CS2105 Comp. Networks Notes

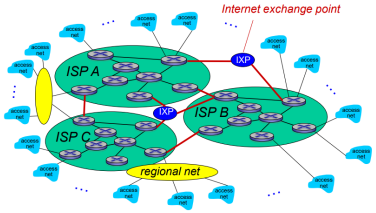
AY23/24 Sem 1. github.com/gerteck

1. Computer Networks Introduction

- Fundamental concepts and principles behind computer networking, with internet as case study.
- Connected by commmunication links and packet switches.

Internet

- The Internet is a network of connected computing devices (e.g. PC, server, laptop, smartphone).
- Such devices are known as hosts or end systems. Hosts run network applications (e.g. Tele, browser, Zoom)
- Packet switching network, users' packets share network resources that are used on demand. Excessive congestion is possible.
- **Network of networks** Hosts connect to Internet via access ISPs (Internet Service Providers), which themselves are interconnected.



Network Edge (Access Network)

- The access network is the network that physically connects an end system to the first router on a path from that end system to any distant end system.
- E.g. Residential access networks, mobile access networks.

Network Core

- A mesh of interconnected routers.
- Data is transmitted through network through:
- **Circuit switching:** dedicated circuit per call
- **Packet switching:** data sent through net in discrete "chunks"

Circuit Switching

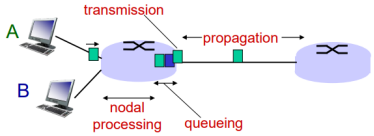
- End-end resources are allocated to and reserved for "call" between source & dest:
- call setup required, circuit-like (guaranteed) performance
- circuit segment idle **silent periods** if not used by call (no sharing)
- commonly used in traditional telephone network

Packet Switching

- Host sending function: breaks application message into smaller chunks, known as packets, of length L bits. Then, transmits packets onto the link at transmission rate R.
- Packets are passed from one router to the next, across links on path from source to destination.
- link transmission rate is aka link capacity or **link bandwidth**
- **Packet transmission delay** (time taken to transmit L-bit packet into Link) = $\frac{L}{R}$
- **Store-and-forward**: entire packet must arrive at a router before it can be transmitted on the next link.

Delay, Loss, Throughput

Four sources of packet delay, suffered at each node, from source to destination. (**Total nodal delay**)



- **Nodal Processing Delay**: Time required to check for bit errors and determine output link. Typically $<$ msec.
- **Queueing Delay**: Time waiting in queue for transmission, depends on earlier arrived packets. Typically $<$ msec. Additionally, possibility of packet loss if router queue full (buffer has reached finite capacity), and drops packet.
- **Transmission Delay**: Time to push (last bit) of packet on wire. $d_{trans} = \frac{L}{R}$.
- **Propagation Delay**: Time to propagate to next router. $d_{prop} = \frac{d}{s}$, where d is length of physical link, s propagation speed.
- End to end packet delay is total time taken for packet to travel from source to destination, consisting of the 4 delays.
- **Throughput**: How many bits can be transmitted per unit time, usually measured for end-to-end communication (as opposed to bandwidth for specific link).

Internet Protocol Stack

Protcols logically organised into 5 "layers" according to purpose. (Additionally presentation and session layers not included)

- **Application**: Where network applications and app-layer protocols reside. Packet here called message.
  Examples: HTTP, SMTP, FTP
- **Transport**: Transports app-layer messages between application endpoints. Packet here called segment.
  Examples: TCP, UDP
- **Network**: Moves packets (datagrams) from one host to another. Includes IP protocol and other routing protocols.
- **Link**: Moves packet from one node to another. Packet here called frame.
  Example: Ethernet, WiFi
- **Physical**: Moves individual bits within link-layer frame from one node to another. Link and transmission medium dependent.

2. Application Layer

Principles of Network Applications

- **Client-Server**:
  - Server waits for incoming requests and provides required services to client. Easy scalability.
  - Client initiates contact with server and requests service.
- **Peer-To-Peer (P2P)**:
  - No dedicated server, instead it relies on direct communication between pairs of intermittently connected hosts called peers.
  - Self-scalability, each peer generates workload but adds service capacity by distributing files.

Process Communication

- **Socket**: A software interface that process uses to send messages and receive messages from network. Generally a combination of IP address and port number.
- **IP Address**: A 32-bit quantity, uniquely identifies host.
- **Port Number**: A 16-bit integer used to identify a receiving process running in a host.

Requirements of Transport Service

- **Reliable Data Transfer**: Data to sent correctly and completely vs. loss-tolerant.
- **Throughput**: Bandwidth-sensitive apps may need guaranteed throughput of r bits/sec.
- **Timing/Delay**: Real-time applications generally require low delays to be effective.
- **Security**: Encryption, data integrity, authentication.

Transport Layer Protocols

Two main protocols for the Internet.

- **Transmission Control Protocol (TCP)**
  - Reliable data transfer, Connection-oriented service: A handshake required.
  - Flow control, Congestion control: Throttle sender when network overloaded
  - Security: Can be enhanced at the app layer with Secure Sockets Layer
  - Does not provide: Timing and throughput guarantee
- **User Datagram Protocol (UDP)**
  - Unreliable data transfer
  - Connectionless: No handshake
  - No flow control, no congestion control
  - Does not provide: Timing and throughput guarantee, security.

Application-Layer Protocols

An application-layer protocol defines:

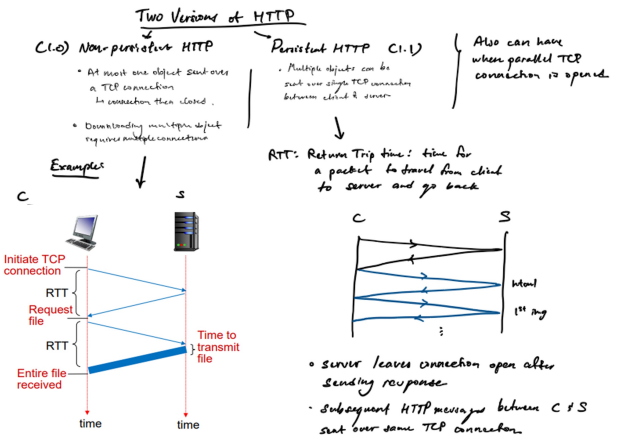
- Types of messages exchanged, e.g. request and response messages.
- Syntax of message types, e.g. fields and how they are delineated.
- Semantics of the fields, i.e. what the information means.
- Rules for when and how to send a message and respond to messages.

Web & HTTP

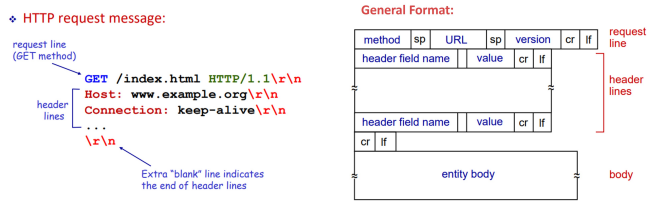
- **Webpage**: Consists of base HTML files and referenced objects. Addressable by URL.
- URL made up of hostname as well as path name. (E.g. http://www.comp.nus.edu.sg/ cs2105/img/doge.jpg)
- **HyperText Transfer Protocol** is Web's app-layer protocol.
- **Client-server model:** Client requests, receives and displays Web objects, server is Web server that sends objects in response.
- **Stateless**: server maintains no information about clients, and **Reliable**: Over TCP.
- **Three-way Handshake**: Client sends small TCP segment to ask for connection, server acknowledges and responds, client acknowledges and sends it back with request message.

HTTP versions

- **RTT**: Round trip time, time taken for packet to travel from server and back to client, does not include transmission delay.
- **Non-persistent HTTP**: Response time = $2*$ RTT$+$ file transmission time (per object)
- **Persistent HTTP**: Server leaves connection open after sending response, subsequent HTTP sent over same TCP connection. Also uses pipelining, send requests back to back.

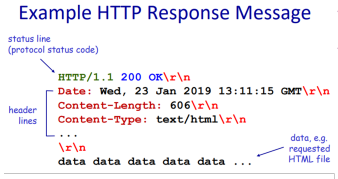


HTTP Request



- **HTTP 1.0 Methods**: GET (gets object), POST (posts form data), HEAD (gets header without body).
- **HTTP 1.1 Methods**: GET, POST, HEAD, PUT (uploads file to path specified), DELETE

HTTP Response



- Status code appears in 1st line in server-to-client response message.
- Some sample codes:

200 OK
- request succeeded, requested object later in this msg

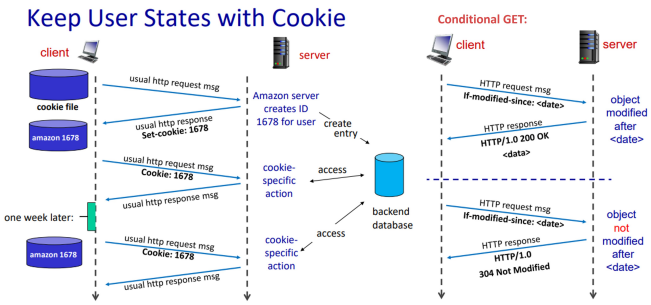
301 Moved Permanently
- requested object moved, new location specified later in this msg (Location:)

403 Forbidden
- server declines to show the requested webpage

404 Not Found
- requested document not found on this server

Cookies

- HTTP is designed to be "stateless", server maintains no information about past client requests.
- Good to maintain states over multiple transactions, e.g. shopping carts
- **Cookie**: http messages carry "state":
  1) cookie header field of HTTP req/res messages
  2) cookie file kept on user host, managed by browser
  3) back-end database at Web site
- **Conditional GET**: In cache, specify date of cached copy in HTTP request. Server response contains no object if cached copy is up to date.



Domain Name System

DNS translates between hostname and IP addresses. Client must carry out a DNS query to determine the IP address corresponding to the server name.

- **DNS: Resource Records (RR)**
  - Mapping between host names and IP addresses (and others) are stored as resource records (RR).

RR format: `(name, value, type, ttl)`

type = A
- `name` is hostname
- `value` is IP address

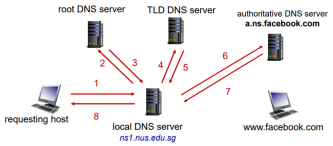
type = NS
- `name` is domain (e.g., `nus.edu.sg`)
- `value` is hostname of authoritative name server for this domain

type = CNAME
- `name` is alias name (e.g. `www.nus.edu.sg`) for some "canonical" (the real) name
- `value` is canonical name (e.g. `mgnzsqc.x.incapdns.net`)

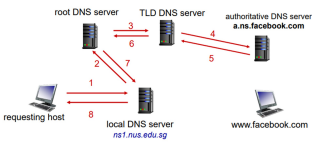
type = MX
- `value` is name of mail server associated with `name`

- **Distributed, Hierarchical Database**: DNS servers form hierarchy to distribute mappings. Contains root servers (for Top Level Domain TLD servers), TLD servers (e.g. uk, sg), Authoritative servers.
- Local DNS servers have local cache, acts as proxy, forward query into hierarchy if answer not found locally.
- **DNS Caching**: Cache mapping, which expires after some time (TTL: time to live).
- DNS runs over **UDP**.

DNS Name Resolution



- This is known as *iterative query*.



- This is known as *recursive query*.
  - rarely used in practice

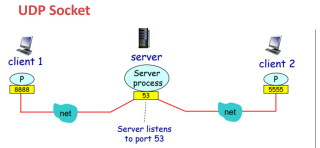
Socket Programming

Applications treat the Internet as black box, send/receive message through sockets.

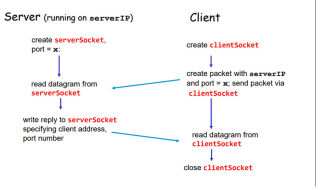
- Two types of sockets
- **TCP**: reliable, byte stream-oriented socket
- **UDP**: unreliable datagram socket

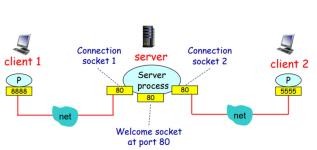
UDP vs. TCP Socket

- **UDP Socket:**Sender attach des IP address + port no. to each packet. (OS inserts add. info source IP and port). Receiver extracts sender IP + port number from packet.
- **TCP Socket:** Attempts to establish TCP connection to server first. Server TCP contacted creates new socket to communicate with client, allows server to talk with multiple clients individually.

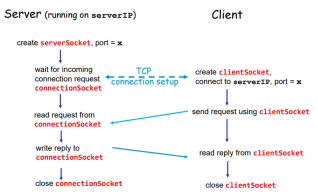


UDP: Client/server Socket Interaction





TCP: Client/server Socket Interaction



TCP vs. UDP Differences

- In TCP, two processes communicate as if pipe between them. The pipe remains in place until one of two processes closes it. Sending process doesn't need to attach a destination IP / port number to the bytes in sending attempt as the logical pipe has been established
- In UDP, programmers need to form UDP datagram packets explicitly and attach destination IP address / port number to every packet.

3. Transport

Transport Layer Services

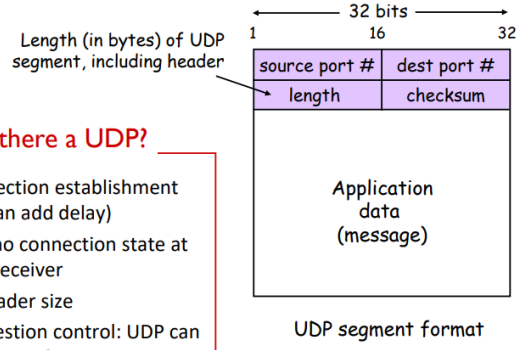
- Transport layer protocols run in hosts.
- Sender side: breaks app message into segments (as needed), passes them to network layer (aka IP layer).
- Receiver side: reassembles segments into message, passes it to app layer.
- Packet switches (routers) in between: only check destination IP address to decide routing. running on different hosts

Transport and Network Layer

- **Transport** layer takes care of logical communication between **processes**.
- **Network** layer takes care of logical communication between **hosts**. (best-effort, unreliable)
- **IP Datagram**: Contains source and dest IP addresses, carries one transport layer segment that contains source and dest port numbers.

UDP: Connectionless Transport

- UDP adds very little service on top of IP.
- **Multiplexing at sender**: UDP gathers data, forms packets, passes to IP.
- **De-multiplexing at receiver**: UDP receives packets from lower layer, checks dest port, and dispatches them to right processes.
- **Unreliable**: UDP transmission used by loss tolerant and rate sensitive apps.



Why is there a UDP?

- No connection establishment (which can add delay)
- Simple: no connection state at sender, receiver
- Small header size
- No congestion control: UDP can blast away as fast as desired

UDP Checksum

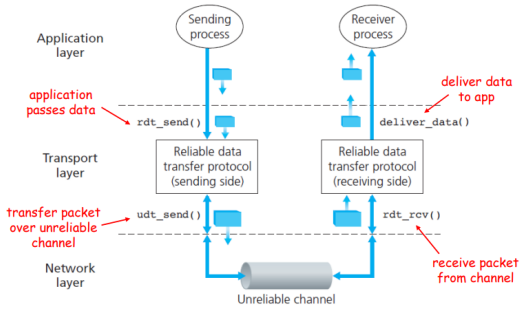
- Allows for error detection, but not correction. May be bit errors when segments are stored and passed in router memory.
- Treat UDP segment as a sequence of 16-bit integers.
- Apply binary addition on every 16-bit integer (checksum field currently 0).
- If carry from MSB, add 1 to result (wrap).
- Compute 1's complement to get UDP checksum.

Principles of Reliable Data Transfer (rdt)

We need to build a reliable transport layer protocol on top of unreliable communication.

- Factors: **Packet corruption, Packet loss, Packet reordering, Packet (Long) Delay.**
- Finite State Machines to describe protocol.

Reliable Data Transfer: **Service Model**



rdt 1.0 (Perfectly Reliable)

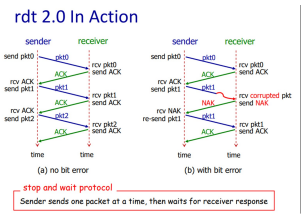
- Assumption: Underlying channel perfectly reliable.
- Sender creates packet and sends, Receiver extracts and deliver data to application.

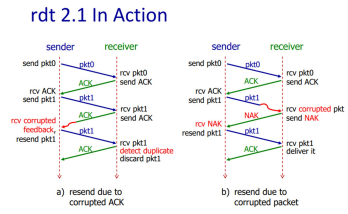
rdt 2.0 (Corruptable Data)

- Assumption: Underlying channel may flip bits. Use **stop and wait** (for receiver response) protocol.
- Receiver uses checksum to detect bit errors, sends NAK if corrupted. Sender resends if NAK received.
- **Problem: If ACK or NACK corrupted**, no guaranteed way to recover. If packet resent, the receiver will not know it's a duplicate.

rdt 2.1

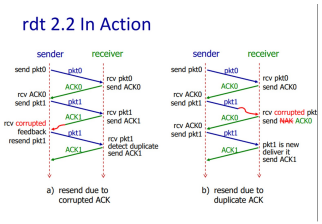
- Add **sequence number to packet**, alternate 1 & 0. Sequence number detects duplicates.
- Same as rdt2.0, but receiver knows if it is duplicate.

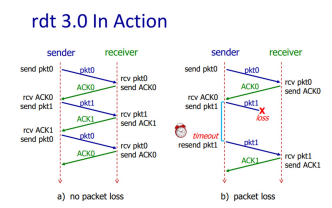


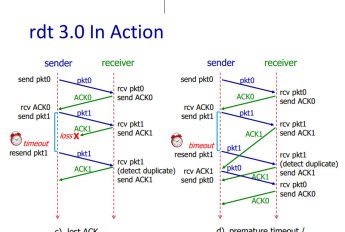


rdt 2.2

- **Use ACK of last packet sequence number for NAK.**
- Receiver explicitly include seq. no, duplicate ACKs results in retransmit current pkt.







rdt 3.0 (Corruptable, Lossy, Delay)

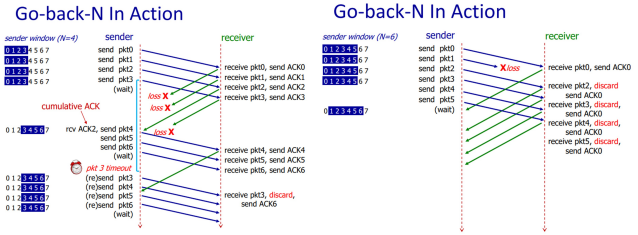
- Assume corruption, packet loss/delay, no re-order.
- To detect packet loss, use **sender timeout**. Sender retransmits if no ACK received till timeout.
- If packet/ACK just delayed, retransmission may generate duplicates but receiver can use seq. no. to detect.
- **rdt 3.0 performance**: Utilisation rate of sender low. For RTT 30ms, L = 8000b, link 1GBps, d_{trans} = 0.008ms, send 8000 bits per 30.008ms. Utilisation 0.027%.
- Stop and Wait limits use of physical resources.

Pipelining

- **Pipelined Protocols**: sender allows multiple, "in-flight", yet to-be-acknowledged packets.
  - range of sequence numbers must be increased
  - buffering at sender and/or receiver
- Number of packets sent at once is called **window size**.
- **Benchmarked Pipelined Protocols**: Go-Back-N (GBN), Selective Repeat (SR).
- Assumption of corruption, packet loss / delay.

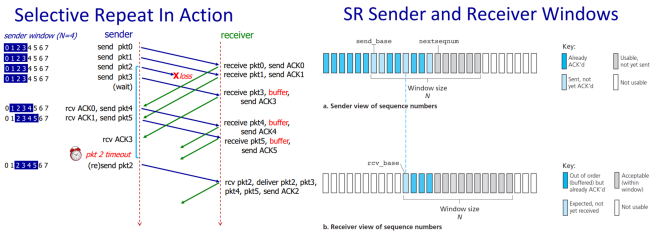
Go-back-N

- Sliding window, slides forward only when ACK received for the leftmost packet in window.
- Requires k bits in packet header for 2^k sequence number.
- Sender keeps only 1 timer for oldest unACKed packet.
- Receiver only accepts ACK packets that arrive in order, discards out of order packets ACK last in-order sequence number. (cumulative ACK).



Selective Repeat

- Receiver individually acknowledges all correctly received packets.
- Buffers out-of-order packets, for eventual in-order delivery to upper layer.
- Sender maintains timer for each unACKed packet, When timer expires, retransmit only unACKed packet.

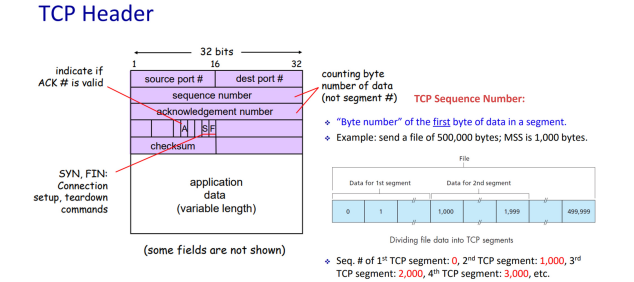


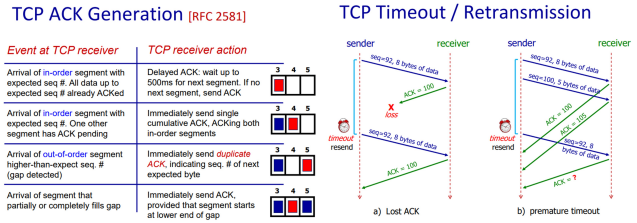
TCP: Connection-oriented Transport

- **Connection oriented**: handshake before sending data. **Point to point**: one sender, one receiver. The connection is **duplex** (bidirectional data flow). **Reliable in-order**.
- TCP socket is fully identified by four-tuple: (source IP addr, source port no., dest IP addr, dest port no.).
- **Multiplexing**: TCP gathers data from processes, form transport-layer segments including app data and 4-tuple pass to network layer.
- **Demultiplexing**: Connection socket created, server already noted 4-tuple. Subsequent packets directed, or demultiplexed, to the appropriate socket using those 4 values.
- TCP creates **buffers** after handshaking.

TCP Segment / Header

- The maximum segment size (MSS) depends on maximum transmission unit (MTU).
- Generally MSS is 1460 bytes, (MTU is 1500 bytes for Ethernet and PPP link-layer protocols.) 40 bytes split half for TCP and IP header.
- TCP Seq. no is "byte no.", first b of data in segment.
- TCP Ack. no is "seq no." of next b expected by receiver.
- Checksum computation uses 1s complement (UDP same)





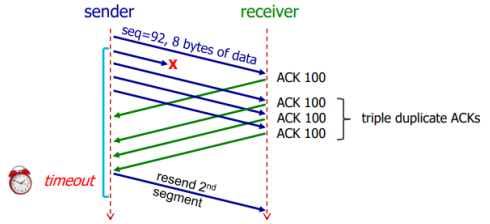
- Random initial sequence number: Minimise probability of some segment from previous connection mistaken as from current connection

TCP Timeout Value

- Determining TCP appropriate timeout value:
- too short timeout: premature timeout and unnecessary retransmissions.
- too long timeout: slow reaction to segment loss. Timeout interval must be longer than RTT – but RTT varies!
- TCP computes (and keeps updating) timeout interval based on estimated RTT. (TimeoutInterval = EstimatedRTT + 4*DevRTT)

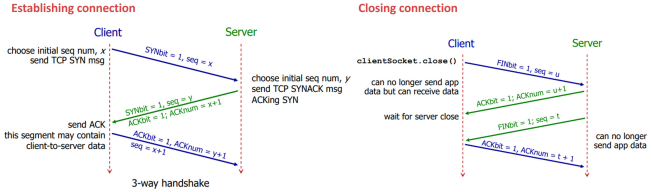
TCP Fast Retransmission

- Timeout period is often relatively long. long delay before resending lost packet.
- **Fast retransmission**: If sender receives 4 ACKs for same segment, suppose segment is lost, resend segment (even before timer expires).



TCP Handshake / Closing

- Before exchanging app data, TCP sender and receiver "shake hands", agree on connection and exchange connection parameters.
- Closing: Client, server each close their side of connection, send TCP segment with FIN bit = 1



4. Network

Network Layer Services

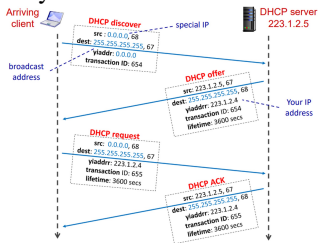
- Network layer delivers packets to receiving hosts.
- Routers examine header fields of IP datagrams passing it.
- **Forwarding**: Moving of incoming packet to appropriate output link.
- **Routing**: Calculation of path taken by packets from sender to receiver.

IP Address

- **IP address** used to identify host / (router), 32-bit integer expressed in binary/decimal.
- Host gets an IP address either through manual configuration by sys admin, or auto assigned by a **DHCP** server.

Some Special IP Addresses

Special Addresses	Present Use
0.0.0.0/8	Non-routable meta-address for special use
127.0.0.0/8	Loopback address. A datagram sent to an address within this block loops back inside the host. This is ordinarily implemented using only 127.0.0.1/32.
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	Private addresses, can be used without any coordination with IANA or an Internet registry.
255.255.255.255/32	Broadcast address. All hosts on the same subnet receive a datagram with such a destination address.



DHCP: Dynamic Host Configuration Protocol

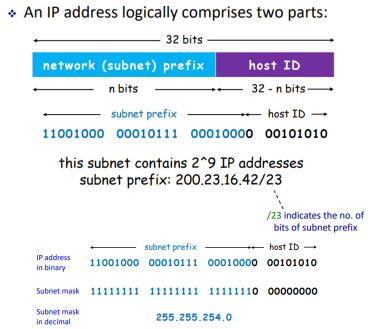
- **DHCP** allows a host to dynamically obtain its IP address from DHCP server when it joins network.
- IP address is renewable, allow reuse of addresses (only hold address while connected), support mobile users to join network.
- DHCP: 4-step process: Host broadcasts "DHCP discover" message, server responds with "DHCP offer" message, Host requests IP address: "DHCP request" message, DHCP server sends address: "DHCP ACK" message
- DHCP may provide host additional network information, e.g. IP of first-hop router, local DNS server, network mask.
- DHCP runs over **UDP**. DHCP server port 67, client port 68.

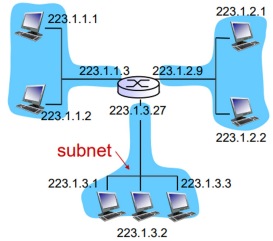
IP Address & Network Interface

- IP address is associated with a network interface.
- Host usually has one or two network interfaces (e.g. wired Ethernet and WiFi), A router typically has multiple interfaces.
- **IP Addr** comprises network/subnet prefix and host ID.

Subnet

- **Subnet** is a network formed by a group of "directly" interconnected hosts.
- Hosts in same subnet have same network prefix of IP addr, can physically reach each other without intervening router. They connect to the outside world through a router
- **Classless Inter-domain Routing** (CIDR): Internet's IP address assignment strategy.
- Subnet prefix of IP addr of arbitrary length, Address format: $a.b.c.d/x$, where x is the no. of bits in subnet prefix of IP addr.
- **Subnet mask** is used to determine which subnet an IP address belongs to.
- made by setting all subnet prefix bits to "1"s and host ID bits to "0"s.





A network consisting of 3 subnets (first 24 bits of IP addr. are network prefix)

IP Address Allocation

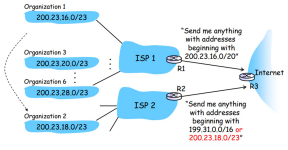
- Organization can buy from registry / rent from ISP's addr space to obtain block of IP addr.
- **Hierarchical Addressing**: Allows efficient way of routing.

IP Address Allocation

	Binary Address	Decimal Address
ISP's block	11001000 00010111 0001 000 0 00000000	200.23.16.0/20
Organization 1	11001000 00010111 0001 **000** 0 00000000	200.23.16.0/23
Organization 2	11001000 00010111 0001 **001** 0 00000000	200.23.18.0/23
Organization 3	11001000 00010111 0001 **010** 0 00000000	200.23.20.0/23
...	...	...
Organization 6	11001000 00010111 0001 **101** 0 00000000	200.23.28.0/23

use 3 more bits to differentiate 6 organizations

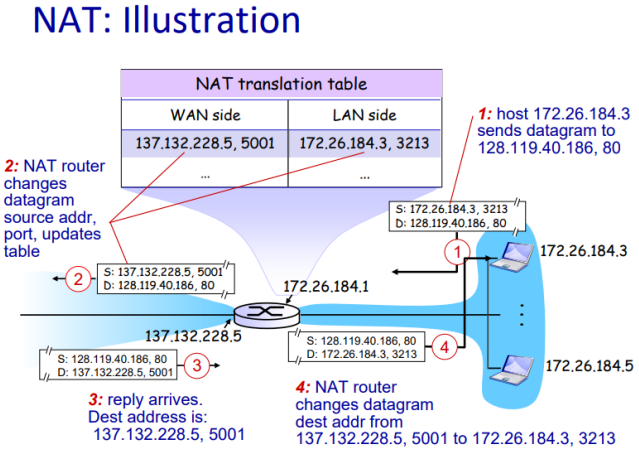
Hierarchical Addressing



- **Longest Prefix Match**: Choose one with longer match. If IP addrr matches all 23 bits for org, packet forwarded, else forwarded to latter.

Network Address Translation (NAT)

- Map IP addr space by modifying network addr info in packets IP header through traffic routing device. NAT Routers must:
- **Replace** (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #), **Remember** (in NAT translation table) the mapping and **Replace** destination fields of every incoming datagram with that stored in NAT translation table.
- **Benefits**: Single public IP for NAT router allows multiple private IP address. Change (private IP) addr of hosts in local network without notifying outside world.
- ISP change w/o changing local host addresses in local network.
- Hosts inside local network not explicitly addressable or visible by outside world (security plus).



Routing Algorithms

- The Internet as **"network-of-networks"**, hierarchy of **Autonomous Systems** (AS), e.g., ISPs, each owns routers and links.
- Due to size and decentralized administration of Internet, routing is done hierarchically.
- **Intra-AS routing**: Finds good path btwn two routers within AS. Commonly used protocols: RIP, OSPF
- **Inter-AS routing (not covered)** Handles the interfaces between ASs, standard protocol: BGP.

Intra-AS Routing

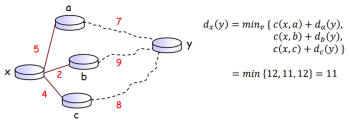
- Abstractly view a network of routers as a graph, vertices are routers, edges are physical links between routers.
- Associate cost to each link. (cost = 1, or inversely related to bandwidth, or related to congestion)
- **Routing**: find least cost path btwn two vertices in graph.
- **Link state Algorithmns**: Centralised routing algo, all routers have complete knowledge of network topology and link costs. Routers periodically broadcast link costs to each other.
- Use Dijkstra algorithm compute least cost path locally! (using global map).
- **Distance vector Algorithms**: Decentralised routing algo, Routers know physically-connected neighbors and link costs to neighbors.
- Routers exchange "local views" with neighbors, update own "local views". Iterative computation: Swap local view with direct neighbours, Update own view, Repeat till no further change.

Distance Vector Algo (Bellman Ford)

Bellman-Ford Equation

$$d_x(y) = min_v \{c(x,v) + d_v(y)\}$$

where min is taken over all direct neighbors v of x

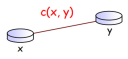


$$d_x(y) = min_v \{c(x,a) + d_a(y), c(x,b) + d_b(y), c(x,c) + d_c(y)\}$$

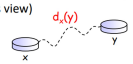
$$= min\{12, 11, 12\} = 11$$

Graph Notation:

- c(x, y): the cost of link between routers x and y
  - = ∞ if x and y are not direct neighbours



- $d_x(y)$: the cost of the least-cost path from x to y (from x's view)



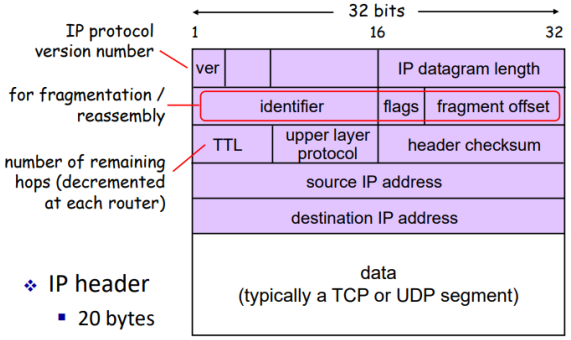
- $d_x(y) = min_v\{c(x,v) + d_v(y)\}$
- To find **least cost path**, x needs to know cost from each of its direct neighbour to y. Each neighbour v sends its distance vector (y, k) to x, telling x that the cost from v to y is k.
- Every router, x, y, z, sends its distance vectors to directly connected neighbors. When x finds y is advertising cheaper path to z than known, x update distance vector to z accordingly and note down all packets for z should be sent to y. Info used to create forwarding table of x.
- After every router exchanged several rounds of updates with direct neighbors, all routers will know least-cost paths to all other routers.

RIP (Routing Information Protocol)

- **RIP** implements the DV algorithm. Uses hop count as the cost metric (insensitive to network congestion).
- Exchange routing table every 30 seconds over UDP port 520.
- "Self-repair": if no update from a neighbour router for 3 minutes, assume neighbour failed.

Internet Protocol (IP): IPv4

IPv4 Datagram Format



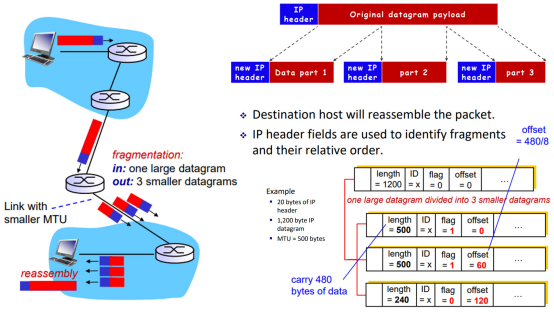
(some fields are not shown)

- Length: of IP datagram + 20b header. Identifier, flags, fragment offset: support fragmentation & reassembly
- TTL: Prevent infinite circulation. Upper layer protocol: Only used at final dest, determine if UDP/TCP (for Internet). Checksum also uses 1s complement.
- **IPv6**: 40b header with 128b IP addr.

IP Fragmentation & Reassembly

- Different links, different MTU (Max Transfer Unit, max amt of data link-level frame can carry).
- "Too large" IP datagrams may be fragmented by routers.

IP Fragmentation Illustration:



- **Flag**(frag flag) is set to 1 if next fragment from same segment, 0 if this is the last fragment.
- **Offset** is in expressed in unit of 8-bytes

Internet Control Message Protocol

- **ICMP**: used by hosts & routers to communicate network-level information.
- **Error reporting**: unreachable host / network / port / protocol.
- Echo request/reply (used by ping).
- ICMP messages carried in IP datagrams, ICMP header starts after IP header.

ICMP Type and Code

- ICMP header: Type + Code + Checksum + others.

Type	Code	Description
8	0	echo request (ping)
0	0	echo reply (ping)
3	1	dest host unreachable
3	3	dest port unreachable
11	0	TTL expired
12	0	bad IP header

Selected ICMP Type and subtype (Code)

5. Link

- **Link Layer**: Concerned about electronics of sending and receiving binary data over a communication channel.
- **Communication Channel**: transmission medium of data signals (e.g. copper wire, satellite, optical fiber)
- **Node**: Devices exchanging data. (E.g. hosts, routers).
- **Link:** Comm. channels that connect adjacent nodes.

Link Layer

- **Link layer** sends datagram between adjacent nodes (hosts or routers) over a single link. Responsible for transfer of datagram from one node to physically adjacent node over link.
- **Frames (layer 2 packet)**: IP datagrams are encapsulated in link-layer **frames** for transmission.
- **Protocols**: Different link-layer protocols may be used on different links, each protocol may provide a different set of services.

Link Layer Services

- **Framing**: Encapsulate datagram into frame, adding header and trailer.
- **Link access control**: When multiple nodes share a single link, need to coordinate which nodes can send frames at a certain point of time.
- **Error detection**: Errors are usually caused by signal attenuation or noise. Receiver detects presence of errors, and may signal sender for retransmission or simply drops frame.
- **Error correction**: Receiver identifies and corrects bit error(s) without resorting to retransmission.
- **Reliable delivery**: Seldom used on low bit-error link (e.g., fiber) but often used on error-prone links (e.g., wireless link).

Network Adapter

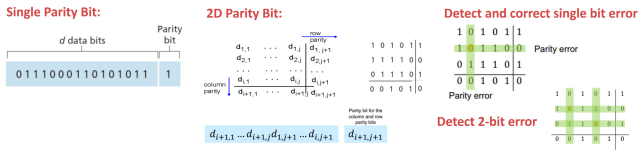
- **Network Adapter**, aka network interface card (NIC), is a single, special-purpose chip that implements the link-layer services above. (e.g. Ethernet card, Wifi Adapter).
- Semi-autonomous, implementing both link & physical layers. Many services implemented in hardware.

Error Detection and Correction Techniques

- **EDC**: Error Detection and Correction Bits.
- **D**: Data protected by error checking, may include header.
- Larger EDC fields added to link layer frame yields better detection (and correction), but larger overhead.
- **Common error detection schemes**: Checksum (used in TCP/UDP/IP), Parity checking, CRC (link layer).
- **Checksum Recap**: treate segment contents as 16 bit int sequences, get 1s complement of sum of segment contents.

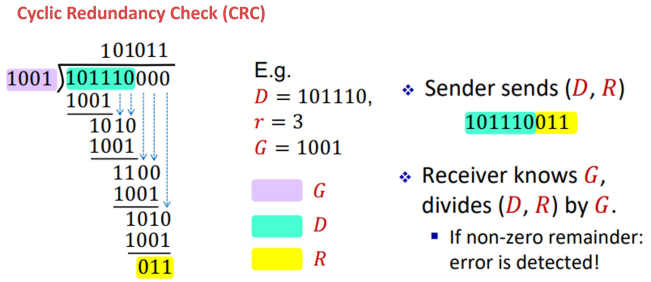
Parity Checking

- **Single Bit Parity**: 1 parity bit for the data.
  – **Even Parity Scheme**: Choose the bit to make the total number of 1s even.
  – **Odd Parity Scheme**: Same but we make the number odd.
- Single bit parity can detect odd number of single bit errors, but cannot detect even single bit errors.
- Errors often clustered together in "bursts", probability of undetected errors in a frame can approach 50%.
- **Two-Dimensional Parity**: Divide the datainto rows and columns, and repeat the above but have parity bits for each column and each row.
  – Can detect and correct single bit errors in data.
  – Can detect two-bit errors.



Cyclic Redundancy Check (CRC)

- "Long division", division replaced by bitwise XOR op.
- **D: d-bit data**, which is also the dividend.
- **G: Generator of r + 1 bits**, which is also the divisor.
- **R: r-bit CRC**, which is also the remainder.
- The resultant $d + r$ bits is "divisible" by G, so the receiver can check for a zero remainder.
- Can detect all odd number of single bit errors.
- Generally done by hardware, so very fast.
- CRC of r bits can detect all burst errors of less than $r + 1$ bits, burst errors greater than r bits with probability $1 - 0.5^r$. Aka polynomial code.

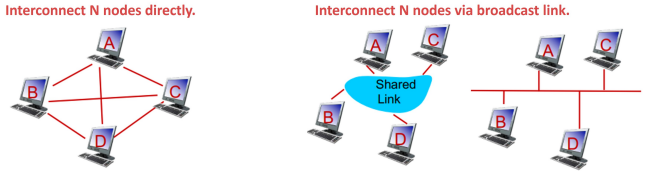


Multiple Access Links and Protocols

- **Multiple Access Protcols:** Categorisable into three broad classes: Random Access, "Taking Turns", Channel Partitioning.
- **Ideal MAP**: Collision free, Efficient, Fairness, Fully Decentralized.
- Additionally, coordination about channel sharing must use channel itself (no out-of-channel signalling).

Connecting N nodes via cable

- **Aim**: Send data between N nodes via cable.
- **Interconnect N nodes directly**: Each link needs to be addressed, N-1 Links needed, does not scale.
- **Interconnect via broadcast link**: Each link needs to be addressed, need to define protocol, need to handle errors.
- Protocol: Framing, Link Access Control. Errors: Detection, Reliability.



Types of Network Links (2)

- **Point-to-point link**: Sender and receiver connected by a dedicated link. No need for multiple access control.
- **Broadcast link:** Multiple nodes connected to same shared broadcast channel. When any one node transmits a frame, all other nodes in the channel receives a copy. We need a Multiple Access Protocol to prevent frame collisions.

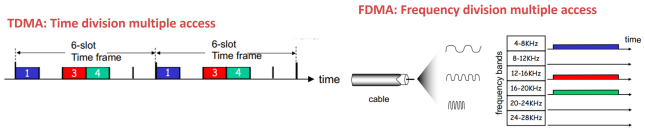
Multiple Access Protocols

Bit Times

- Common unit used in multiple access protocols is bit times.
- Bit transmision time, is time taken to transmit 1 bit.
- Often, used as such: propagation delay is equals to 800 bit times. This means the propagation delay is equals to the time it takes to transmit 800 bits onto the link.

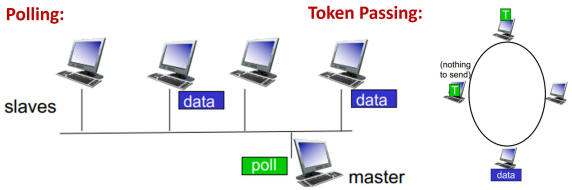
Channel Partitioning Protocols

- **TDMA: Time Division Multiple Access**
- Each node gets fixed length time slot (time frame), where length = frame transmission time. This repeats in rounds. Unused slots go idle.
- **FDMA: Frequency Division Multiple Access**
- Channel spectrum is divided into frequency bands, and each node is assigned one band. Bandwidth has thus decreased, thus transmission is slower. Unused transmission time in frequency bands go idle.
- Both TDMA and FDMA: Collision Free, Inefficient, Perfectly Fair and fully Decentralized.



Taking Turns Protocols

- **Polling**
- A master node invites each of the other nodes (slaves) to transmit in turns. Minor polling overhead. A single point of failure, which is the master node.
- **Token Passing (Token Ring) / Round Robin**
- Control token is passed from one node to the next sequentially. There is overhead for the token and single point of failure as well, which is the token.
- Even if only a few of the nodes have data to send, it can still be quite efficient.



Random Access Protocols

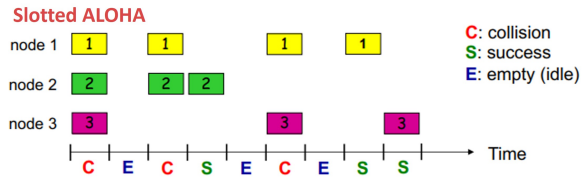
Generally these protocols specify how to detect and recover from collisions (When two or more transmitting nodes). Thus, no need for centralised coordination, thus no single point of failure.

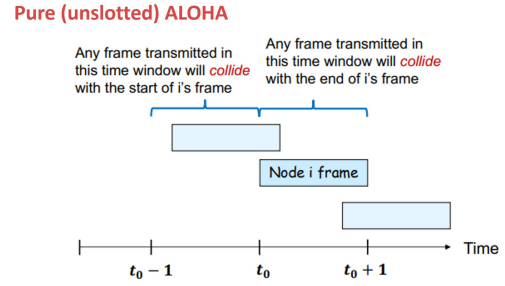
Slotted ALOHA

- Assume all frames are of the same size, and split the time into slots of equal length, where length = time to transmit 1 frame = $L(bits)/R(rate)$.
- A node will only transmit at the start of a slot.
- Each node listens to the channel while transmitting. If a collision occurs, it retransmits in each subsequent slot with probability p until success.
- p depends on network congestion
- **Effectiveness**: Not collision free, Efficiency high when only one node is active, but maximum efficiency falls to 37% when many active nodes (collision & empty slots). Perfectly fair and decentralized.

Pure (Unslotted) ALOHA

- Like ALOHA but no slots nor synchronisation, just transmit when there's a fresh frame.
- Chance of collision increases, as now it can collide with frames both in front and behind $(t_0 - 1, t_0 + 1)$.
- **Effectiveness**: Not collision free, Efficiency high when only one node is active, but maximum efficiency falls to 18% when many active nodes (collision & empty slots). Perfectly fair and decentralized.



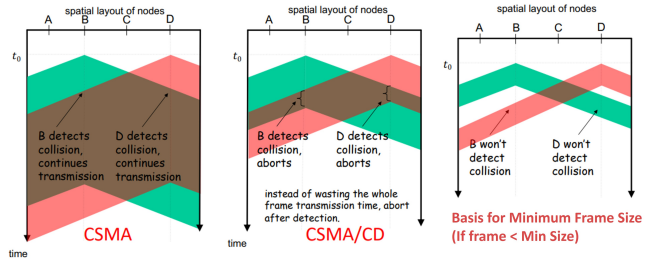


Carrier Sense Multiple Access (CSMA)

- Sense if the channel is **idle**, if so, transmit frame.
- Still collides when two nodes sense that a channel is idle.
- Collision: propagation delay, nodes far apart, two nodes may not hear each other's transmission immediately.
- CSMA transm. does not stop despite collision detection.

Carrier Sense MA/ Collision Detection (CSMA/CD)

- **Backoff Algorithm**: Same as CSMA except the moment a collision is detected, the node stops transmission.
- The node then retransmits after some random amount of time. (probability p for each sub. frame until success).
- **Binary Exponential Backoff**: More collisions implies heavier load, adapt retransmission attempt to estimated current load. Longer back-off interval with more collisions.
- After the m^{th} collision, we choose K at random from $\{0, 1, ..., 2^m - 1\}$, then wait K time units before retransmitting. (Each K has $p = 1/2^m$)
(Ethernet, 1 time unit = 512 bit transmission. times).
- **Effective:** Both Efficent, Fair, Decentralized, but both CSMA/[CD] not collision free.



Minimum Frame Size

- For CSMA and CSMA/CD above, need minimum frame size so that collisions can always be detected.
- Ethernet has a minimum size of 64 bytes.

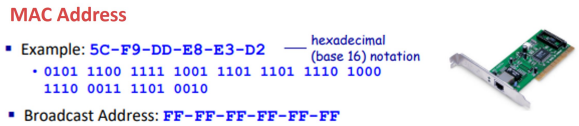
CSMA / Collision Avoidance (CSMA/CA)

- Collision detection can be hard for wireless LANs, as energy levels drop too quickly.
- Hidden Node Problem: When two nodes cannot detect each other but a node in-between encounters a collision.
- As such, an ACK is required from the receiver as well.

Switched Local Area Networks

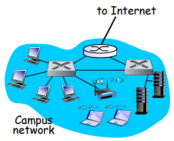
MAC Address

- Every adapter (NIC) has a unique MAC address (aka physical or LAN address).
- **MAC**: Media Access Control.
- Used to send and receive link layer frames, when adapter receives a frame, it checks if the destination MAC address of the frame matches its own MAC address.
- If yes, adapter extracts enclosed datagram and passes it to the protocol stack.
- If no, adapter discards the frame w/o interrupting host.
  - **48 bits:** Burned in NIC ROM (read-only memory).
  - **IEEE**: Administers the MAC address allocation. First three bytes of MAC identifies the vendor of adapter.
  - If somehow MAC is manually configured to be not unique, and the NICs are on the same subnet, transmission will be severely affected.

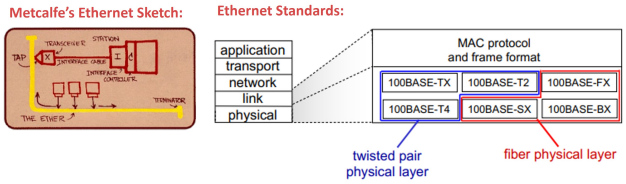


Local Area Network (LAN)

- **LAN** is a computer network that interconnects computers within a geographical area such as office building or university campus.
- Example LAN technologies: IBM Token Ring: IEEE 802.5 standard, Ethernet: IEEE 802.3 standard, Wi-Fi: IEEE 802.11 standard



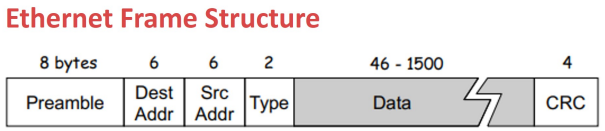
Ethernet



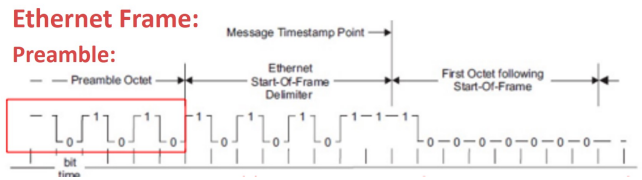
Ethernet

- **Ethernet**: "Dominant" wired LAN technology, developed in mid 1970s, Standardized by Xerox, DEC, Intel in 1978.
- Simpler and cheaper than token ring and ATM (asynch transf mode).
- MAC protocol, frame format remain unchanged over years.

Ethernet Frame Structure Breakdown



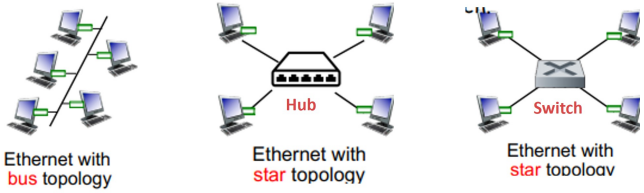
- **MTU**: Maximum Transmission Unit.
- Sending NIC (adapter) encapsulates IP datagram in Ethernet frame.
- **Preamble**: 7 bytes with pattern 10101010 (AA_{Hex}), Followed by 1 byte with pattern 10101011 (AB_{Hex}). Also called "start of frame".
- Preamble allows sender and receiver to synchronise clock rates, as alternating bits form a square wave. Lets receiver know how long 1 bit is.



- **Source and dest MAC address**: If NIC receives frame with matching **destination address** or with **broadcast address**, passes data in frame to network layer protocol, otherwise, NIC discards frame.
- **Data / Payload**: The maximum size is 1500 bytes, which is link MTU mentioned in IP fragmentation. Minimum size is 46 bytes, to ensure that collision will always be detected.
- **CRC**: Cyclic Redundancy Check, For corruption detection.
- **Type**: Higher level protocol used, usually IP. (others e.g. ARP, AppleTalk), permits Ethernet to multiplex network-layer protocols.

Ethernet Topology

- **Bus Topology**: (broadcast LAN) All nodes are connected and can collide with each other.
- **Star Topology**: Switch / Hub in the centre and nodes are connected to that switch. Do not collide with each other.

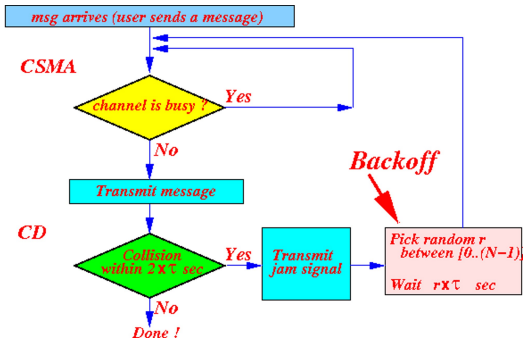


Ethernet Delivery

- **Connectionless**: No handshaking btwn. sender & receiver.
- **Unreliable**: NIC does not send ACK/NAK. Data in dropped frames recovered only if initial sender uses higher layer rdt (e.g. TCP).
- **Ethernet's multiple access protocol**: CSMA/CD with binary (exponential) backoff.

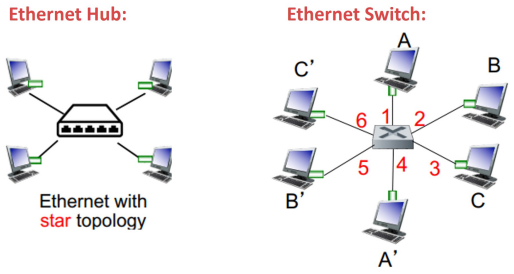
Ethernet CSMA/CD Algorithm

1. NIC receives datagram from network layer, creates frame.
2. If NIC senses that the channel is idle, start frame transmission. Else, wait until idle.
3. If NIC transmits the entire frame without detecting another transmission, NIC is done.
4. If another transmission is detected, NIC aborts and sends a **jam signal**, tells all other nodes that a collision has been detected and NIC will be retransmitting.
5. After aborting, NIC enters exponential (binary) backoff, and repeat from step 2



Ethernet Hub

- **Hub**: Physical-layer, acting on indiv. bits. (not frames)
- When bit arrives from one interface, hub re-creates the bit, boosts energy strength, and transmits the bit onto all the other interfaces.
- **Adv:** Cheap, easy to maintain (modular design of network).
- **Disadv:** Slow, not ideal for larger networks (collisions).



Ethernet Switch

- **Switch** is link-layer device used in LAN that also stores and forwards Ethernet frames. (**Switches act on frames.**)
- **Layer 2 device:** Unlike routers, which is a layer 3 device (i.e. it goes up to the network layer), switches only have 2 layers, i.e. up to link layer.
- **No IP address**: For the reason above, it has no IP address.
- **Transparent to hosts**: Hosts unaware of switch presence.
- **Collision-free**: Each host has dedicated connection to the switch, (separate collision domains). Connection has two channels, i.e. fully duplex and frames sent two-way simultaneously.
- **Store and Forward packet switch**: Switch buffers frames, e.g. if currently forwarding another frame to the outgoing link.

Routers vs. Switches

- **Routers**: Check IP address, Store-and-forward, Compute routes to destination.
- **Switches**: Check MAC address, Store-and-forward, Forward frame to outgoing link or broadcast

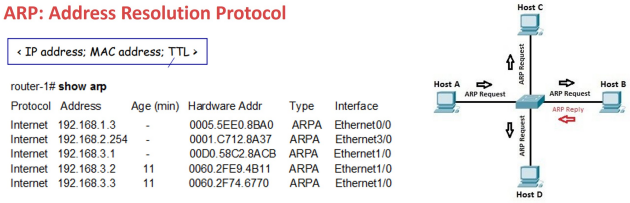
Switch Forwarding Tables

- A switch has multiple interfaces, needs to know which nodes are reachable via which interface.
- Done via **switch forwarding tables**, which have entry format:
  <MAC address of host, interface to reach host, TTL>
- **Self-Learning:** Whenever the switch receives a frame from host A, it will record that interface for A for future frames.
- **Broadcast**: If destination host not found in the switch forwarding table, the switch will broadcast the frame to all outgoing links.

Address Resolution Protocol (ARP)

How to know MAC address of receiving host, knowing its IP address? Use ARP [RFC 826].

- **ARP** provides query mechanism to learn MAC address.
- All nodes have an ARP table containing mappings of IP addresses and MAC addresses of other neighbouring nodes in the same subnet.
- **Plug & Play**: nodes create ARP tables without intervention from network admin.
- **Entry format**: (TTL typically a few minutes)
  <IP address; MAC address; TTL>



ARP: Sending Frame in Same / Another Subnet

1. If A and B in same subnet, A knows B's MAC address from its ARP table:
   (a) A just creates frame with B's MAC address and send.
   (b) Only B will process frame, all other hosts ignore.
2. If A and B in same subnet, A does not know B's address:
   (a) A broadcasts an ARP query packet, containing B's IP address. The destination MAC address is set to FF-FF-FF-FF-FF-FF.
   (b) All other nodes in the same subnet will receive this ARP query packet, but only B will reply it.
   (c) A caches B's IP-to-MAC address mapping in its ARP table (until TTL expires).
3. If A and B in **different subnets** (assume router R directly connecting the two subnets, and A and B):
   (a) A will need to send a frame with R's MAC address but B's IP address as destination.
   (b) R will realise it needs to forward this frame as the IP doesn't match when MAC matches.
   (c) R will forward the datagram to an outgoing link and construct a new frame with B's MAC address.

IP Addresses vs. MAC (Ethernet) Addresses

- **IP address**: 32 bits in length, network-layer address used to move datagrams from source to dest.
- Dynamically assigned, hierarchical (to facilitate routing).
- **MAC address**: 48 bits in length, link-layer address used to move frames over every single link.
- Permanent, to identify the hardware (adapter).

IP address	MAC address
32 bits in length	48 bits in length
network-layer address used to move datagrams from source to dest.	link-layer address used to move frames over every single link.
Dynamically assigned; hierarchical (to facilitate routing)	Permanent, to identify the hardware (adapter)
Analogy: postal address	Analogy: NRIC number

- **ARP** resolves mapping from network layer (IP) address to link layer (MAC) address.

6. Network Security

Properties of Security

- **Confidentiality**: Only sender & receiver should be able to understand message contents.
- **Authentication**: Ability of sender and receiver to confirm the identity of each other.
- **Message Integrity**: Ensuring messages are not altered without detection.
- **Access and Availability**: Services must be accessible and available to users.

Potential Issues

- Alice, Bob and Trudy (Intruder).
- Eavesdrop, Insert or delete messages.
- Impersonation or spoofing (fake) source address
- Hijacking connection
- Denial of service

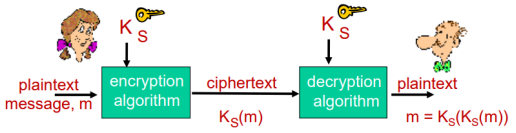
Principles of Cryptography

Terminology

- **Plaintext**: The original message
- **Ciphertext**: The encrypted message
- **Encryption Key**: Used to encrypt messages
- **Ciphertext Only Attack**: Crack the encryption using only the encrypted text. Generally through brute force or frequency analysis.
- **Known Plaintext Attack**: When you have the ciphertext and the corresponding plaintext.
- **Chosen Plaintext Attack**: Presumes that the attacker can obtain ciphertexts for arbitrary plain texts

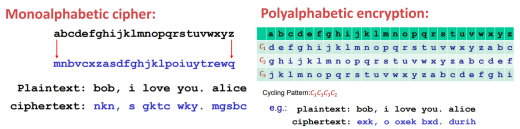
Symmetric Key Cryptograpy

Both sender and receiver share the same key $K_a = K_b = K_s$
Issue lies in initial secure communication of shared key.

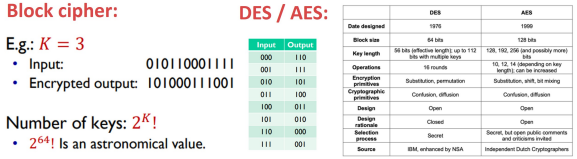


Symmetric Key Cryptograpy

- **Substitution Cipher**: Replace one character for another. Encryption key is the 1-1 mapping of characters used.
- **Caesar's cipher**: Fixed shift of the alphabets. Encryption key is the shift number, 25 possible values, easy to break.
- **Monoalphabetic ciper**: Substitute one letter for another. Mapping from set of 26 letters to itself, 26! possible mappings. Easily broken with statistical analysis of frequent letters.
- **Polyalphabetic encryption**: Use multiple mappings (substitutions ciphers). Define a cyclic pattern and for each new plaintext symbol, use subsequent subtitution pattern in cyclic pattern. Encrption key is the n sub. ciphers and cycling pattern.



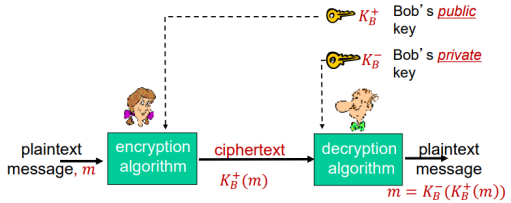
- **Block Ciper**: Message encrypted is processed in blocks of K bits, to encode one block, ciper uses a one-to-one mapping.
- DES: Data Encryption Standard and AES: Advanced Encryption Standard are two symmetric key ciphers.



Asymmetric Key Cryptography

Aka **Public Key Cryptography**. Sender and receiver use different keys $K_a \neq K_b$, and do not share a secret key.

- Sender uses a **public encryption key** known to all, receiver uses a private decryption key known only to receiver. Diffie and Hellman (1978).
- Public-key cryptography has enabled effective encryption in the SSL (Secure Socket Layer) of the HTTPS protocol.



Asymmetric Key Cryptography

- **Requirements** for public key encryption algorithms:
- Need $K_B^+(.)$ and $K_B^-(.)$ such that

$$m = K_B^-(K_B^+(m)) = K_B^+(K_B^-(m))$$

- With pub. key K_B^+, private key K_B^- impossible to compute.

RSA (Rivest, Shamir, Adleman) Algorithm

Basis of Algorithm: Modular Arithmetic

- x **mod** n = remainder of x when divided by n.
- $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
- $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
- $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$
- Thus,

$$(a \bmod n)^d \bmod n = a^d \bmod n$$

- E.g. $a = 14$, $n = 10$, $d = 2$.
  $(14 \bmod 10)^2 \bmod 10 = 16 \bmod 10 = 6$.
  $14^2 \bmod 10 = 6$.
- **message**: bit pattern uniquely represented by an integer number. Ecrypting a message eqv. to encrypting a number. Plaintext $\rightarrow$ ciphertext.

RSA: Creating public/private key pair

1. Choose two large prime numbers p, q.
2. Compute $n = pq$, $\phi(n) = z = (p-1)(q-1)$.
3. Choose e (with $e < n$) that has no common factors with z (e and z are relatively prime.)
4. Choose d such that $ed - 1$ is exactly divisible by z. ($ed \bmod z = 1$).
5. **public key** is $(n, e) = K_B^+$. **private key** is $(n, d) = K_B^-$.

RSA: Encryption, decryption

1. Given (n, e) and (n, d) as computed.
2. To encrypt message m, (note: $m < n$), compute

$$c = m^e \bmod n$$

3. To decrypt received bit pattern c, computer

$$c^d \bmod n$$

4. **key principle:**

$$(m^e \bmod n)^d \bmod n = m$$

RSA property:

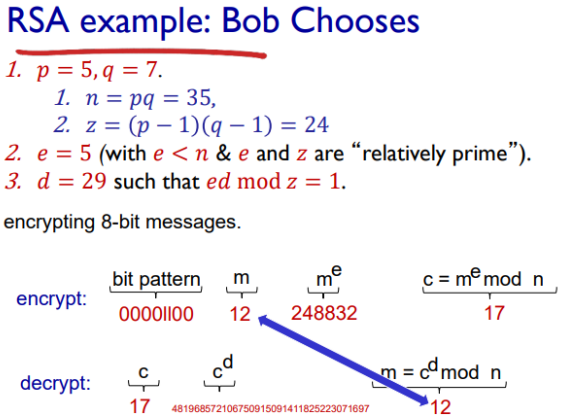
$$m = K_B^-(K_B^+(m)) = K_B^+(K_B^-(m))$$

- Follows from modular arithmetic:

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$
$$= m^{de} \bmod n = (m^d \bmod n)^e \bmod n$$

RSA security, in practice

- With public key $K_B^+ = (n, e)$, determine d. We would need to find the two prime factors of n. without knowing two factors p and q. Hard to factor integer $>$ 512-bit.
- **In practice**: Exponentiation in RSA computationally intensive, DES $>$ 100 times faster than RSA, but needs prior knowledge of symmetric session key K_S.
- Hence, exchange K_S, using public key cryptography, then use K_S to communicate from then.



Message Integrity

Sender, receiver ensure message **not altered without detection**.

Unoptimal error detection as Integrity check

- **Parity, Checksum**: Many-to-one, easy to find another message with same checksum value, designed to detect accidental errors and not attacks.
- **CRC**: Better than checksum, but poor as output is biased to input, minor changes in input produce minor changes in output. Easy to find messages of same CRC.

Password Hashing

Passwords are not stored in plaintext but hashed and stored. To make it hard to find similar plaintexts by simply checking the hashes, a salt i.e. random string of characters is added to the front of the password before being hashed.

Cryptographic Hash Function

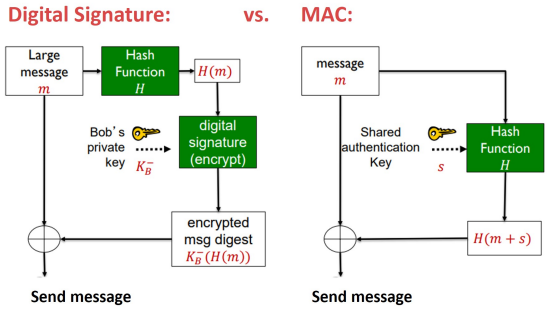
- **Hash Function**: Function $H(.)$ that takes input m, produce **short, fixed-length** message digests (e.g. 128 bits **fingerprint**). Useful for longer inputs.
- **Cryptographic Hash function**: Hash function, computationally infeasible to find any two different messages x and y where $H(x) = H(y)$.
- Computationally infeasible for intruder to substitue one message for another message.
- Small change in input $\rightarrow$ large change in hash output.
- **Common hash functions**:
- MD5: Computes 128-bit message digest in 4-step process. Obsolete now as collisions can be found within a minute.
- SHA-1: US standard. Computes 160-bit message digest, replaced by SHA-2, SHA-3.

Message Authentication Code (MAC)

- **Message Integrity**: Require **Authentication Key** s.
- We cannot just send $(m, H(m))$, as attacker can simply replace both the message and hash.
- Sender, receiver share **authentication key** s**, send** $(m, H(m+s))$**.** s used to generate authentication code from m and compare with received code.

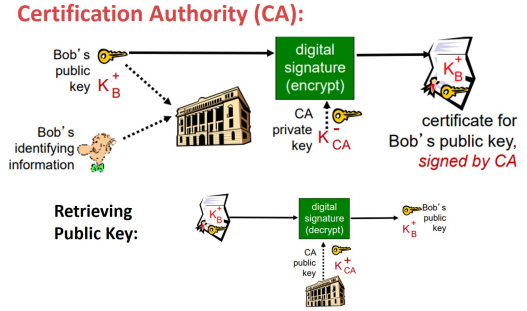
Digital Signatures

- Needs to be **verifiable & unforgeable**.
- **Digital signature**: signed message digest.
- Fixed-length, easy-to-compute digital fingerprint. First hash message, then encrypt hash with private key. Receiver can verify with sender's public key.



Public-key Certification

- Need trustworthy source to store and get list of public keys. Provided by a **Certificate Authority** (CA).
- CAs bind public keys to entities.
- An entity, e.g. a host or a router, registers public key with a CA, with some proof of identity.
- CA creates certificate binding that entity to its key (along with a lot of other information) and certificate is digitally signed by the CA.
- A sender gets receiver's certificate from CA, applies CA's public key to certificate to get receiver's public key.



Network Access & Availability

Ensure services **accessible & available** to users.

Firewalls

- **Firewalls**: combination of hardware and software, isolates organisation's internal network from larger Internet, allowing some packets to pass, blocking others.
- **Prevents denial of service (DoS) attacks:** e.g. SYN flooding, attacker establishes many bogus TCP connections, no resources left for legit. connections
- **Prevents illegal modification/access of internal data.**
- **Allow only authorized access to inside network:** set of authenticated users/hosts.
- UDP: generally firewall either filters all or no UDP.
- **Tradeoff**: between security and degree of communcation with outside outer Internet.
- **Limitation**: IP spoofing, router cannot really know if data comes from claimed source.
- **Three types of Firewalls**: stateless packet filters, stateful packet filters, application gateways.

Stateless (Traditional) Packet Filtering

- Internal network connected to Internet via router firewall. Router **filters packet by packet**, based on policy and firewall settings. Decision to forward/drop packet typically based on:
- source IP address, destination IP address'
- TCP/UDP source and destination port numbers
- ICMP message type
- TCP SYN and ACK bits

Examples:

- Block in/outcoming datagrams with IP protocol field = 17. Result: all in/outgoing UDP flows blocked.
- Block inbound TCP segments with ACK = 0. Result: prevents external clients from making TCP connections with internal clients, but allows internal clients to connect to outside.

Policy	*Firewall Setting*
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Stateful Packet Filtering

Stateful firewalls filter packets based on full context of connection, tracks status of every TCP connection, such asconnection setup, teardown, etc. Uses concept of state table that stores state of legitimate connections.
If inactive connections timeout at firewall, incoming packets will be rejected.

Access Control List (ACL):

ACL: table of rules, applied top to bottom to incoming packets: (action, condition) pairs

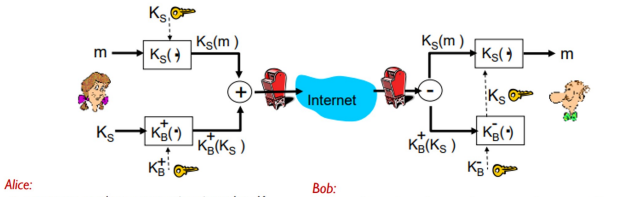
action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all

Application Gateways

- **Application gateways** are more intelligent firewall systems that know details of applications that generate the packets. Also work as proxy servers between client computer and real server.
- Different gateway is needed for each application.
- Slows down network performance due to detailed checking of packets, more expensive than packet filters.

Secure e-mail

Confidentiality:

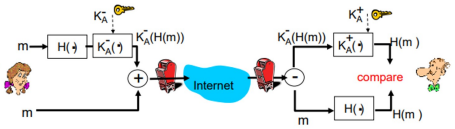


Alice:
- generates random *symmetric* private key, K_S
  - encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
  - sends both $K_S(m)$ and $K_B(K_S)$ to Bob

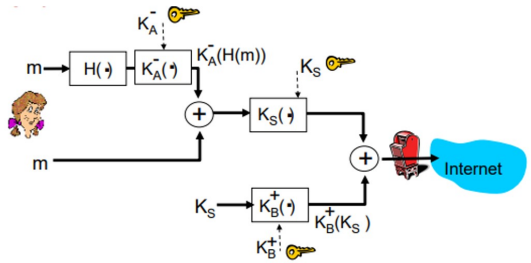
Bob:
- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Authentication:

- Alice digitally signs message
- sends both message (in the clear) and digital signature



Secrecy, Authentication, Integrity:



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

7. Multimedia Networking

Multimedia network application defined as any network application that employs **audio or video**.

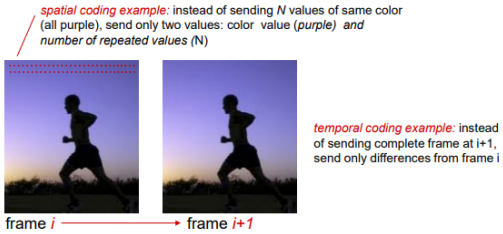
- Videos are often delivered Over-the-Top (OTT), i.e. streaming media services offered directly to viewers via the Internet, bypassing cable, broadcast etc.

Few main application types

- **Streaming** stored audio, video: can begin playout before downloading entire fire, storing / buffering at client.
- **Conversational (two-way live)** voice/video over IP: interactive nature limits delay tolerance, delay more than 400 milliseconds intolerable.
- **Streaming live (one-way live)** audio, video: done with CDNs (Content distrib. network) e.g. live sporting events, higher delay allowable.

Video Multimedia

- **Video** is a sequence of images displayed at constant rate, where a digital image is some array of pixel (luminance, color), each represented by bits.
- **High Bit Rate**: Video streaming consumes a high bandwidth, having a bit rate of more than ten times greater than that of photo or music applications.
- **Video Compression**: Use redundancy within and between images to decrease num. of bits used to encode image.
- Spatial Redundancy (within image): Instead of sending same value N times, send value number of repeats, N.
- Temporal Redundancy: Only send differences between two consecutive images.



- **Constant Bit Rate (CBR)**: video encoding rate fixed. Not responsive to complexity of video, need to set bitrate high to handle complex segments of video. Consistency well suits real-time encoding.
- **Variable Bit Rate (VBR)**: best suited for on-demand video due to longer time allowed to process the data.

Audio Multimedia

- **Sampling**: Sample audio analog signal at fixed rate to real number value to obtain digital signal. Telephone: 8,000 samples/sec, CD: 44,100 samples/sec.
- **Quantisation**: Each sample rounded to one of finite number of values. The number of quantisation values is typically a power of two, e.g. 256.
- **Concatenation & Decoding**: All values represented as bits, and samples are concatenated. To decode, we convert it back to analog signal, which is an approximate of original (quality reduction), as certain sounds lost in sample and encode.

Compression Ratio of media codec: bitrate of uncompressed media stream divided by bitrate of same compressed media stream.

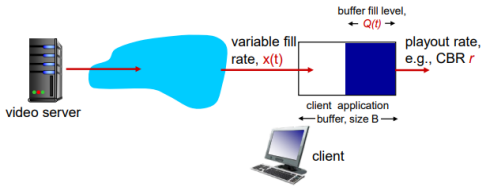
Streaming Stored Video

- **Streaming nature**: Begin playout w/o entire file download.
- **Stored (at server/CDNs**: Can transmit faster than audio / video render rate (imply store / buffer at client)
- **Continuous Playout**: Once client playout begins, playback must match original timing, despite network delays (jitter).
- **Interactivity**: Video pause, fast-forward, jump etc.

Client-side Buffering

Compensates for network-added delay, delay jitter.

- average fill rate: $\bar{x}$, playout rate r.
- **Initial fill of buffer** until client playout at time t_p. Buffer level varies over time as fill rate $x(t)$ varies, but r constant.
- If $\bar{x} < r$, buffer eventually empties, video playout freezes until buffer fills again. If $\bar{x} > r$, buffer will not empty if initial playout delay enough to absorb variability in $x(t)$.
- If buffer filled, fill rate will be capped at playout rate.
- **Client side buffer absorbs variation in delay & bandwidth**, for long as buffer is not drained.
- **Initial playout delay tradeoff** between likelihood of buffer starvation and initial delay.

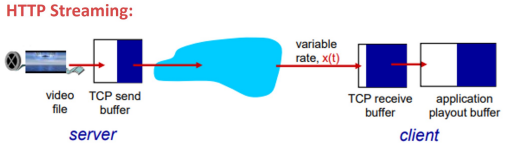


UDP Streaming

- **Push-based Streaming**: Server sends at rate matching client's consumption rate. As UDP has no congestion control mechanism, server transmits w/o rate-control restrictions of TCP.
- E.g. video consumption rate = 2Mbps, each UDP packet carries 8,000 bits of video. Server needs to transmit one UDP packet into socket every $\frac{(8000bits)}{(2Mbps)} = 4msec$.
- **Short playout delay**: (2-5 seconds) to remove network jitter. Small buffer, susceptible to bandwidth changes, video may freeze or skip frames.
- **Error recovery**: Application level, if time permits.
- **RTP**: Video chunks encapsulated using Real-Time Transport Protocol, control connection maintained separatedly using RTSP (RT stream protocol), used for establishing and controlling media sessions, and allowing issuing of commands e.g. play, record, pause.
- **Drawback**: UDP may not go through firewalls, users behind firewall cannot receive. Need for separate media control server (RTSP), increases cost, complexity.

HTTP Streaming

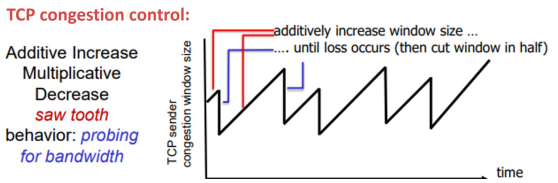
- **Pull-based Streaming**: multimedia file retrieved via HTTP GET, sent at max possible rate under TCP.
- **Prefetching:** Cccurs naturally as TCP's congestion avoidance tries to use all available bandwidth. If bandwidth is greater than consumption rate, prefertching occurs.
- **Sending Flow**:Video file chunks → server send buffer → client TCP receive buffer → client TCP application buffer (application decompresses and displays).



- If application buffer is full, send rate reduced to consumption rate. If video paused, sending and buffering continues till buffer full. Then sending will block until video resumed.
- Repositioning: If user skips forward in video, all buffered frames wasted. Medium sized client application buffer reduce wastage.

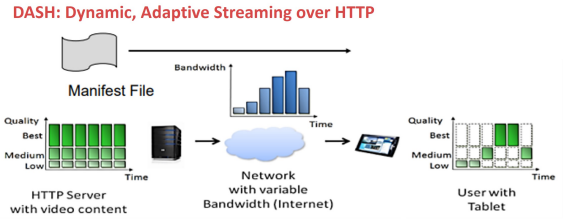
HTTP Streaming cont.

- **Advantage**: HTTP/TCP passes more easily through firewalls, network infrastructure (CDNs, routers) more fine tuned for HTTP/TCP.
- **Drawback**: Fill rate fluctuates due to TCP congestion control and retransmissions (in-order delivery). Also, larger playout delay than push-based UDP.



DASH (Dynamic, Adaptive Streaming, HTTP)

- Video-on-Demand (VoD) video streaming increasingly uses HTTP streaming, but simple HTTP streaming GETs (whole) video file from HTTP server. $\rightarrow$ wasteful, needs large client buffer.
- **Issue**: All clients receive same encoding (resolution) of video, despite variation in device/network bandwidth.
- **DASH: Dynamic, Adaptive** Streaming over HTTP.



DASH Server, Client

- **Various Encodings**: Video encoded into diffferent versions w. differing bit rates and quality levels.
- **Manifest and Byte Range**: HTTP server stores a .m3u8/.mpd manifest file, providing URL for each version, along with bit rate. Client first requests for the manifest file to know the various versions.
- **Pull-based Streaming**: client periodically measures bandwidth, dynamically request sustainable versions of chunks of video segments of 2-10 seconds in length, that current bandwidth can support.
- **Adaptive Bitrate Algorithm: (ABR)** Client runs algorithm to decide which quality next chunk should be.
- **Streamlets or Byte Range**: Thereafter, client either GET request for specific streamlet, if server had split the video into streamlets during pre-processing, or use GET requests with specified byte range in header to get the right chunk.
- **Advantage**: Works with Web Caching: DASH works well with existing web caching infrastructure of ISPs and Content Delivery Networks (CDN). Server is simple, no firewall problems.
- **Disadvantage**: DASH based on media segment transmissions, (2-10 sec in length typically). By buffering a few segments at client side, DASH does not provide low latency for interactive, two-way applications (e.g., video conferencing).

Voice-over-IP

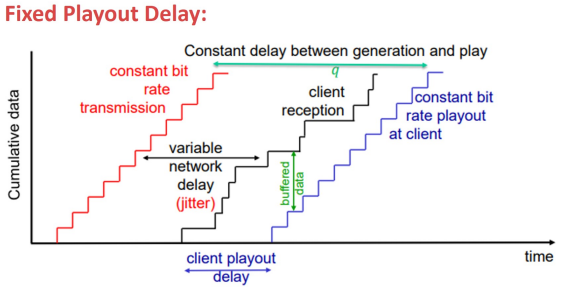
- **VoIP end-end-delay requirement**: needed to maintain 'conversational' aspect. < 150 msecs not perceived by human listener, delays > 400 msecs impair interactivity, in between acceptable but not ideal.
- **Talk Spurts**: Periods of silence during conversation, nothing is sent. When speaking: $8,000$ samples/sec, of 8 bits each, 256 quantization levels. Data $= 64kbps$. $20msec$ chunks, hence 160 bytes per chunk.
- **App-layer header**: Header added to each chunk, then encapsulated into (generally) UDP segment. Send segment into socket every 20 msec during talk spurt.
- **Challenge:** Internet (IP layer) best effort service, no upper bound on delay or packet loss.
- **Packet Loss**: UDP segments containing a chunk and special header may be lost in transit. TCP unusable as retransmission mechanisms delay unacceptable.
- **Loss tolerance**: depending on voice encoding, loss concealment, loss rates between 1% - 10% tolerable. Receivers generally disregard packets beyond certain delay threshold (delay loss).
- **Delay Jitter**: Varying network delay (end-to-end time delay over network).

Removing Jitter for Audio

- **Timestamp**: Chunk has timestamp t, at which chunk is generated.
- **Playout Delay**: Delay needs to be long enough so most packets received before scheduled playout times.

VoIP: Fixed Playout Delay

- Receiver attempts to play out each chunk at exactly q msecs after chunk generated. (chunk timestamped with time t will be played at time $t + q$.)
- Packets arriving after $t + q$ discarded.
- **Tradeoff**: large q, less packet loss, big q, more interactive experience
- No value of q can guarantee optimal performance, eventually will have packet loss or waste playout time.



VoIP: Variable Playout Delay

- **Goal**: low playout delay, low late loss rate, through **adaptive playout delay adjustment**.
- Estimate network delay, adjust playout delay at beginning of each talk spurt. (adjusting silent periods).
- Silent periods compressed and elongated, chunks still played out every 20msec during talk spurt.
- Use **Exponentially weighted moving average (EWMA)**:

Adaptively Estimate packet delay (EWMA):

Delay Est: $d_i = (1-\alpha)d_{i-1} + \alpha\,(r_i - t_i)$

- d_i: delay estimate after ith packet
- α: small constant, e.g. 0.1
- $r_i - t_i$: time received - time sent (timestamp); measured delay of ith packet

Avg. Std. Dev.:

$v_i = (1-\beta)v_{i-1} + \beta\,|r_i - t_i - d_i|$

- v_i: estimate of average deviation of delay after ith packet

- Estimates, d_i *and* v_i calculated for every received packet, but used only at start of talk spurt
  - for first packet in talk spurt, playout time is:

$$playout\text{-}time_i = t_i + d_i + 4v_i$$

  - remaining packets in talk spurt are played out periodically

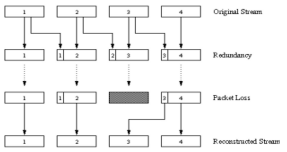
VoIP: Packet Loss Recovery

Recovering from Packet Loss given small tolerable delay between transmission and playout.

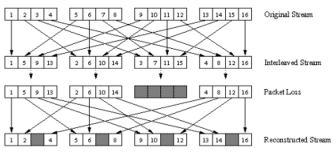
Forward Error Correction (FEC)

- **Redundancy**: send enough bits to allow recovery without retransmission.
- **Simple FEC**: Every group of n chunks, create redundant chunk by XOR-ing n chunks. Send these n + 1 chunks. Increase bandwidth by a factor of $1/n$, allows reconstruction of up to 1 lost chunk from $n + 1$ chunks.
- Increases playout delay, wait for entire group of packets before playout.
- **Lower-Resolution Audio Stream**: "piggyback lower quality stream". For n^{th} chunk, append lower quality ver. of it to $n + 1^{th}$ chunk.
- E.g., nominal stream PCM encoding at 64kbps, redundant stream is GSM at 13kbps.
- Non-consecutive loss of packets, receiver can conceal loss.
- **generalization**: extend further, append more low bit-rate chunks, ($n - 1 \& n - 2$ low bit rate chunk to n^{th} chunk).
- **Interleaving**: Divide audio chunks into smaller units, each packet contains small units from different chunks.
- If packet lost, still have most of every chunk, concealable by packet repetition or interpolation.
- No redundancy overhead, but increases playout delay as reordering needed on both sides.

Lower – Resolution Audio Stream:



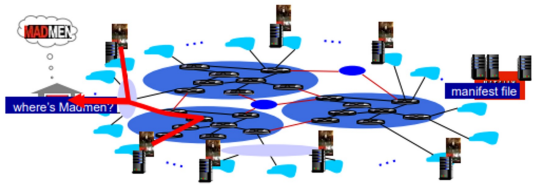
Interleaving:



Content Distribution Networks

- **Challenge**: Stream content (selected from millions of videos) to multitude of simultaneous users.
- Using single, large mega server: single point of failure, point of network congestion, long path to distant clients, multiple copies of video sent over outgoing link, solution does not scale.
- **Multiple Sites**: store/serve multiple copies of videos at multiple geographically distributed sites (CDN).
- **enter deep**: push CDN servers deep into many access networks, Usually at ISP, close to users. (e.g. Akamai, 1700+ locations)
- **bring home**: smaller number (10's) of larger clusters in IXPs (I. exchange point) near (but not within) access networks. (e.g. used by Limelight)
- Hence, CDN stores copies of content at **CDN nodes**, client requests content, service provide returns manifest, and using manifest client retrieves content at highest supportable rate.
- Client may choose different rate or copy (from different CDN node) automatically if network path congested.

Content Distribution Network (CDN):



Commands

`traceroute`
Firstly, it tells you that it's tracing the route to some domain, tells you the IP address of that domain, and what the maximum number of hops will be before it times out. Next it gives information about each router it passes through on the way to its destination. Each of the 3 columns are a response from that router, and how long it took (each hop is tested 3 times).

- 1 is the internet gateway on the network this traceroute was done from.
- The next few routers are likely part of the ISP that the origin computer is connected to.
- The next few are likely global gateways.
- We will then see that we will head towards the destination's local ISP.
- Finally, we will get a router on the network that the domain is hosted on, and lastly the host that the domain is hosted on directly.

`nslookup`
This is usually used to find the IP address that corresponds to a host, or the domain name that corresponds to an IP address (a process called "Reverse DNS Lookup"), by retrieving the relevant address information directly from the DNS cache of name servers.

- The initial server and address are those of our DNS server.
- An authoritative answer comes from a nameserver that is considered authoritative for the domain which it's returning a record for (one of the nameservers in the list for the domain you did a lookup on).
- A non-authoritative answer comes from anywhere else (a nameserver not in the list for the domain you did a lookup on).

`ping`
It sends packets of data to a specific IP address on a network, and then lets you know how long it took to transmit that data and get a response. It uses the echo request and echo reply messages within ICMP. When a `ping` command is issued, an echo request packet is sent to the address specified. When the remote host receives the echo request, it responds with an echo reply packet.

By default, the ping command sends several echo requests, typically four or five. The result of each echo request is displayed, showing whether the request received a successful response, how many bytes were received in response, the Time to Live (TTL), and how long the response took to receive, along with statistics about packet loss and round trip times.

`dig`
Domain Information Groper (dig) allows us to query DNS servers.

- Let's say we do `dig google.com`.
- Lines beginning with ; are comments not part of the information.
- The first line tell us the version of the `dig` command.
- Next, it shows the header of the response it received from the DNS server.
- Next comes the question section, which simply tells us the query, which in this case is a query for the "A" record of google.com. The IN means this is an Internet lookup (in the Internet class).
- The answer section tells us that google.com has quite a few IP addresses.
- Lastly there are some stats about the query. You can turn off these stats using the +nostats option.

`ifconfig` or `ipconfig`
`ipconfig` stands for Internet Protocol Configuration. This command is used to view all the current TCP/IP network configurations values of the computer, mainly used in Microsoft Windows operating system.

- `ipconfig/registerdns`: Refreshes all DHCP leases and reregisters the DNS names.
- `ipconfig/displaydns`: Displays the information that is stored in the DNS Resolver cache.
- `ipconfig/renew`: Requests a new IP address.
- `ipconfig/flushdns`: Clears the DNS Resolver cache containing previous DNS information.

The `ifconfig` command is mainly used in a Unix-like operating system.

- `ifconfig [interface name]`: This command gives information about the network configuration of the specified interface only.
- `ifconfig {a`: This command gives the network configuration information about all the connected interfaces, whether they are active or not.

`telnet`
Telnet is a computer protocol that was built for interacting with remote computers. It is one of the simplest ways to check connectivity on certain ports. The format is as such: `telnet [domain name or ip] [port]`. If the connection succeeds, a blank screen will show up, meaning that the computer port is open. A failed connection will be accompanied by an error message. It can indicate either a closed port or the fact that the indicated remote server is not listening on the provided port.

`arp`
Displays and modifies entries in the ARP cache, which contains one or more tables that are used to store IP addresses and their resolved Ethernet or Token Ring physical addresses. There is a **separate table** for each Ethernet or Token Ring network adapter installed on your computer. Used without parameters, `arp` displays help.

- `arp -a`: Displays current ARP cache tables for all interfaces.
- `arp -d hostname`: Deletes an entry with the specified `hostname`. To delete all entries, use `arp -d *`, i.e. flush your ARP cache.
- `arp -s hostname ether_addr`: Adds a static entry to the ARP cache that resolves the IP address `hostname` to the physical address `ether_addr`.

`md5sum`
Prints a 32-character (128-bit) checksum of the given file, using the MD5 algorithm.

`curl`
Transfers data to or from a server, using any of the supported protocols (HTTP, FTP, IMAP, POP3, SCP, SFTP, SMTP, TFTP, TELNET, LDAP or FILE). The most basic use is typing the command followed by the URL.