1 Computer Networks and the Internet

1.1 What is the Internet?

- Internet is a computer network that connects billions of computing devices.
- Devices connected to the internet are called **hosts** or **end systems**.
  - Hosts can be further split into **clients** and **servers**.
- Hosts are connected through **communication links** (e.g. fiber, copper, radio, satellite) or **packet switches** (e.g. routers and switches)

1.2 Network Edge

1.2.1 Access Network

- Network that physically connects an end system to the first router
- They are what end-users connect to if they want to access the Internet

1.3 Network Core

Network Core is the mesh of interconnected routers that links the Internet's end systems (Runs low level protocols)

1.3.1 Packet Switching

Data sent through the net in discrete chunks where each **packet** is transmitted at full link capacity (transmission rate / bandwidth)

- **Transmission rate (R)** = # of bits transmitted into the link per second
- **Transmission delay** = Time needed to transmit L-bit packet into link = $\frac{L(bits)}{R(bits/sec)}$
- **Store-and-forward**: entire packet must arrive at a router before it can be transmitted to the next link.
- **Queueing and loss:**
  - packets enters a queue if arrival rate > transmission rate
  - packets can be dropped if buffer (queue) fills up
- **Forwarding Tables and Routing Protocols:** Routers use these to determine the link it should forward the packet to.
- **Benefits:** Message segmentation **reduces delay**, **allow for retransmission** of smaller parts in case of corruption, and are **easier to buffer + queue** behind (more fair for other packets).
- **Drawbacks**: Packet needs to be re-arranged at destination and there is more overhead

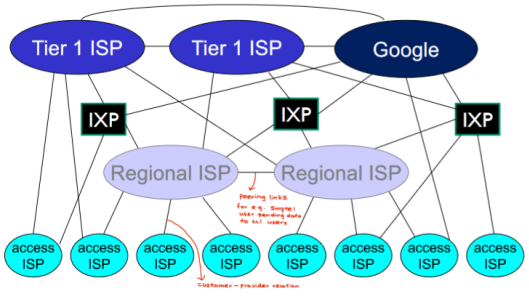
1.3.2 Circuit switching

Dedicated circuit per call

- end-end resources reserved for call between source & dest
- call setup required
- circuit-like (**guaranteed**) performance
- circuit segment idle if not used (no sharing)
- commonly used in telephone networks

1.3.3 Internet Structure

- End systems access the Internet through **access Internet Service Providers (ISPs)**
- Access ISPs must be interconnected so that any 2 hosts can send packet to each other
- Results in a network of network:
  - **Regional ISP:** ISPs that access nets in the region connect to.
  - **Tier-1 commercial ISP:** Provides national and international coverage. Regional ISPs connect to Tier-1 ISPs (e.g. Sprint, AT&T)
  - **Internet Exchange Point (IXP):** Allows multiple Tier-1 ISPs to interconnect with each other
    - Peering: bilateral. more expensive but much faster
    - IXP: multilateral, cheaper but slower
  - **Content-Provider Networks:** Private network own by company that connects their own data centres to the Internet, often bypassing tier-1 and regional ISP (e.g. Google)
- If we were to connect all N devices in a network, we would need $N \times (N-1)/2$ links



1.4 Delay, Loss and Throughput in Networks

1.4.1 Packet Loss

As the queue for router has finite capacity, a router will **drop** a packet if the queue is full and the packet will be lost. It **can be retransmitted** by previous node, source end system or not at all.

1.4.2 4 Sources of Packet Delay

- **Nodal Processing (d_{proc}):** check bit errors, determine output link, typically in microseconds or less
- **Queueing (d_{queue}):** time waiting in output link for transmission, depends on congestion level of router. Typically in microseconds to milliseconds.
- **Transmission ($d_{trans} = \frac{L}{R}$):** L = packet length (bits), R = link bandwidth (bps), time taken to push data onto link
- **Propagation ($d_{prop} = \frac{d}{s}$):** d = length of physical link, s = propagation speed in medium ($2 \times 10^8 m/sec$), time taken to travel through physical link
- For $N-1$ routers between source host and destination host, we have:

$$d_{end-to-end} = N(= d_{proc} + d_{queue} + d_{trans} + d_{prop})$$

1.4.3 Throughput

- Rate (bits/unit time) at which bits are transferred between sender/receiver
  - **average:** rate over longer period of time (note that size of file does not matter in this case!)
  - **instantaneous:** rate at given point in time

1.5 Protocol Layers and Service Models

Protocols: Define format, order of messages sent and received among network entities, and actions taken on message transmission, receipt

To provide structure to the design of network protocols, network designers organise protocols in layers. We are interested in the services that a layer offers to the **layer above**, i.e. the **service model** of a layer.

1.5.1 Why layer?

- explicit structure **allows identification**
- modularization **eases maintenance**, updating of system
  - change in implementation of a layer's service is transparent to the rest of the system, i.e. changing one layer **does not** affect the rest of the layers

1.5.2 5 Layers

- **Application**: Where network applications and their application-layer protocols reside. A packet of information at this layer is called a **message**. Examples: FTP, SMTP, HTTP
- **Transport**: Transports application-layer messages between application endpoints. A transport-layer packet is called a **segment**. Examples: TCP, UDP
- **Network**: routing of **datagrams** from source to destination. Examples: IP, routing protocols
- **Link**: Data transfer of **frames** between neighbouring network elements (nodes). Examples: Ethernet, 802.11 (WiFi), PPP
- **Physical**: bits "on the wire". Link dependent and transmission medium-dependent

2 Application Layer

Goal: Write programs that run on different end systems, communicate over the network (e.g. web server software ↔ browser software) Note that network-core devices (routers) have **no application or even transport layer** which allows for rapid application development.

2.1 Principles of Network Applications

2.1.1 Client-Server Architecture

- **Server**: always-on, waits for incoming requests, provides requested service to client, data centres for scaling, has **permanent IP address**
- **Client**: initiates contact with server, typically requests service from server, may have **dynamic IP addresses**, do not communicate with each other, may be intermittently connected

2.1.2 Peer-to-Peer (P2P)

- No always-on server
- Arbitrary **end systems directly communicate**
- Peers request and provide services to other peers
- **Self-scalability**: new peers bring new service capacity, as well as new service demands
- Peers are intermittently connected and change IP addresses (lead to complex management)

2.2 Process Communication

2.2.1 Sockets

- A **software interface** (between application & transport layer) that a process uses to send messages into, and receive messages from the network. Uses **IP address and a port number** as identifier.
- **IP Address:** 32-bit Integer that uniquely identifies a host
- **Port Number:** 16-bit integer that identify what services a process is using (e.g. HTTP Server: 80, mail server: 25, DNS: 53)

2.2.2 Transport Services

- **Data integrity**: 100% reliable (e.g. file transfers, web transactions) vs tolerant to data loss (e.g. audio, video stream, multimedia)
- **Timing**: some apps require low delay (latency) to be "effective" (e.g. interactive games, zoom)
- **Throughput:** require minimum amount of throughput to be "effective" (e.g. gaming)
- **Security:** encryption scheme, data integrity

application	data loss	throughput	time sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	
interactive games	loss-tolerant	few kbps up	yes, few secs
text messaging	no loss	elastic	yes, 100's msec
			yes and no

2.2.3 Transport-Layer Protocols

TCP	UDP
Reliable data transfer	**Unreliable** data transfer
Flow control: sender won't overwhelm receiver	**No flow control**
Congestion control: throttle sender when network is overloaded	**No congestion control**
Connection-oriented: setup required between client and server	**No setup** needed
Does not provide: timing, minimum throughput guarantee, security	**Does not provide**: timing, throughput guarantee, security

- Why UDP?
  - UDP has less overhead and complexity and has its uses in multimedia where data integrity is not as important and delay might be shorter

application	application layer protocol	underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (e.g., YouTube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	TCP or UDP

2.2.4 Definition of App-layer Protocols

- **Types of Messages exchanged,** e.g. request, response
- **Message syntax**, e.g. message fields and how they are delineated
- **Message semantics**: meaning of information in fields
- **Rules** for when and how application send and respond to messages

There are also open protocols defined in RFCs (Request for Comments) such as HTTP (allows for interoperability) and some proprietary protocols (e.g. Skype)

2.3 Web and HTTP

2.3.1 Web Page

- consists of a base HTML file and several referenced objects
- objects can be HTML file, JPEG image, Java applet, audio file etc.
- object is addressable by a URL:
  - **Hostname**: www.comp.nus.edu.sg
  - **Path name**: / ~cs2105/img/lect1.pdf

2.3.2 HTTP

- **HyperText Transfer Protocol** is the Web's **application layer** protocol
- **Client/server model**: Client is the browser that requets, receives and "displays" web objects whereas the server is a Web Server that sends objects in response to requests
- **Over TCP** (Reliable!)
- **Stateless**: server maintains no information about past client requests

2.3.3 Persistent HTTP

- **Multiple objects** can be sent **over single TCP** connection
- TCP connection is left open after sending response
- **Persistent with pipelining**: client may send requests as soon as it encounters a referenced object – as little as 1RTT for all referenced objects.

2.3.4 Non-Persistent HTTP

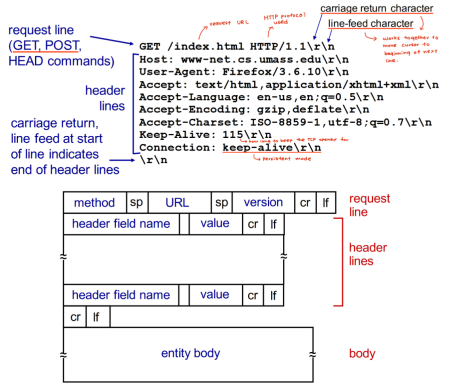
- At most 1 object sent over a TCP connection (TCP connection is closed after each request)
- downloading multiple objects requires multiple connections
- Requires 2 RTTs per object (1 to initiate TCP connection, 1 to request the file), and 2 RTTs get the initial HTML file
- Response time = ($2 \times RTT$+ file transmission time) / object
- incurs **overhead** for each TCP connection → solution is to **open parallel TCP connections**

* **RTT**: time for a small packet to travel from client to server and back (does not include transmission delay!)

$$RTT = d_{proc} + d_{queue} + d_{prop}$$

2.3.5 HTTP Request Format

* Note that it is not possible to find IP address of Host in the HTTP header



2.3.6 HTTP Method Types

- **HTTP/1.0** (Default to Non-Persistent):
  - `GET`: Gets an object
  - `POST`: Posts form data
  - `HEAD`: asks server to leave requested object out of response (mostly for debugging)
- **HTTP/1.1** (Default to Persistent):
  - `GET, POST, HEAD`
  - `PUT`: uploads file in entity body to path specified in URL
  - `DELETE`: deletes file specified in URL

2.3.7 HTTP Response Status 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:)
- `400 Bad Request`: request msg not understood by server
- `404 Not Found`: requested document not found on this server
- `500 HTTP Version Not Supported`

2.4 Cookies

As HTTP is stateless, we need cookies to help carry states. Cookies has four components:

1. cookie header line of HTTP response message
2. cookie header line in next HTTP request message
3. cookie file is stored on user's host and is managed by user's browser
4. back-end database at website

Cookies are typically used for: authorization, shopping carts, recommendations, user session state (Web email). The state is maintained at sender/receiver over **multiple transactions**

2.4.1 Conditional GET

Web heavily **utilises caching to reduce response time** by caching objects in a proxy server (typically much closer to clients). If a request object is in the proxy server, cache returns the object else it will request object from origin server. (lowers link utilisation)

Since there is a cache, we must handle and prevent transferring stale objects. This is done through a `If-Modified-Since: <data>` header line which returns a `200 OK` and returns new object if data has been modified, else `304 Not Modified` and no object if data is not modified.

2.5 DNS: The Internet's Directory Service

The **Domain Name System** helps to map hostnames (e.g. google.com) to their IP address and vice versa so we as humans do not have to remember individual IP addresses for websites. (Default port 53)

- **Distributed, Hierarchical Database** in the Application Layer
- **Root Server**: answers requests for records in the root zone by returning a list of the authoritative name servers for the appropriate TLD
- **DNS Caching**:
  - cache entries timeout after some time (**TTL** - Time to live)
  - TLD servers are typically cached in local name servers → root name servers are often not visited
  - cached-entries **may be out of date** → if a host changes IP address, it may not be known until TTL expires
- Runs over **UDP**

2.5.1 Local DNS name server

- when host makes DNS query, first sent to the Local DNS server
- contains local cache of recent name-to-address translation pairs (but **might be out of date**)
- acts as proxy to forward query into hierarchy
- **does not strictly belong to hierarchy**

2.5.2 Root name server

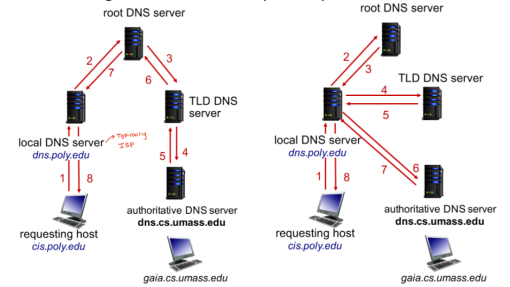
- contacted by local name server that cannot resolve name
- provides IP address of TLD (Top Level Domain e.g. .com, .gov, .sg) servers

2.5.3 Authoritative servers

- organization's owned DNS servers that provides authoritative hostname to IP mappings for organization's named hosts
- maintained either by organization or service providers

2.5.4 DNS Name Resolution

- **Iterative query**: Local DNS server makes DNS requests one by one in the hierarchy
- **Recursive query** (rarely used): each server in the hierarchy asks one server higher in the hierarchy (heavy load on root DNS server)



(a) Recursive (b) Iterative

3 Socket Programming

Socket: the software interface between app processes and transport layer. Allows for communication between processes over the Internet.

Relies on:

1. **Port Number**: 16-bit unsigned integer (0-1023 are reserved)
2. **IP Address**: 32-bit unsigned integer

3.1 UDP (User Datagram Protocol)

- **unreliable** datagram, **connectionless**
- no handshaking before sending data
- sender (client) explicitly attaches destination IP address and port number to each packet
- receiver (server) extracts sender IP address and port number from the received packet
- transmitted data **may be lost** or **received out-of-order**

* For clients, port number can be random whereas server port number must be assigned

3.2 TCP (Transmission Control Protocol)

- reliable **byte stream**, connection-oriented
- When client creates socket, client TCP establishes a connection to server TCP. (Server must have "welcome socket")
- When contacted by client, server TCP creates a new socket (with same port number) for server process to communicate with that client
- Allows server to talk with multiple clients individually.
- Communicates as if there is a pipe between 2 processes, sending process doesn't need to attach a destination IP address and port number in each sending attempt.

UDP	TCP
Server uses one socket to serve all clients (n clients -> 1 socket)	Server creates a new socket for each client (n clients -> n+1 sockets)
No connection is established before sending data	Client establishes connection to server
Sender explicitly attaches destination IP address + port#	Server uses connection to identify client
Unreliable datagram: Data may be lost, received out-of-order	Reliable stream pipe: data guaranteed to be received in order

4 UDP, Reliable Data Transfer

4.1 Transport-layer Services and Protocols

- provides **logical communication** (allows processes to feel like they are directly connected to each other) between app processes running on different hosts
- **Sender**: Breaks app messages into segments, passes them to network layer
- **Receiver**: Reassembles segments into message, passes it to app layer
- Primarily uses 2 protocols:
  1. TCP (reliable, more overhead, slower)
  2. UDP (unreliable, less overhead, faster)

4.2 Transport vs Network layer

- Transport layer: logical communication between processes
  - relies on and enhances network layer services
- network layer: logical communications between hosts/interfaces
  - unreliable "best-effort" (relies on TCP for reliability)

4.3 Connectionless Transport: UDP

- UDP adds very little on top of IP
  - **Multiplexing** at sender (aggregate packets that needs to be transported out)
  - **Demultiplexing** at receiver (determines which packets are for which processes)
  - **Checksum**
- UDP transmission is unreliable, often used by streaming multimedia app (loss tolerant & rate sensitive apps)

4.3.1 Why UDP?

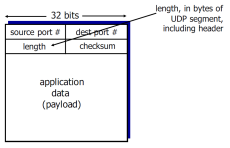
No connection establishment (less delay), simple (no connection state at sender, receiver), small header size, no congestion control (UDP can send data without worrying about data loss)

4.3.2 Connectionless De-multiplexing

When UDP receiver receives a UDP segment:

- Check destination port number in segment, and direct that segment to the socket with that port number.
- basically same dest. port # → same socket at destination

4.3.3 UDP Header



UDP segment format

* Note that the minimum length of UDP segment is 8 bytes which is just the header itself

4.3.4 UDP Checksum

1. Treat UDP segment as sequence of 16-bit integers
2. Apply binary addition
3. Add carry to result (if any)
4. Compute 1's complement to get the UDP checksum

Result of adding checksum and the original sum should give all 1's. Also note that UDP checksum is done in complement with other checksums (in network layer) to ensure reliable data transfer (passing UDP checksum alone is not guaranteed that no bit error occurred)

4.4 Principles of Reliable Data Transport

To build a reliable transport layer protocol, we have to handle:

- Packet corruption
- Packet loss
- Packet reordering
- Packet delay (arbitrarily long)

4.4.1 RDT 1.0

Assumption: Underlying channel is perfectly reliable (no bit errors and no loss of packets).

1. (Sender) Make packet and send packet to receiver using network layer.
2. (Receiver) Extract packet from network layer and deliver data to application.

4.4.2 RDT2.0

Assumption: Underlying channel may flip bits i.e. corruption. We use the stop and wait protocol (server send one packet at a time then wait for recv response).

1. (Sender) Make packet and send packet to receiver.
2. (Receiver) Check if packet is corrupt. If not corrupt, deliver data to application and send ACK to sender, else just send NAK to sender.
3. (Sender) If received NAK, resend the same packet, else go to step 1 for next packet.

Problems: If ACK or NACK are corrupted, there is no guaranteed way to recover. If we simply resend the packet, the receiver will not know it's a duplicate.

4.4.3 RDT2.1

Assumption: Same as rdt 2.0 - corruption. In addition to what is covered in rdt 2.0, we now add a sequence number to the packet. This number alternates between 1 and 0. Duplicates are detected using sequence number.

1. (Sender) Sends pkt0.
2. (Receiver) Sends ACK or NAK.
3. (Sender) If NAK or corrupted message received, retransmit message. Else send pkt1.
4. (Receiver) If duplicate pkt0 is received, drop it and reply ACK. Else continue as per usual.

4.4.4 RDT2.2

Assumption: Same as rdt 2.0 and 2.1 - corruption. In addition to what is covered in rdt 2.1, we now explicitly include the sequence number of the packet being acknowledged, removing the need for a NAK. From the sender's perspective, we basically resend current packet if a duplicate ACK is received.

1. (Sender) Sends pkt0.
2. (Receiver) Sends ack1 if corrupted, else ack0.
3. (Sender) So long as ack# is different from packet seq# sent earlier or corrupted ACK, keep resending that packet.
4. (Receiver) If duplicate pkt0 is received, drop it and reply ack. Else continue as per usual.

4.4.5 RDT3.0

Assumption: Corruption, packet loss and packet delays, but no reordering (i.e. the order sent is the order received). We need to add sender **timeouts** to rdt 2.2 to handle packet loss and delays. One important difference is that duplicate ACKs are ignored. To detect packet loss, the timer is used.

1. (Sender) Sends pkt0.
2. (Receiver) If received and pkt# correct, send ack0. Else if received and is corrupted, send ack1.
3. (Sender) If no response is heard by a certain time (either due to delay or loss on either side), resend pkt0. Else if ack1 received, also resend pkt0. Only if ack0 is received do we move on to next packet. (Corrupt ACKs ignored)
4. (Receiver) Same as step 2, except if packet 1 has been received before and it's a duplicate, we also send ACK1.
5. (Sender) If repeated ack is received, it will ignore the ack (i.e. it will not send pkt again)

4.4.6 Performace of RDT3.0

Performance of stop-and-wait protocol is very bad!

e.g. if 1 Gbps link, $D_{prop} = 15ms$, 8000 bit packet:

$$D_{trans} = \frac{L}{R} = \frac{8000 \text{ bits}}{10^9 \text{ bits / s}} = 8 \text{ microseconds}$$

$$U_{sender} = \frac{L/R}{RTT + L/R} = \frac{.008}{30.008} = 0.0027\%$$

if RTT=30 msec, 1KB pkt every 30 msec: 33kB/sec throughput over 1 Gbps link → network protocol limits resources!

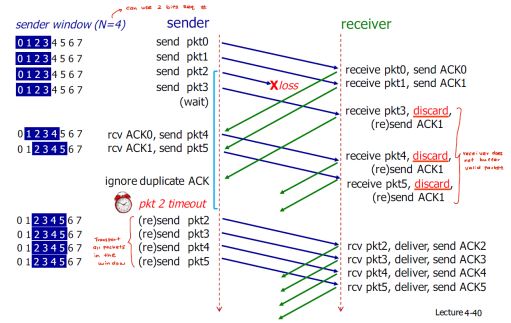
4.5 Pipelining

Generally, N-packet pipe-lining will increase utilization by a factor of N. To allow for pipelining, we have to: increase range of sequence number and buffer at sender and/or receiver

4.5.1 Go-back-N

Pros: only 1 timer needed for oldest in-flight pkt, only need to remember `ExpectedSeqNum`, cumulativeACK → if ACK lost, just take highest ACK# and carry on from there

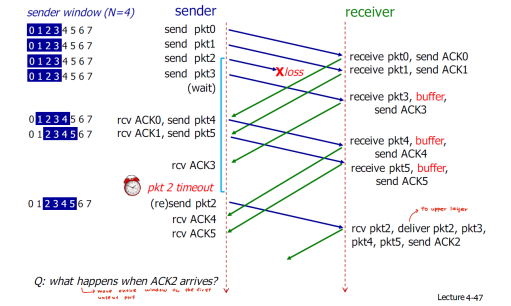
Cons: If one packet loss halfway and the rest is received, everything that is received will be discarded → not efficient use of resources



4.5.2 Selective Repeat

Pros: Only retransmits unACKed pkts and buffer out-of-order pkts → more efficient use of resources, no need to retransmit so many pkt

Cons: much more complex since need to maintain multiple timers for all the packets in the window



5 Connection-oriented Transport: TCP

- **Point-to-point**: 1 sender, 1 receiver
- **Connection-oriented**: handshake before data exchange
- **Full duplex data**: bi-directional data flow in same connection
- **Reliable, in-order byte stream**: no "message boundaries" (i.e. format of data sent does not matter → whether its a .jpg or .txt file, both is seen as stream of bytes)
- **Pipelined**: **dynamic** window size set by congestion/flow control

5.1 TCP: buffers and Segments

- two buffers: send and receive, created after handshaking at both sides. Buffers are regarded as a stream of bytes
- **Max Segment Size (MSS)**: typically 1460 bytes, max app-layer data one TCP segment can carry.
- Limited by maximum transmission unit (MTU), largest link-layer frame e.g. 1500 bytes for Ethernet

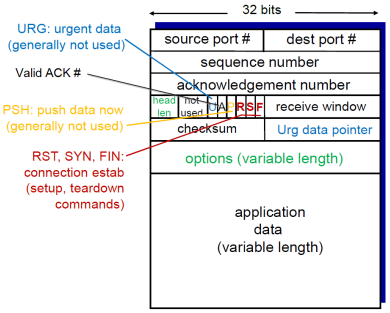
* Why 1460 Bytes specifically? Max = 1500 bytes, 20 bytes for IP header, 20 bytes for TCP header

5.2 Connection-oriented de-multiplexing

A TCP connection/socket is identified by 4-tuple (`srcIPAddr, srcPort, destIPAddr, destPort`). If there are multiple connections to same destination port and address number, with different source IP and/or source port#, server demultiplex connections to different sockets.

* Note that server is usually threaded so that it can handle multiple connections

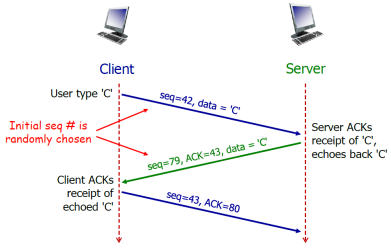
5.3 TCP Segment



- **Sequence Number**: byte number of the first byte of data in a segment (if it exceeds $2^{32}-1$ then it will wrap around back to 0).
  - Seq# of sender and receiver is different and is randomly picked at the start of the connection (to prevent confusion during packet reorder)
  - if we have a file of 500,000 bytes, MSS = 1000 bytes, then SEQ# will be 1000, 2000, ...
- **ACK number**: sequence number of the next byte of data expected by the receiver. Uses cumulative ACK (similar to go-back-N). ACK# = client_seq# + len(data)
  - Allows for **piggy-backing**: can send data along with ACK. ACK will be processed as long as ACK bit is 1.
  - **Delayed ACK**: Receiver can wait up to 500ms for the next segment before sending an ACK (saves 1 cycle of ACK)
- `ACK` bit: indicates whether segment includes an ACK and whether acknowledgement # is valid. If ACK bit is 0, indicates that segment is a pure data segment
- `SYN` bit: used to signal a request for connection setup
- `FIN` bit: used to signal connection tear down
- **Receive Window**: # of bytes rcvr willing to accept

* Note that TCP specifications doesn't say how to handle out-of-order segments, but it is usually buffered in practice (similar to selective repeat)

5.3.1 Example of transmission of TCP segment



5.4 TCP Flow Control

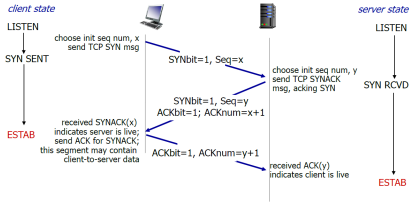
Flow control allows rcvr to control sender, making sender dynamically change the window size so that rcvr's buffer won't overflow.

Steps involved:

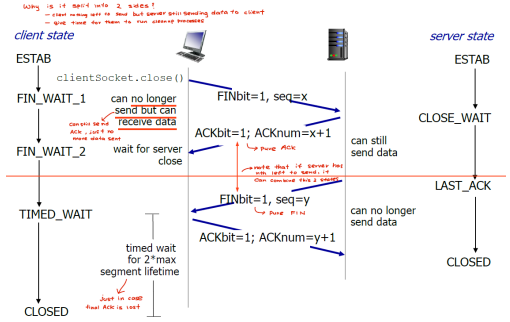
1. receiver "advertises" free buffer space (receive window in header (`rwnd`), typically defaults to 4096 bytes)
2. sender limits amount of unacked data to rcvr's `rwnd` value
3. this mechanism will hence guarantee rcvr buffer will not over flow and prevents data loss due to buffer overflow

5.5 TCP 3-way handshake

5.5.1 TCP Set-up



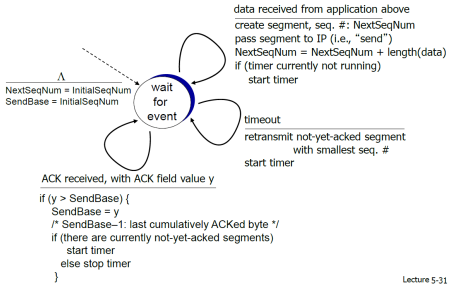
5.5.2 TCP Tear-down



5.6 TCP RDT

- TCP creates rdt service on top of IP's udt service: pipelined segments, cumulative acks, single retransmission timer, checksums
- Retransmissions triggered by: timeouts, duplicate ACKs

TCP sender (simplified)



Lecture 5-31

5.6.1 TCP ACK Generation

event at receiver	TCP receiver action
arrival of in-order segment with expected seq #. All data up to expected seq # already ACKed	delayed ACK. Wait up to 500ms for next segment. If no next segment, send ACK
arrival of in-order segment with expected seq #. One other segment has ACK pending	immediately send single cumulative ACK, ACKing both in-order segments
arrival of out-of-order segment higher-than-expect seq. # . Gap detected	immediately send *duplicate ACK*, indicating seq. # of next expected byte
arrival of segment that partially or completely fills gap	immediate send ACK, provided that segment starts at lower end of gap

5.7 TCP Timeout

Considerations:

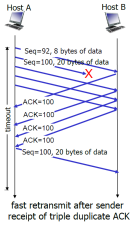
- Must be longer than RTT but RTT varies
- too short: premature timeout, unnecessary retransmissions
- too long: slow reaction to segment loss

Important equations:

$$EstimatedRTT = (1-\alpha)\times EstimatedRTT + \alpha \times SampleRTT$$
$$DevRTT = (1-\beta)*DevRTT + \beta*|SampleRTT - EstimatedRTT|$$
$$TimeoutInterval = EstimatedRTT + 4*DevRTT$$

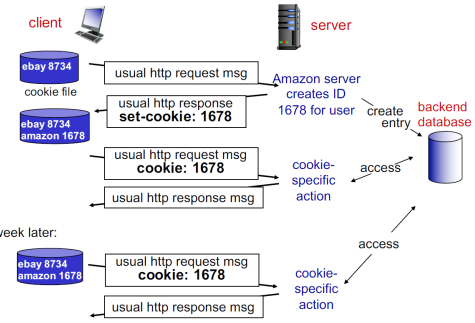
- exponential weighted moving average
- influence of past sample decreases exponentially fast
- typical value of $\alpha = 1/8$
- typically $\beta = 1/4$

5.8 TCP fast retransmit



6 Miscellaneous

Cookies Illustration

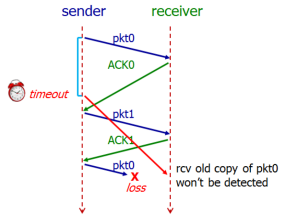


RDT Summary

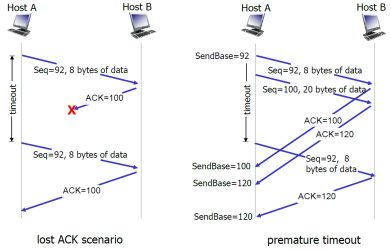
rdt	What gets corrupted?	Checksum?	Sequence Number?	Can packets be lost?
1.0	Perfectly Reliable	No	No	No
2.0	Packet	Yes	No	No
2.1	Packet, ACK and NAK	Yes	Yes	No
2.2	Packet, ACK	Yes	Yes	No
3.0	Packets, ACK	Yes	Yes	Yes

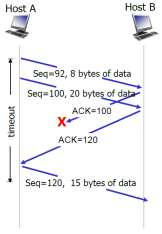
Packet Reordering Issues using RDT3.0

Solution is to increase `seq#` to hold more than 2 bits; use TTL to limit the lifespan of packets so that by the time `seq#` is reused, TTL of old packet expires; Use random `init seq#`



TCP Retransmission Scenarios





Pipe-lining Summary

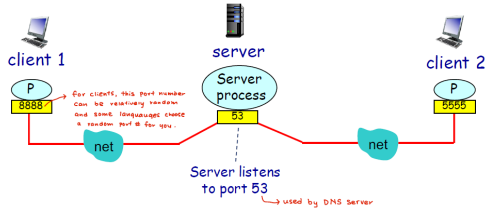
Go-back-N:

- Receiver
  - doesn't buffer or ack out-of-order pkts
  - only sends *cumulative ack* for the most-recent pkt
- Sender
  - has timer for *oldest* unacked packet
  - when timer expires, retransmit *all* unacked pkts

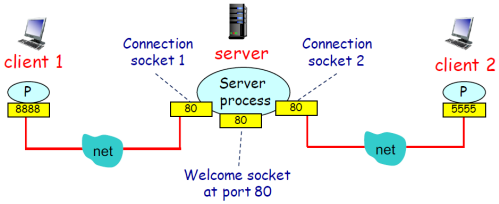
Selective Repeat:

- Receiver
  - maintain a *sliding window to buffer* out-of-order pkts
  - sends individual ack for each packet
- Sender
  - maintains timer for each unacked packet
  - when timer expires, retransmit *only* that unacked pkt

UDP Socket Illustration



TCP Socket Illustration



Throughput vs Latency

- **Latency**: how much time needed for information to be sent across to end user. (e.g. if someone on zoom speaks, how long does it take for other people to hear it?)
- Low latency important for live streaming service (e.g. Zoom, Twitch)
- **Throughput**: amount of data received in a given time period (e.g. When streaming Netflix, you need a throughput of say 5Mbps)
- Throughput does not account for delay (e.g. when streaming Netflix, you can allow for it to buffer for a few minutes, don't have to be live)

Lecture 6: IP Addressing

6.1 Network Layer

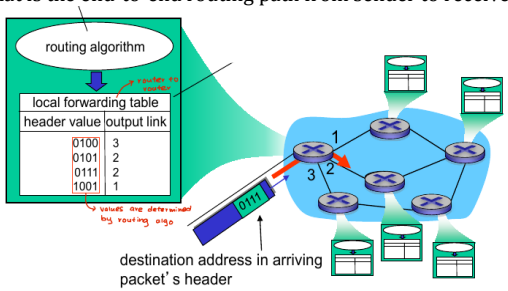
largely involves routing of datagrams from source to destination which involves the IP protocol and routing protocol

- transport segment from sending to receiving host
  - sending side encapsulates segments into datagrams
  - receiving side delivers segments to transport layer
- network protocols exist in every host and router
- intermediate routers examine header fields in all IP datagram passing through it to see the IP address so that it knows where to send to

6.2 Key Network-layer functions

2 main functions:

1. Forwarding: move packets from router's input to appropriate router output (which router to go next?)
2. Routing: determine route taken by packets from source to destination (what is the end-to-end routing path from sender to receiver?)



* Note that routing protocol must be ran before forwarding to obtain the forwarding tables

6.3 Data Plane, Control Plane

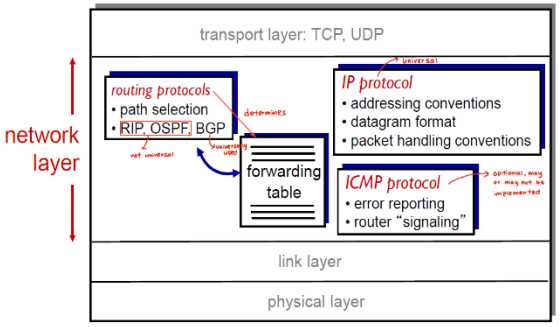
6.3.1 Data planes

- local, per-router function that takes care of forwarding from router to router
- determines how datagram arriving on router input port is forwarded to router output port based on forwarding table
- typically operates in small time scales (milliseconds)

6.3.2 Control Planes

- network-wide logic that determines end-to-end routing path
- determines how datagram is routed among routers along end-to-end path from source to host
- 2 common approaches:
  1. traditional routing algorithms: data plane and control planes both implemented in routers
  2. software-defined networking (SDN): implemented in remote servers, provides a zoomed-out overview of the network, don't have to rely on router-to-router communications

6.4 Overview of network layer



6.5 IP Addressing

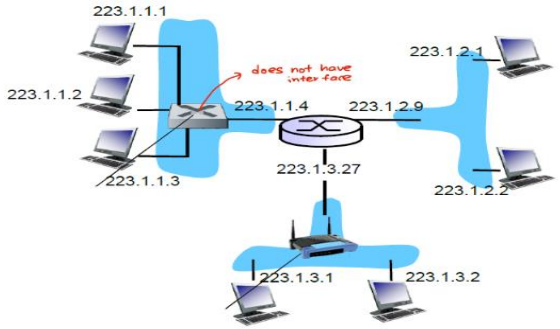
- IP addresses: 32-bit unique identifier for host or router interface
- Interface: connection between host/router and the physical link
  - routers typically have multiple interfaces
  - host typically have 2 interfaces (1 for wired ethernet and 1 for wireless)
- IP address is associated with each interface

6.5.1 How are interfaces connected

- Wired ethernet interfaces are connected by Ethernet switches (switches do not have interfaces)
- Wireless WiFi interfaces are connected by WiFi base station

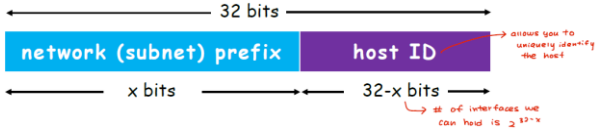
6.6 Subnets

- network formed by a group of "directly" interconnected hosts
  - hosts within same subnet can physically reach each other without intervening routers (within subnet don't require IP protocol)
  - for hosts to send data to host from another subnet, must pass through router
  - hosts in the same subnet have the same network prefix (Note: same prefix ≠ same subnet)



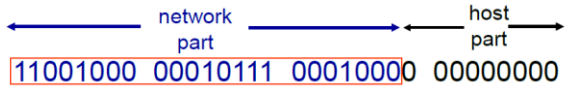
6.6.1 CIDR

- IP address comprises of 2 parts:
  - Network (subnet) prefix (Most significant x bits)
  - Host ID: used to uniquely identify the host (least significant $32 - x$ bits)
  - # of interfaces we can hold is 2^{32-x}



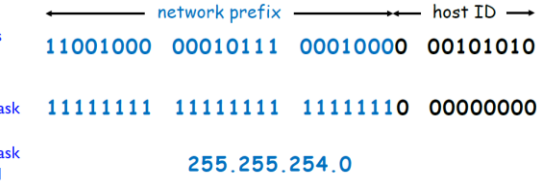
- CIDR: Classless Inter-Domain Routing
  - arbitrary length for the network prefix
  - address format: a.b.c.d/x, where x is # of bits in subnet portion of address

Example 200.23.16.0/23



6.6.2 Subnet mask

- made by setting all network prefix bits to "1"s and host ID bits to "0"s
- used to determine which network and IP address belongs to (using bitwise AND operation)



6.7 How to get IP address

6.7.1 How do ISPs obtain block of IP address

ICANN: Internet Corporation for Assigned Names and Numbers is an organization that allocates addresses, manages DNS, assigns domain names and resolves disputes. ISPs registers a block of addresses from ICANN and distributes it to their clients.

6.7.2 Special/Reserved 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.
10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	Private addresses, can be used without any coordination with an Internet registry.
255.255.255.255/32	Broadcast address. All hosts on the same subnet receive a datagram with such a destination address.

Private addresses are used within a local network (e.g. within a company/university, local area network) and are not used to communicate with devices outside of the local network

6.7.3 How do Organizations obtain a block of IP addresses?

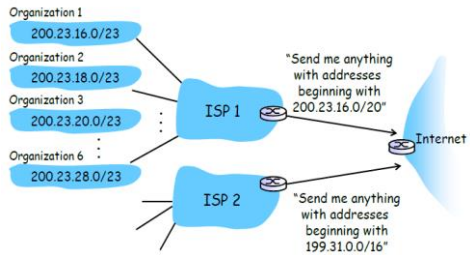
Bigger organizations like Google or Meta may buy IP addresses directly from registry. Smaller organizations who are customers of ISPs will rent from ISP's address space.

	Binary Address	Decimal Address
ISP's block	11001000 00010111 0001 0000 00000000	200.23.16.0/20
Organization 0	11001000 00010111 0001 0000 00000000	200.23.16.0/23
Organization 1	11001000 00010111 0001 0010 00000000	200.23.18.0/23
Organization 2	11001000 00010111 0001 0100 00000000	200.23.20.0/23
...	...	...
Organization 7	11001000 00010111 0001 1110 00000000	200.23.30.0/23

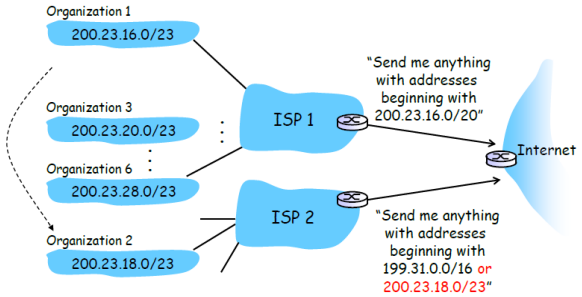
use 3 bits to differentiate 8 organizations

6.7.4 Hierarchical Addressing: Route Aggregation

Allows for efficient advertisement of routing information



Suppose that an organization wants to switch ISP, we don't have to renumber all router and hosts



6.7.5 Longest Prefix Matching

Steps to find out which router to route to:

1. Convert dotted representation of IP address to binary
2. See which destination address range matches with the prefix of the given IP address

- Packet with destination IP 200.23.20.2 ⇨ R1
  - (Binary: 11001000 00010111 00010100 00000010)
- Packet with destination IP 200.23.19.3 ⇨ R2
  - (Binary: 11001000 00010111 00010011 00000011)

match the longest prefix

Forwarding Table at R3

Destination Address Range	Destination Address Range in Binary	Next Hop
200.23.16.0/20	11001000 00010111 00010000 00000000	R1
200.23.18.0/23	11001000 00010111 00010010 00000000	R2
199.31.0.0/16	11000111 00011111 00000000 00000000	R2
...		...

6.7.6 How do end hosts get IP address?

- hard-coded by system admin in a file
- DHCP (Dynamic Host Configuration Protocol): dynamically get address for server, "plug-and-play"

6.8 DHCP

Goal: allow host to dynamically obtain IP address from network server (follows a client-server model)

- can "borrow" and renew lease on address from network
- allows for reuse of address (only hold address while connected)
- support for mobile users who want to join network

Overview:

- host broadcasts "DHCP discover" msg [optional]
- server responds with "DHCP offer" msg [optional]
- host requests IP address: "DHCP request" msg
- server sends address: "DHCP ack" msg

* Note: 1st 2 steps can be skipped if the hosts already know the IP address of the server (e.g. when it renews lease)

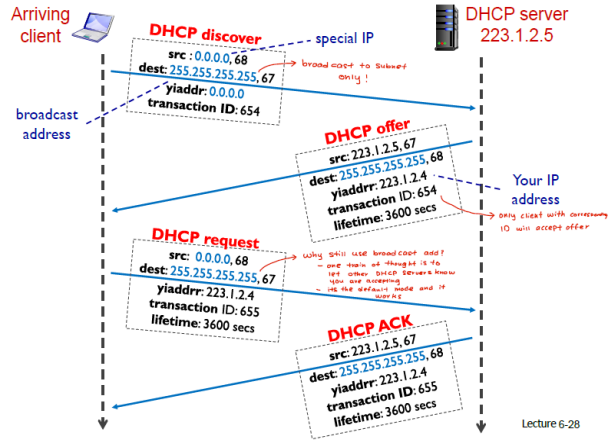
DHCP returns:

- address of first-hop router
- name and IP address of DNS server
- network mask

DHCP runs over UDP (Application layer)

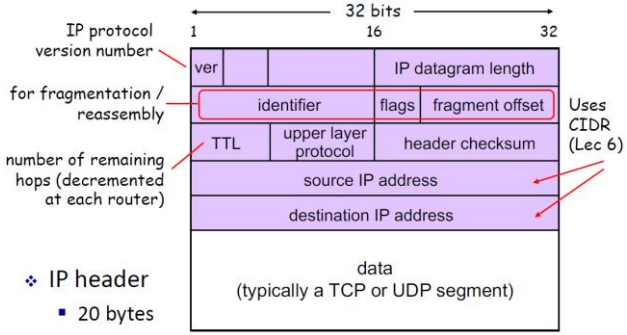
- DHCP server port number: 67
- DHCP client port number: 68

If DHCP only exists on 1 subnet, the intermediate router between multiple subnets must be a relay agent to forward DHCP requests to subnet with DHCP server. Router is known as a relay agent.



Lecture 7: Network Layer

Network layer delivers packets to receiving hosts. In this layer we are only concerned with the IP header fields of the IP datagrams passing through it.



7.1 IPv4 Datagram Format

Fields (total of 20 bytes):

- Version number: IPv4 or IPv6
- IP datagram length: length of IP datagram including IP header
- Flags: 3-bit value
- Fragment offset: 13-bit value (reason why we express offset in units of 8 bytes)
- TTL: used to prevent packet from circulating in the network forever, once TTL is 0 packet is dropped
- Header checksum: checksum for IP header (only)

7.2 IP Fragmentation, Reassembly

- Different links have different MTU (Max Transfer Unit) – the maximum amount of data a link-level frame can carry
- Datagrams that are "too large" needs to be fragmented by routers
- Reassembly of packets is done at the destination host and will only push it up the network layer once all it has received ALL the fragments
- IP header fields of each fragment are mostly the same except for the part responsible for fragmentation/assembly

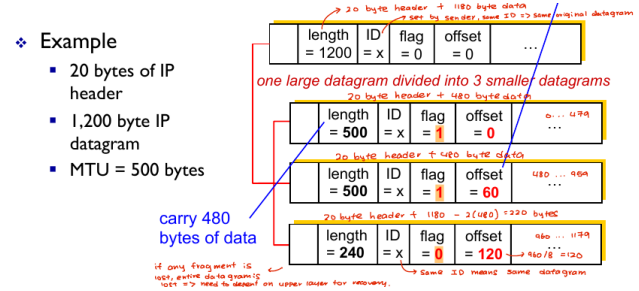
7.2.1 Fragmentation/Assembly bits in IP header

Flag (frag flag) set to:

- 1 if there is next fragment from same segment
- 0 if it is the last fragment

Offset: starting byte of payload, expressed in units of 8 bytes. If offset is not divisible by 8 then we must add padding (\0 bytes)

ID: 3-bit field that helps to identify whether different fragments belong to the same datagram



* Note that if any fragment is lost, it would mean that the entire datagram is lost → requires upper layer for recovery mechanisms

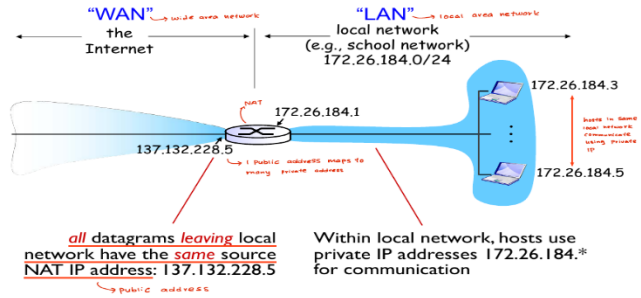
7.3 NAT (Network Address Translation)

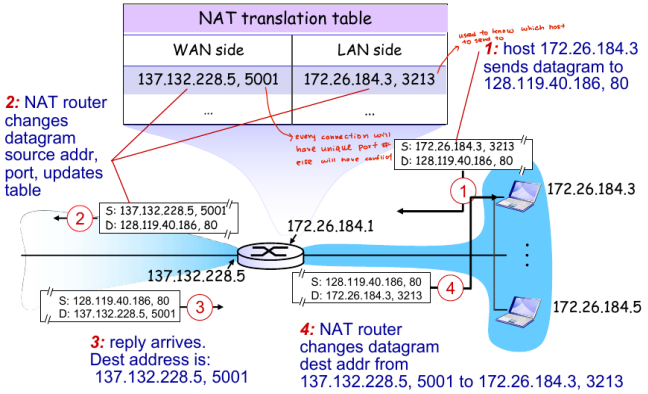
Internet has 2 types of IP addresses:

- Public IP addresses
  - globally unique and routable
- Private IP addresses (reserved IP ranges)
  - e.g. 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16
  - not globally unique (NUS and NTU can use the same IP addresses within their respective subnets)
  - used in organizations, universities, homes
  - not routable on the backbone internet, but are routable within an organization (e.g. NUS uses 172.26.x.y and any device that is connected to NUS WiFi can communicate to each other)
  - ISPs will drop any packet with src IP that are part of the private networks

NAT routers must:

- Replace (src IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
- Remember (in NAT translation table) the mapping from (src IP address, port #) to (NAT IP address, new port #)
- Replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (src IP address, port #) stored in NAT translation table.





* Max number of connection to a NAT router is limited by the # of port numbers available = $2^{16} = 65536$
* WAN side port number must be different, but LAN side port number can be the same

Motivations and benefits:

- No need to rent a large range of public IP addresses from ISP: 1 public IP for NAT routers can service 65536 devices within the subnet
- All hosts use private IP addresses. We can change addresses of hosts in local network without notifying the outside world
- We can change ISP without changing addresses of host in local net
- Hosts inside network not explicitly addressable and visible by outside world (security benefit)

Challenges

Peer-to-peer applications don't work directly since many residential computers use NAT. 2 devices who are in 2 different private networks cannot communicate directly

7.4 Routing Algorithms

The Internet exists as a hierarchy of Autonomous Systems (AS) which are organization units e.g. Facebook, Google, Singtel Starhub etc. Each AS will then have multiple subnets within them. Routing on the Internet is hence done hierarchically due to the decentralized administration.

2 types of routing:

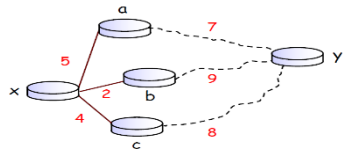
1. **Intra-AS routing**
   - Finds least cost path between 2 routes within AS
   - Single admin, no policy decisions are needed
   - Focused on performance
   - Commonly used protocol: RIP, OSPF
2. **Inter-AS routing** (NOT COVERED)
   - Handles interface between AS
   - Admin wants control over how traffic is routed
   - Policy may dominate performance
   - Uses BGP

7.4.1 Distance Vector Algorithm

- Routers know physically-connected neighbours and link costs to neighbours
- Routers exchange local views with neighbours and update own local view based on neighbour's view
- Iterative process
  1. Swap local view with direct neighbours
  2. Update own local view
  3. Repeat until no change to local view
- Follows Bellman-Ford algorithm:

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

where $d_x(y)$ is the cost of the least-cost path from x to y, $c(x,y)$ is the cost of link between router x and y, min is taken over all direct neighbours of x



$$d_x(y) = min_v\{c(x,a) + d_a(y), c(x,b) + d_b(y), c(x,c) + d_c(y)\}$$
$$= min\{12, 11, 12\} = 11$$

7.4.2 RIP (Routing Information Protocol)

- Implements DV algorithm but uses hop count as the cost metric (i.e. insensitive to network congestion)
- Entries in the routing table are aggregated subnet masks (routing to destination subnet)
- Exchange routing table every 30 seconds over UDP port 520
- "Self-repair": if no update from a neighbouring router for 3 minutes, assume neighbour has failed
- Properties:
  - Distributed: Each node received info from one or more of its neighbours, performs calculations then distributes it back to neighbours
  - Iterative: Process continues until no more updates available
  - Asynchronous: does not require all node to operate in step with each other

7.5 ICMP (Internet Control Message Protocol)

- used by hosts & routers to communicate network-level information
  - error reporting: unreachable host/network/port/protocol, TTL reach 0
  - echo request/reply (ping, traceroute)
- Carried in IP datagrams (ICMP header starts after IP header)
- Types and Codes:

❖ 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

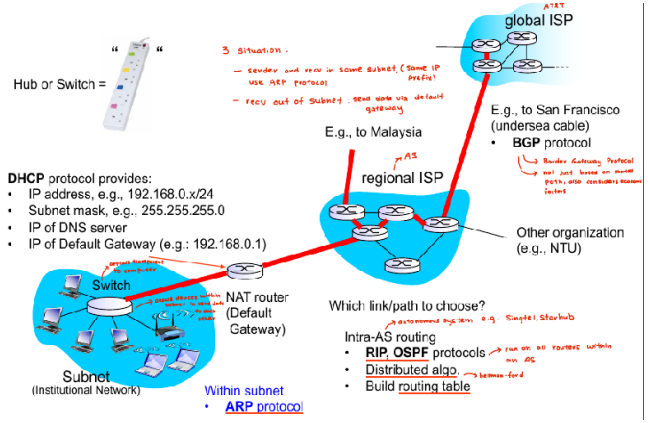
Selected ICMP Type and subtype (Code)

Lecture 8: Link Layer Part 1

IP datagram may travel through multiple routers and links before it reaches its destination.

- Link layer sends datagram between adjacent nodes (hosts or routers) over a single link
  - IP datagrams are encapsulated in link layer frames for transmission
  - Different link-layer protocols may be used on different links
    - each protocol provides a different set of service (e.g. PPP, 4G/5G, Ethernet)
    - however, there is a basic set of services that the link layer provides to network layer

8.1 Routing Overview

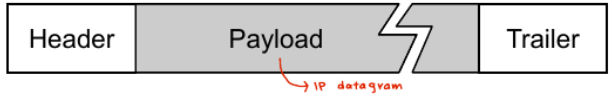


Mainly 3 scenarios:

- sender and receiver in same subnet (same IP prefix): use ARP protocol
- receiver outside of subnet: send data via default gateway, use RIP/OSPF protocol to route within AS
- receiver outside AS: use BGP protocol

8.2 Link Layer Services

1. **Framing**: Encapsulate datagram into frame, adding header and trailer
   - Trailer and header changes at each link



2. **Link access control**: coordinates which nodes can send frames at certain point of time when multiple nodes share a single link
3. **Reliable delivery**: error detection and correction, often used on error prone link e.g. wireless link but not used on low bit-error link e.g. fibre
4. **Error Detection**: detect errors caused by signal attenuation or noise and possibly signal sender for retransmission or drop corrupted frame
5. **Error Correction**: receiver identifies and correct bit error(s) without retransmission

8.3 Link Layer Implementation

- Link layer is implemented in "adapter" (aka Network Interface Card [NIC]) or on a chip
  - e.g. PCIe card, 802.11 (WiFi card)
- Typically implemented as hardware for faster speed
- Adapters are semi-autonomous, implementing both link and physical layer

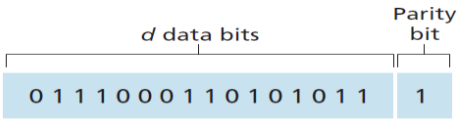
8.4 Error Detection and Correction

Popular error detection scheme:

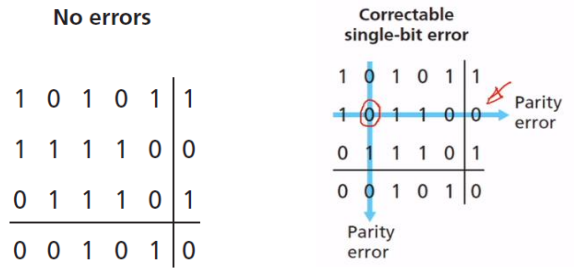
- Checksum (used in TCP/UDP/IP)
- Parity checking
- CRC (Cyclic Redundancy Check): typically faster than checksum due to using bitwise arithmetic, often used in link layer
- Error detection schemes are not 100% reliable
  - may miss some errors but rarely
  - larger error detection and correction (EDC) fields yields better detection but uses more overhead
    - e.g. redundant information, additional calculation etc.

8.4.1 Parity Checking

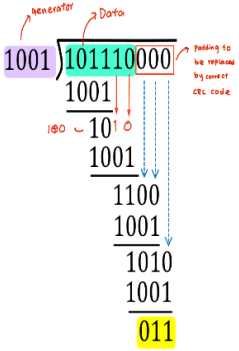
1. Single bit parity
   - detect single bit errors in data (unable to correct)
   - parity bit is 1 if # of "1" bit is odd else 0 if # of "1" bit is even
   - only has 1 bit overhead



2. Two-dimensional bit parity
   - can detect and correct single bit errors in data
   - can detect any 2-bit error in data



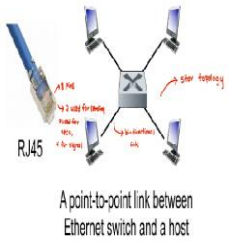
8.4.2 Cyclic Redundancy Check (CRC)



- Powerful error-detection (high error detection rate) used in ethernet and WiFi
  - D: data bits, viewed as binary number
  - G: generator of $r + 1$ bits agreed by sender and receiver
  - R: generate CRC of r bits (the higher the R value, the higher the detection rate)
- CRC calculation done in bit-wise XOR without carry/borrow, done in link layer since it has access to hardware. Not done in upper layer since it would have to rely on software which is slow
- Receiver divides (D, R) by G and would know if an error occurred if non-zero remainder detected

8.5 Types of Network Links

1. Point-to-Point Link



- sender and a receiver connected by a dedicated link
- Example protocols: Point-to-point (PPP), Serial Line Internet Protocol (SLIP)
- e.g. RJ45 Ethernet, point-to-point link between ethernet switch and host

2. Broadcast link (shared medium)

- multiple nodes connected to a shared broadcast channel
- when a node transmits a frame, the channel broadcasts the frame and each other node receives a copy
- e.g. 802.11 WiFi, Satellite (StarLink), Ethernet with Bus Topology

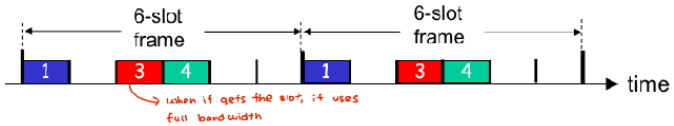
8.5.1 Multiple Access Protocols

- In a broadcast channel, if ≥ 2 nodes transmit simultaneously, every node will receive multiple frames at the same time → frames collide at nodes and data will be invalidated
- Have multiple access protocol which is a distributed algorithm that determines how nodes share channel, i.e. when a node can transmit
  - coordination about channel sharing must use channel itself → no out-of-band channel signalling
- 3 broad classes:
  - Channel partitioning
    - Divide channel into fixed, smaller pieces (e.g. time, slots, frequency)
    - allocate piece to node for exclusive use
  - "Taking Turns"
    - nodes take turns to transmit
  - Random Access
    - channel not divided; collisions still possible
    - must have ways to recover from collision

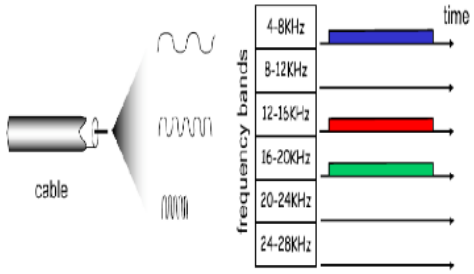
8.5.2 Channel Partitioning Protocols

No collisions, but low efficiency of bandwidth

- TDMA (time division multiple access)
  - Access to channels in "rounds"
  - Each node gets fixed length slot (length = frame transmission time) in each round)
  - Unused slots are idle

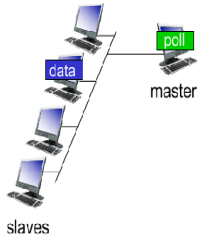


- FDMA (frequency division multiple access)
  - Channel spectrum divided into frequency bands (e.g. radio uses this)
  - Each node assigned 1 frequency band
  - Unused transmission time in frequency bands go idle
  - Nodes don't have to wait but only uses $\frac{1}{n}$ of bandwidth



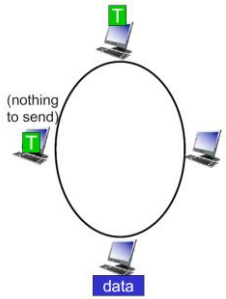
8.5.3 "Taking Turns" Protocol

- Polling



  - master node "invites" slave nodes to transmit in turn (no idling)
  - **concerns**:
    - polling overhead: last node must wait for master to poll whole list before getting its turn (minor overhead)
    - single point of failure (master node)

- Token passing

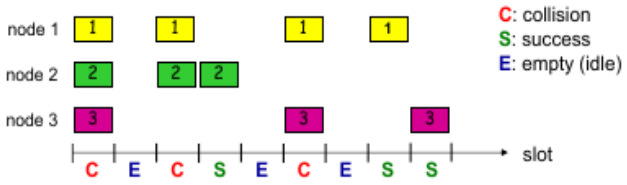


  - control token passed from 1 node to the next sequentially (round robin)
  - **Concerns**:
    - single point of failure (token)
    - if client with token disconnect must ensure only 1 token is regenerated (lots of code to handle error)
    - token overhead

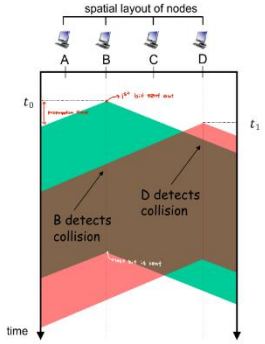
8.5.4 Random Access Protocol

- when node has packets to send
  - no prior coordination among nodes
  - if ≥ 2 transmitting nodes → collision
- Random Access Protocol must specify:
  - how to detect collision
  - how to recover from collision (e.g. delayed retransmission)

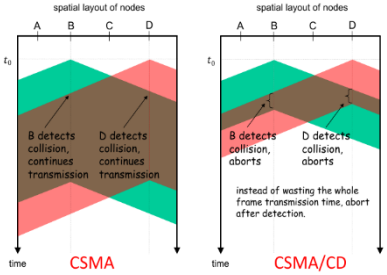
- Slotted ALOHA
  - Assumes all frames are of equal size, time is divided into slots of equal length (length = frame transmission time), nodes start to transmit only at beginning of slot
  - **Operations**:
    - listens to channel while transmitting (collision detection)
    - if collision happens: node attempts to retransmit a frame in subsequent slot with probability p until it succeed
    - p small: more unused slots, p big: collisions



- Pure (unslotted) ALOHA
  - no slot, no synchronization
  - when there is a fresh frame → transmit immediately
  - Chance of collision increase!
    - Frame sent at t_0 collides with other frames sent within 1 time unit: $(t_0, t_0 + 1)$
- CSMA (Carrier Sense Multiple Access)

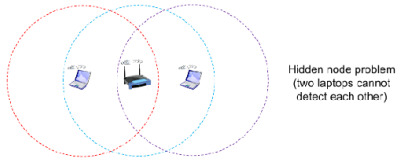


  - sense channel before transmission
    - channel sensed idle: transmit
    - channel sensed busy: defer transmission
  - collision still can happen e.g. when two nodes sense the channel idle at the same time (due to propagation delay) and both start transmission
- CSMA/CD (Collision Detection)
  - Works similarly to normal CSMA but transmission aborted when collision detected (reduce channel wastage)



  - Retransmit after random time
  - If frame size is too small:
    - collision happens without being detected
    - no retransmission
    - must have minimum frame size: Ethernet uses minimum frame size of 64 bytes

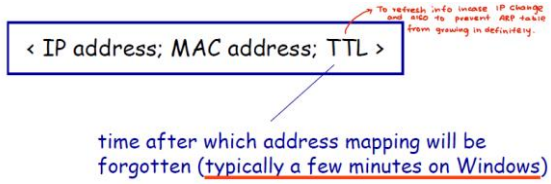
- CSMA/CA (Collision Avoidance)
  - Collision detection is easy in wired LANs, but difficult in wireless LANs. For example,



  - 802.11 (Wi-Fi) uses CSMA/CA protocol instead.
    - Receiver needs to return ACK if a frame is received OK.

Lecture 9: Link Layer Part 2

9.1 MAC (Media Access Control) Address



- Every NIC has a 48-bit MAC address (aka Physical or LAN address) e.g. 5C-F9-DD-E8-E3-D2
  - MAC address allocation administered by IEEE and first 3 bytes identifies vendor of adapter
  - used to send & receive link layer frames
  - On receiving a frame, NIC checks if destination MAC address matches its own address using hardware (very fast).
    - If yes, extracts datagram and passes to protocol stack.
    - Otherwise, discards the frame.

9.1.1 IP vs MAC Address

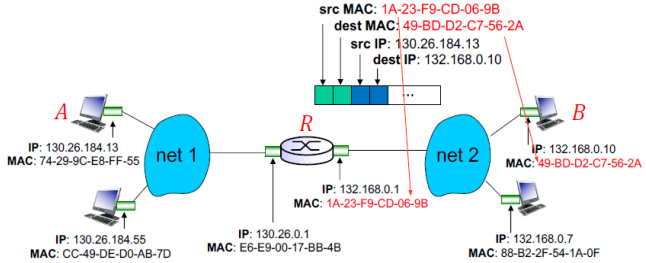
IP address	MAC address
32 bits in length	48 bits in length
network-layer address used to move datagrams from source to dest.	link-layer address used to move frames over every single link.
Dynamically assigned; hierarchical (to facilitate routing)	Permanent, to identify the hardware (adapter)
Analogy: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

9.2 ARP (Address Resolution Protocol)

- Each IP node (routers, hosts) has an ARP table
  - acts as a caching mechanism
  - empty when computer starts up and slowly fills ups with each request
  - stores mapping of IP address and MAC address of other nodes in the same subnet

9.2.1 Sending Frames

- In the **same subnet**:
  - If *A* knows *B*'s MAC address from ARP table:
    - create frame with *B*'s MAC and send it
    - Only *B* will process this frame, other nodes may receive the frame but will ignore it
  - Else:
    - *A* **broadcast an ARP query packet** containing *B's* IP address (Dest MAC address set to **FF-FF-FF-FF-FF-FF**)
    - Other nodes in subnet will receive packet but only *B* replies with its own MAC address
    - *A* will cache *B*'s MAC address to its ARP table
- To **another subnet**:
  - *A* cannot just put *B*'s MAC address as dest MAC since router will ignore the frame due to mismatch of MAC address
  - *A* create a link-layer frame with router's MAC address and *B*'s IP address.
  - Router will then move datagram to outgoing link, construct a new frame with *B's* MAC address



9.3 Local Area Networks (LAN)

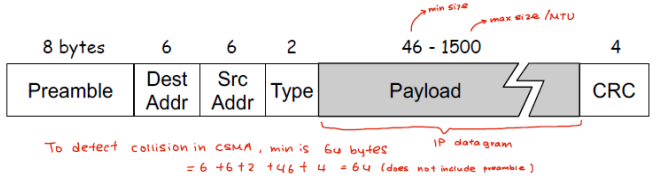
- computer network that interconnects computers within a geographical area (e.g. office, universities)
- Examples:
  - Ethernet: IEEE 802.3 standard
  - WiFi: IEEE 802.11 standard
- Exists other networks like MAN (Metropolitan area network), WAN (wide area network)
- Within LAN can have many subnets

9.4 Ethernet

9.4.1 Ethernet Topology

- **Bus Topology**: All nodes are connected and can collide with each other.
- **Star Topology**: switch in the centre and nodes are connected to that switch. They do not collide with each other.

9.4.2 Ethernet Frame Structure



- **Preamble**: 7 bytes of `10101010` and 1 byte of `10101011`
  - used to synchronize receiver and sender clock rate (allow receiver to know how long is 1 bit) so that it can know how long is 20 bits of 0 accurately
- **Destination MAC Address**: If the NIC receives a frame with either matching address or the broadcast address, it will pass the data in the frame to the network layer protocol. Else it will discard.
- **Type**: Higher layer protocol used, usually IP
- **CRC**: for corruption detection
- **Payload**: Minimum size of 46 bytes for collision detection, max size is 1500 (MTU).
  - maximal payload size makes it easier to implement efficient data buffer management algorithms in the NICs, switches and routers.
  - also helps to make sure that senders do not "hog" shared links and other senders get a chance to transmit as well.

9.4.3 Ethernet Data Delivery Services

- **Connectionless**: No handshaking between sender and receiver.
- **Unreliable**: No ACK or NAK. Data in dropped frames will be recovered only if the initial sender uses higher-layer rdt (e.g. TCP). Corrupted frames are also dropped.
- **Multiple Access Protocol**: CSMA/CD with binary (exponential) backoff

9.4.4 Ethernet CSMA/CD Algorithm

1. NIC receives datagram from the network layer and creates a frame.
2. If NIC senses that the channel is idle, it will start frame transmission. Else it will 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, which tells all other nodes that a collision has been detected
5. NIC enters exponential backoff
   - after m^{th} collision, NIC chooses K at random from $\{0,1,2,\ldots,2^m - 1\} \rightarrow$ more collisions, longer wait time (attempt to adapt retransmission to estimated current load)
   - NIC waits K*512 bit times and returns to step 2

9.4.5 Ethernet Switch

Link-layer device used in LAN that stores and forward ethernet frames. It examines incoming frame's MAC address and forward frame to ≥ 1 outgoing links.

- **Layer 2 device**: Only implements physical and link layer protocols. No IP address.
- **Transparent to hosts**: hosts are unaware of presence of switches
- **Collision-free**: each host have dedicated (full duplex) connection to switch and switch buffers frames→ hosts can send frames to each other simultaneously.
  - Note that that may be collisions within the switch itself

9.4.6 Switch Forwarding Table

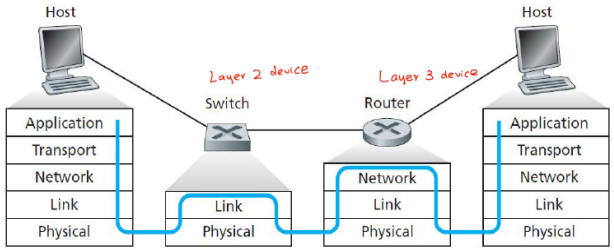
A switch has multiple interfaces, and it will need to know which nodes are reachable via which interface. This is done via switch forwarding tables, which have the following entry format:

< MAC address of Host, interface to reach host, TTL >

- **Self-learning**: Whenever the switch receives a frame from host A, it will record the interface that reaches A in its switch table, and send all frames for A to that interface.
- **Broadcast**: If the destination host is not found in the switch forwarding table, the switch will broadcast the frame to all outgoing links.
- **Hierarchical**

9.4.7 Switch vs Routers

- 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



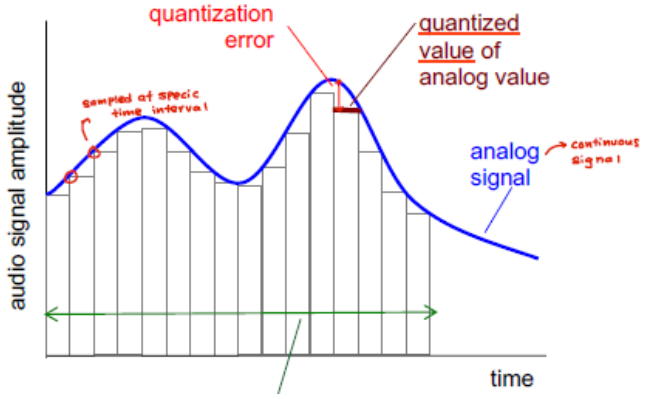
Switch	Router
Layer 2 Device	Layer-3 Device
Self-learning	Need Manual Configuration
Forwards link layer frames	Forward IP datagrams
Used in a subnet	Used to connect subnets

Lecture 10: Multimedia Networking

There is a rise in use of multimedia applications where it is projected that 80% of traffic will be video in 2021. Videos are delivered **Over-the-Top (OTT)**

10.1 Properties of Audio

- **Sampling**: To convert audio analog signal to digital signal, we must sample the signal at some constant rate.
  - **Telephone**: 8000 samples/sec
  - **CD**: 44,100 samples/sec
- **Quantisation**: Each sample is rounded to one of a finite number of values. Number of quantisation typically in power of 2. e.g. 2^{16} for CD.
  - Nyquist-Shannon sampling theorem: $f_s \geq d \times 2$, f_s = sampling frequency, d = highest signal frequency



- **Concatenation and Decoding:** All values represented as bits and all samples will then need to be concatenated back together. To decode, we convert it back to analog signal. This signal is an **approximation of the original** since some sounds are lost in sampling and quantisation.
  - **Speech Encoding:** 8000 samples/sec, 256 quantized values (8bits) = 64kbps
  - **CD:** 44,100 samples/sec, 16 bit quantized value = 705.6kbps for mono and 1.411 Mbps for stereo
- **Compression:** lossless (e.g. zip) and lossy (mostly used for media)

10.2 Properties of Video

- Video is just a sequence of images displayed at a constant rate e.g. 30 images/sec (fps)
- each image is an array of pixel and each pixel is represented by RGB bits (8 bits for each of R,G,B)
- Redundancies:
  - **Spatial:** instead of sending N values of same colour, send (colour, N)
  - **Temporal:** only send difference between 2 consecutive image instead of complete frame
- **Bit Rate:**
  - **CBR (constant bit rate):** video encoding rate. Might lead to variable quality whereby we have low quality for complex frames and high quality for simple frame
  - **VBR (variable bit rate):** video encoding rate changes according to spatial and temporal coding.
- **Compressors:**
  - **MPEG 1 (CD-ROM):** 1.5 Mbps
  - **MPEG2 (DVD):** 3-6Mbps
  - **MPEG4/H.264:** used in internet, < 2Mbps

10.3 3 Application Types

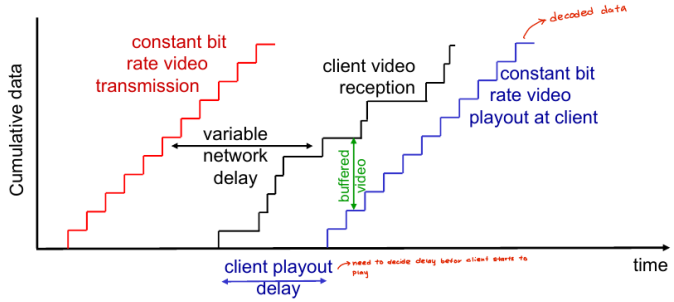
- **streaming, stored audio, video**
  - streaming: can begin playout before downloading entire file
  - stored (at server): transmit faster than audio/video will be rendered (need to buffer at client)
  - e.g. YouTube, Netflix, Hulu
- **conversational ("two-way live") voice/video over IP**
  - interactive nature limits delay tolerance
  - e.g. Zoom, Skype, Teams
- **streaming live ("one-way live") audio, video**
  - can be a few seconds late
  - e.g. live sporting event

10.4 Streaming Stored Video

10.4.1 Challenges

- **Continuous playout constraint:** once client playout begins, playback must match original timing
  - hard to achieve as network delays are variable (jitter) so will need client-side buffer to match playout requirement
- **interactivity**: pause, fast-forward, rewind, jump through video etc.
- video **packet may be loss**

10.4.2 Client-side Buffering



When video arrives at client, we do not immediately play the video, instead we keep a reserve of video in the buffer of client. Only play when we have certain amount of video in buffer (typically a few seconds worth).

- **Absorbs variation in delay**: If a certain piece is delayed, as long as it arrives before the reserves are depleted, user will not notice delay
- **Bandwidth change resistance**: If bandwidth briefly drops, user can continue to continuously playback as long as buffer not depleted
- If **fill rate < playout** rate: buffer eventually depletes (cause freezing of video until buffer fills again)
- If **fill rate > playout** rate: less likely to deplete buffer but larger delay until user begins watching

10.5 UDP streaming

- **Pushed-based streaming**: send rate = encoding rate = constant rate. Transmission rate can be oblivious to congestion level since no rate-control protocol in UDP
- **short playout delay**: 2-5 seconds to remove network jitter
- **UDP may not go through firewall**
- **Uses RTP (real-time transport protocol)**
- **Small buffer:** usually buffer < 1s

10.6 HTTP Streaming

- **Pull-based streaming**: retrieve multimedia file through HTTP GET request
  - **bigger length of data**: each segment can be a few MiB
  - sends at max rate under TCP
- **Fill rate fluctuates:** due to TCP congestion control, retransmission
- **Larger playout delay:** wait till buffer fills up
- HTTP/TCP passes through firewall more easily

10.7 VoIP

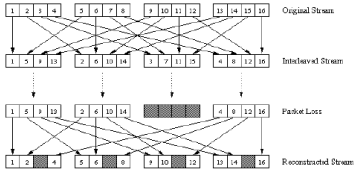
- Requirements:
  - delay of < 150ms is considered good, any delay > 400ms would be noticeable and may impair interactivity
- Characteristics:
  - **Talk spurts**: We don't always talk during a conversation. Hence we can just generate packets during talk spurts at 64kbps. 20ms chunks at 8KB/s = 160 bytes
  - application-layer header added to each chunk and encapsulated in UDP or TCP segment and then sent at 20ms interval during talk spurts
- Losses/delays:
  - **network loss**: IP datagram loss due to network
  - **delay loss**: IP datagram arrive too late for playout, typically 400ms max tolerable delay
  - loss tolerance: loss between 1-10% is okay
- fixed playout delay: receiver attempts to playout each chunk at exactly q ms after chunk was generated
  - any chunk that arrives at chunk t+q will be too late
  - tradeoff in choosing q
    - large q: less packet loss, more delay
    - small q: low delay but more loss
  - Solution: **Adaptive playout Delay**
    - aim to estimate network delay and adjust playout delay at beginning of each talk spurt
    - silent periods are compressed and elongated
    - chunks still playout at every 20ms

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

history; delay of latest packet

delay estimate after ith packet | small constant, e.g. 0.1 | time received - time sent (timestamp) = measured delay of ith packet

- Recovery from Packet Loss
  - Forward Error Correction (FEC): Send enough bits to allow recovery without retransmission
    - **Simple FEC:** for every N chunks, we create a redundant chunk by XOR-ing the original N chunks and send N + 1 chunks (1/N overhead). Allows us to reconstruct 1 chunk from N+1 chunk.
    - **Piggybacking**: Send lower resolution stream as redundant information
  - Interleaving:



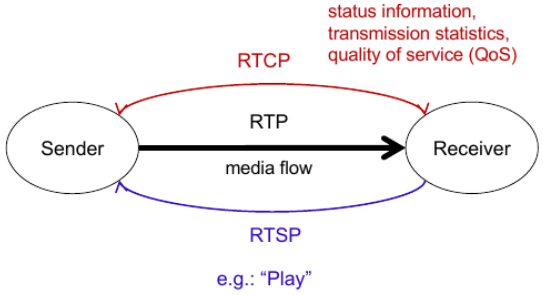
interleaving to conceal loss:

- audio chunks divided into smaller units, e.g. four 5 msec units per 20 msec audio chunk
- packet contains small units from different chunks
- if packet lost, still have *most* of every original chunk
- no redundancy overhead, but increases playout delay

10.8 Real-time Protocol

RTP specifies packet structure for packets carrying audio, video data. RTP packet provides: payload type identification, packet sequence number numbering, time stamping.

- RTP runs in end systems.
- **UDP**: RTP packets are encapsulated in UDP segments. RTP libraries provide **transport-layer** interface that extends UDP
  - port#, IP address
  - payload type identification
  - packet sequence numbering
  - time-stamping
- **Interoperability**: if 2 VoIP applications both incorporate RTP, they may be able to communicate with each other
- **RTP Suite**:



- audio chunk + RTP header forms a RTP packet which is encapsulated in UDP segment
- **RTP Header**:

  - **payload type** (7bits): indicate type of encoding used. If sender changes encoding during call, sender informs receiver via payload type field (refer to lect notes)
  - **Sequence#** (16bits): increment by one for each RTP packet sent. Used for detecting packet loss and restoring packet sequence
  - **timestamp** (32bits): sampling instant of first byte in RTP packet (refer to lect notes for more info)
  - **SSRC** field (32 bits): identify source of RTP stream. e.g. Used to distinguish which skype stream if you have multiple skype client opened

10.9 WebRTC

WebRTC is an open framework for the web that enables Real-Time Communications (RTC) capabilities in the browser, via simple JavaScript APIs. It makes use of RTP and RTCP .

10.10 DASH (Dynamic Adaptive Streaming over HTTP)

Advantages:

- **Simple server**: regular web server (no state, scalable)
- **No firewall issue**: uses standard port 80, TCP
- Standard (image) web caching works

Disadvantages:

- DASH based on media segment transmission, typically 2-10s in length
- **Does not provide for low latency** for interactive 2-way communication
- **High Overhead:** original media need to be split into different streamlet and transcoded to different quality

10.10.1 How it works?

Idea: Split video into streamlets, where each streamlet is of a different quality (e.g. low, medium high)

- Webserver provides a playlist (typically of .mpd format) which is a file that specifies all available qualities and all streamlets available
- As client plays video, execute adaptive bitrate algorithm (ABR) to determine which segment to download next

Lecture 11: Network Security

11.1 What is Network Security

Properties:

- **Confidentiality**: Only sender should be able to read and understand the message.
- **Integrity**: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection
- **Availability/Availability**: Services must be available to authorised parties who need it.
- **Authenticity**: Ability for sender and receiver to confirm the identity of each other.
- **Non-repudiation**: Sender cannot convincingly deny having sent something.

Potential Attacks:

- **Eavesdrop**: intercept message
- **Insert** message into connection (compromise integrity)
- **Impersonation**: can fake (spoof) source address in packet (or any field in packet)
- **Hijacking**: "take over" ongoing connection by removing sender or receiver, inserting himself in place
- **Denial of Service (DoS)**: prevent service from being used by others (e.g., by overloading resources)

Terminologies:

- **cipher-text-only attack:** attacker has ciphertext that he can analyse. Broken either by brute force or statistical analysis
- **known-plaintext-attack:** attacker has plaintext corresponding to ciphertext (e.g. attacker know pairing of letters in Caesar cipher)
- **chosen-plaintext-attack:** attacker can get ciphertext for plaintext

11.2 Symmetric Key Cryptography

Both Alice and Bob share same key. Challenge is how to communicate said key to the other party securely.

- **Substitution Cipher:** Replace one character for another. The encryption key is a one-to-one mapping of the characters used.
  - Mapping is static
  - Key size is huge
  - Weak against frequency attack
- **Caesar Cipher**: Left or right rotation of the alphabets or characters used. Encryption key is the shift number.
  - Still has static mapping but key size is only 1 int
- **Multiple Substitution Cipher**:
  1. Choose *n* substitutions ciphers.
  2. Choose a random cyclic pattern of numbers 1 to *n*.
  3. Repeat this cyclic pattern until it's the plaintext length.
  4. For each character in plaintext, find the corresponding number, then use that substitution for that character.

Key need not be just a n-bit pattern which makes it **more resistant to frequency attack**

- **DES & AES**

	DES	AES
Date designed	1976	1999
Block size	64 bits	128 bits
Key length	56 bits (effective length); up to 112 bits with multiple keys	128, 192, 256 (and possibly more) bits
Operations	16 rounds	10, 12, 14 (depending on key length); can be increased
Encryption primitives	Substitution, permutation	Substitution, shift, bit mixing
Cryptographic primitives	Confusion, diffusion	Confusion, diffusion
Design	Open	Open
Design rationale	Closed	Open
Selection process	Secret	Secret, but open public comments and criticisms invited
Source	IBM, enhanced by NSA	Independent Dutch Cryptographers

11.3 Public Key Cryptography

Sender and receiver has a pair of private and public key. They do not share their private key but public key can be known to anyone. Often **used in HTTPS** protocol

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

Requirements:

Given public key, it should be computationally impossible to compute private key.

11.3.1 RSA

1. Choose two distinct large prime numbers p and q (e.g. 1024 bits each)
2. Compute $n = pq$ and $z = (p-1)(q-1)$
   - n used as the modulus for both public and private key
3. Choose e such that $1 < e < \lambda(n)$ and e is coprime with z (no common factor)
4. Choose d such that $e \cdot d - 1$ is exactly divisible by z i.e. $ed \bmod z = 1$
5. Public key is (n, e) and private key is (n, d)

11.3.2 RSA Encryption and Decryption

1. To encrypt m ($< n$), compute $c = m\text{^}e \bmod n$

2. To decrypt c, compute $m = c\text{^}d \bmod n$

11.3.3 Why is RSA Secure?

Factoring n which is a huge number is computationally difficult without knowledge of p and q

11.3.4 Combining Public and Symmetric Key Cryptography

Using Public key cryptography to encrypt messages is too slow (DES is 100 times faster than RSA). Hence, we can first use RSA to conduct symmetric key exchange and then use the symmetric key to secure future communications.

11.4 Recommended Key Lengths

	1982	1995	2002	2010	2020	2030	2040
Sym	56	66	72	78	86	93	101
RSA	417	777	1028	1369	1881	2493	3214

11.5 Digital Signature

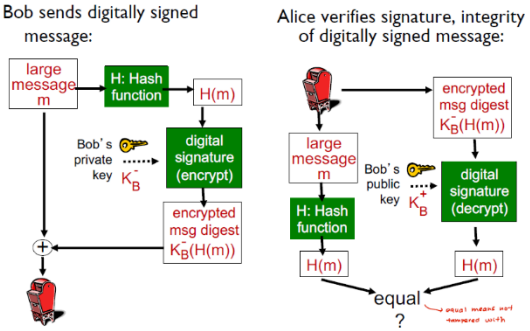
We want to have a signature so that we know who is sending the message and that that person cannot refute having sent it, i.e. **authenticity** and **non-repudiation.**

The easiest way to achieve this is to use private key of RSA to sign the message then the other person can use public key of RSA to decrypt and verify that it is indeed from the authentic sender. However this is **computationally expensive**.

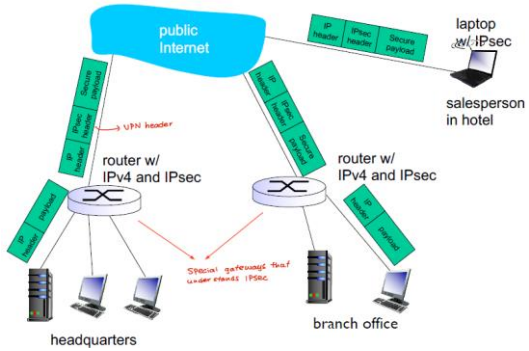
11.6 Hash Functions

Properties:

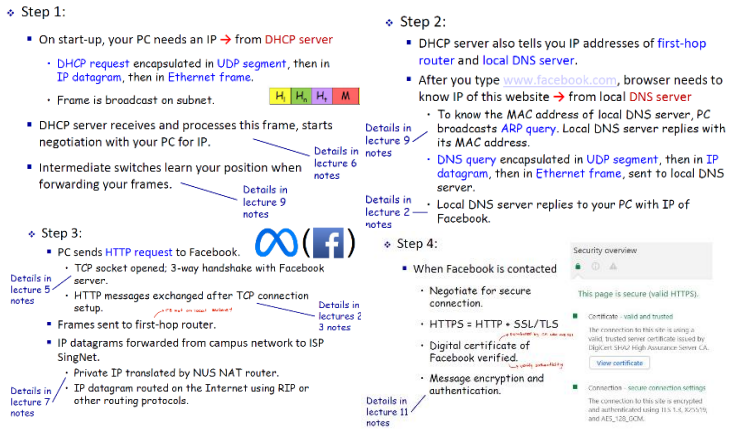
- many-to-1 (pigeon hole principle)
- produces fixed size message digest
- Given a digest x, it is computationally infeasible to find m such that $x = H(m)$
- Internet checksum has some properties of Hash but it is easy to find another message with same hash function



- Common hash function:
  - **MD5**: Computes **128-bit** message digest in 4-step process. Is actually obsolete now as collisions can be found within a minute.
  - **SHA-1**: 160-bit message digest
- **Password Hashing:**
  - Passwords are not stored in plaintext but rather 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.
- **Public Key Certification**
  - We need a trustworthy source to get the public keys from
  - Public keys are provided by an organisation known as **Certificate authority (CA)**
  - An entity, e.g. a host or a router, needs to register its public key with a CA, often requiring to provide some proof of identity.
  - CA creates a certificate binding that entity to its key (along with a lot of other information) and this certificate is digitally signed by the CA.
  - A sender will thus get the receiver's certificate from a CA and apply the CA's public key to the certificate to get the receiver's public key.
- VPNs:
  - Institutions often want **private network** for security but its very costly (separate routers, links, DNS, infrastructure)
  - VPN encrypts inter-office traffic before sending out into the public internet and is logically separate from other traffic



Full Path Taken After Entering www.facebook.com



Miscellaneous

Modern routers are capable of many functions:

- acts as DHCP server for local network
- acts as DHCP client for ISP (gets dynamic IP address from ISP)
- acts as NAT gateway
- routing

QnA

Why not use MAC instead of IP for routing?

An IP address logically comprises two parts: network prefix and host ID. This is designed to facilitate routing: routers check prefix and deliver a packet to an aggregated destination network. If MAC address is used instead, hierarchical routing cannot be achieved. For example, MAC address is burnt in ROM and usually cannot be changed. When people carry their laptops around the world, devices in a subnet won't have common prefix in MAC addresses. This makes routing difficult as routers must remember routing for every single MAC address.

FDMA vs TDMA

- **FDMA:** Good for sending messages that requires short latency since channel will be readily available. However, if big files are sent then FDMA might take longer than TDMA since it only has 1/N * Bandwidth per channel
- **TDMA:** Good for sending large files in a short time since it has dedicated bandwidth usage for 1/N of the time. Bad if you want to send message with low latency since you have to wait for the next slot.

Applications that won't function properly from a computer connected to a NAT router

- Web servers
- Email servers

Things that works:

- Messaging client
- Networked printer
- Email client

Utilization of Link in Sliding Window

$$U = \frac{NL/R}{L/R + RTT} = \frac{NL}{L + R \times RTT}$$

where U = utilization, L = packet length, R = link bandwidth, N = window size