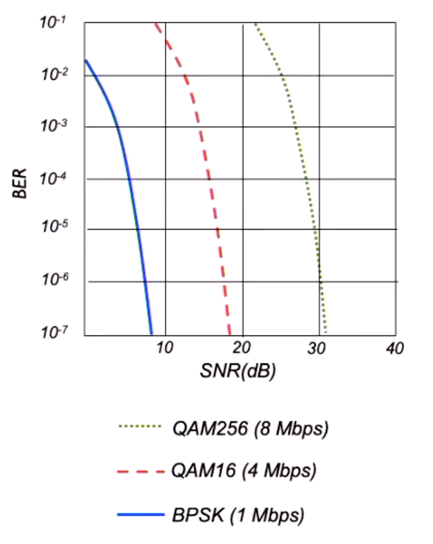


**Signal-To-Noise-Ratio (SNR):** Ratio between max signal strength that a wireless connection can achieve + noise present in connection. Larger SNR = better signal / easier to extract signals from noise.

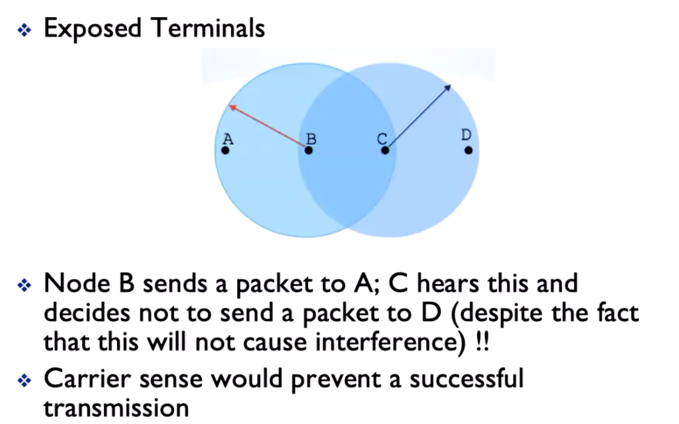
**Bit Error Rate (BER)**: #bit errors per unit of time.

**SNR vs BER Trade-off**:

* Given a physical layer: AIM is to increase SNR + decrease BER
* Given an SNR: AIM is to choose a physical layer that meets BER requirement + giving highest throughput.

****

* When using QAM256, if the BER increases you gotta switch to a slower QAM16



***Exposed Terminal Problem***

* CSMA can prevent successful transmission
* Example right:
  + If B 🡪 A, and C 🡪 D, it should be okay because theres no interference at the receiver
    - However, with CSMA, C will hear the transmission from B 🡪 A and won’t send a packet

***802.11 Wireless LAN***

**Joining a network**

Passive Scanning:

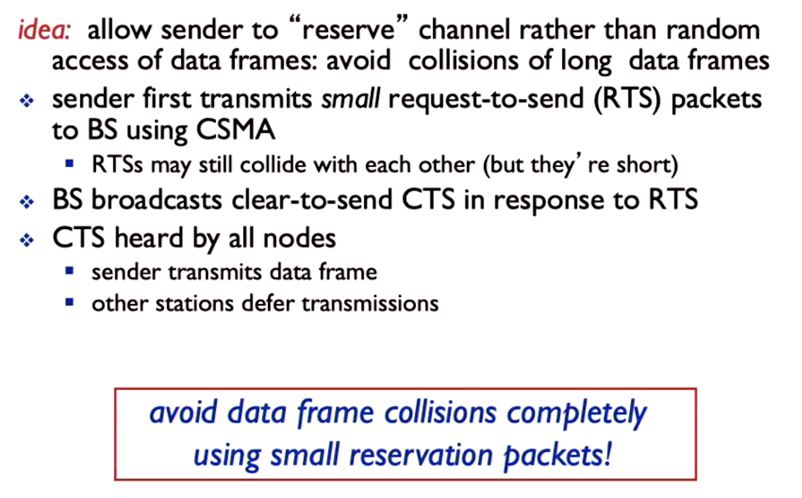
* Beacon frames sent from Ap’s (Access Points like routers)
* Association request frame sent from client to selected AP
  + Looks for the highest SNR AP
* Association response frames sent from AP to client

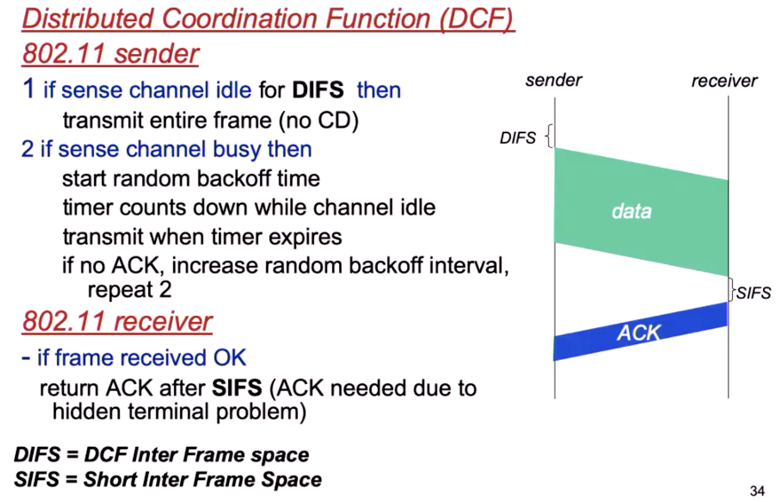
Active Scanning:

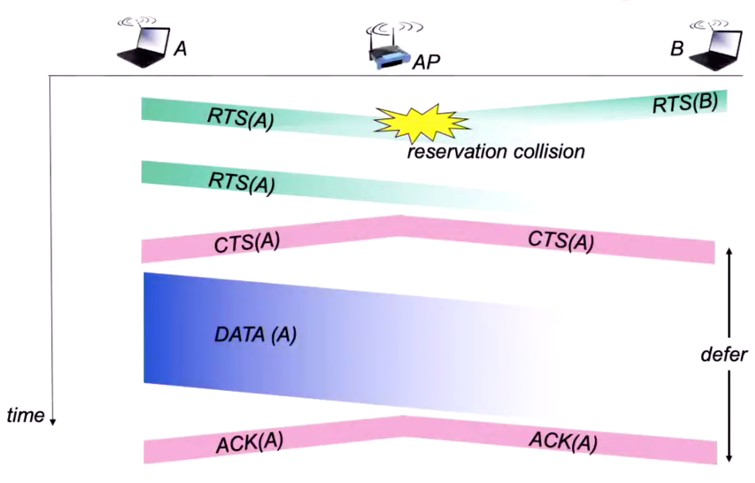
* Probe request frame broadcast from client
* Probe response frames sent from AP to client
* Association request frame sent from client to selected AP
* Association response frame sent from AP to client

**Collision Avoidance in Wireless Networks**

* Collisions occur are the receiver







**Day in the Life**

1. ARP to get mac addresses
2. DNS to get IP of dest
   1. Uses UDP
3. HTTP request to server
   1. Uses TCP