1 Network overview

1.1 Access networks

Digital subscriber line (DSL) uses existing telephone lines to transmit data at high frequencies, and voice at low frequencies. These two frequency ranges are separated using a DSL access multiplexer (DSLAM), with the voice frequencies going to the telephone network while the data goes to the internet.

Cable based access uses frequency division multiplexing (FDM) to allow many channels to transmit simultaneously by assigning each a frequency band. It can use TV lines, or dedicated internet lines. Shared wireless access networks connect end systems to routers via an access point or base station.

Wireless local area network (WLAN) typically connect in or around a building (100ft range) with an up to 10 Gbps transmission rate. A wide area cellular access network is provided by a mobile service provider, with ranges in the 10s of kilometres and lower transmission rates (100 Mbps for 4G, 1 Gbps for 5G). An enterprise network consists of a mix of wired and wireless access points.

1.2 Physical media

Twisted pair - a pair of insulated copper wires twisted together, which reduces electromagnetic interference. Cat5 and Cat6 cables use a number of twisted pairs, reaching 100 Mbps-1 Gbps and 10 Gbps speeds respectively.

Coaxial cables consist of an central copper wire, a layer of insulation, another layer of copper, and an outer sheath. This allows data to be bidirectionally transmitted, one way on each layer. Broadband cables are coax, with multiple frequency channels on each cable and can reach a speed of 100 Mbps per channel.

A **fibreoptic cable** contains a glass fibre carrying light pulses, with each pulse being a bit. They are extremely high bandwidth, with transmission rates up in the 10-100+ Gbps range, have a low error rate as they are immune to EM interference, but are expensive and inflexible.

Wireless radio - instead of using a cable, radio/microwaves are used to carry signals, which propagate across the environment. This removes the need to physically connect devices to the network, but comes at the cost of variable signal strengths, interference, and obstruction by non-radio-transparent objects. Types include: terrestrial microwave (up to 45 Mbps), wireless LAN (up to 1 Gbps), wide area (up to 1 Gbps), and satellite (up to 45 Mbps, has either high latency for geosynchronous orbits or higher cost/complexity for low-earth-orbit satellites).

1.3 Internet structure

The internet is assembled from a collection of ISPs of different scales:
A small number of large networks: **Tier-1** commercial ISPs (e.g. BT, Virgin Media, Sprint, AT&T) which provide national and international coverage, and **content provider networks (CDNs)** (e.g. Google, Facebook) - private networks for connecting data centres to the internet, bypassing tier-1 and regional ISPs.
A large number of small networks: various lower tier ISPs, which resell access to some tier 1 ISP hardware or using their own regional networks. ISPs connect to each other at **IXPs (internet exchange points)**

1.4 Delays

Processing delay
When a packet arrives at a node, the header is examined and used to determine where the packet should be directed - this takes some time and is **processing delay**. Processing delays in modern high-speed routers are usually microseconds or less.

Queuing delay
After a packet is processed, it is added to queue to wait for its transmission link to become available. This is dependent on how many packets have arrived before it that are still waiting to be transmitted, if no packets have arrived recently then it will be close to 0, but if traffic is heavy then there may be a long delay. In practice, queuing delays are between microseconds and milliseconds.

Transmission delay
A link between routers will have a transmission rate in (R bits per second). If a packet is of size L bits, then it will take L/R seconds to transmit one full packet. Transmission delays tend to be between microseconds and milliseconds in practice.

Propagation delay
Once a bit is pushed into a link, it must propagate down it to the next router. This is governed by the propagation speed (the speed that the signal can move within the link's medium, e.g. speed of light ($3 * 10^8$ m/s) in fibre lines or the speed of electricity ($2 * 10^8$ m/s) in copper). Propagation delay can be milliseconds in large networks, or nearly negligible in local ones.

Overall delay
If we let d_{proc}, d_{queue}, d_{trans}, d_{prop} be the processing, queuing, transmission, and propagation delays respectively, then the total nodal delay

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop}$$

Queuing delay in more detail
Queuing delay is the most complex part of nodal delay, as it varies packet by packet. This means that we can't discuss absolute values but instead statistical measures of mean/median delays and their distributions, or in probabilities. If we have R = transmission rate, a = packets per second, and L = packet size, then we can calculate the traffic intensity:

$$\text{Traffic intensity } = \frac{La}{R}$$

If $\frac{La}{R} > 1$ then the average rate of bits arriving at the queue exceeds the rate at which they can be transmitted, resulting in the queue increasing forever and the queuing delay approaching infinity. In the case $\frac{La}{R} \leq 1$ the nature of arriving traffic impacts the queuing delay. E.g. if one packet arrives exactly every $\frac{L}{R}$ seconds then each packet will arrive to an empty queue and the will be no queuing delay. If the packets arrive in periodic bursts, e.g. N packets every $\frac{NL}{R}$ seconds, then the first packet will have no queuing delay, the second an $\frac{L}{R}$ second delay, third a $\frac{2L}{R}$ delay, and so on with the nth packet having a $\frac{(n-1)L}{R}$ delay.

2 Protocol layers

Application: works with messages. Is where applications and their protocols exist. Examples include HTTP, SMTP, FTP, and DNS.
Transport: works with segments. Transports application layer messages between specific applications on different hosts. Examples include TCP and UDP.
Network: works with datagrams. Transports segments from one host to another. Examples include IP and ICMP.
Link: works with frames. Transports datagrams between neighbouring network elements. Examples include Ethernet and WiFi.
Physical: works with bits. The physical medium connecting neighbouring network elements. Examples include copper cable, fibre optics, and radio waves.

2.1 Application layer

An application is identified by a 32 bit **IP address** and 16-bit **port number**. The IP address identifies the host, and the port number identifies the application on that host.

2.1.1 The Web and HTTP

HTTP is a stateless protocol which determines how web clients (such as browsers) interact with web servers. HTTP uses TCP, and either establishes one persistent session across many requests, which removes the expensive setup time for each request, or establishes a new session for each request, which prevents the client and server from needing to store information about a session that is not currently in use. When non-persistent sessions are used, each request takes approximately 2 · round trip time + file transmission time, while with persistent sessions once a connection is established, each request takes approximately round trip time + file transmission time.

An **HTTP request** has this format:

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Common request methods are: **GET** (request an object from the server), **POST** (send some data and the server should reply with a response object), **HEAD** (request an object, but the server only responds with the HTTP header, used for debugging), **PUT** (upload an object), **DELETE** (delete an object from the server). **GET** can also send data to the server, by appending queries to the URL, e.g. `somesite.com/search?search=a+search+string`.

An **HTTP response** has this format:

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html
data goes here
```

Common response codes are: **200 OK** (request succeeded), **301 Moved Permanently** (requested object has been moved, the client will automatically request the new URL), **400 Bad Request** (message not understood by server), **404 Not Found** (requested document not found), **505 HTTP Version Not Supported** (server does not support the HTTP protocol version used in the request).

Sessions can be tracked using **cookies** by setting the **Set-Cookie** header in the response, and the client will send the cookie back in the request. Cookies can be used to store session information, user preferences, and shopping cart contents, or they can be IDs that are used to index a cookie database on the server.

HTTP/2 is a new version of HTTP that is binary instead of textual, fully multiplexed, uses one connection for parallelism, uses header compression, and allows servers to push responses proactively into client caches. **Web caches** satisfy HTTP requests on behalf of a server. A browser must be configured to request from a cache, and then if the page is already in the cache it can respond faster than the server could have. Web caches use conditional GETs to check if the page has been updated since the last time it was cached, by setting the `If-Modified-Since` header line.

2.1.2 Email and SMTP

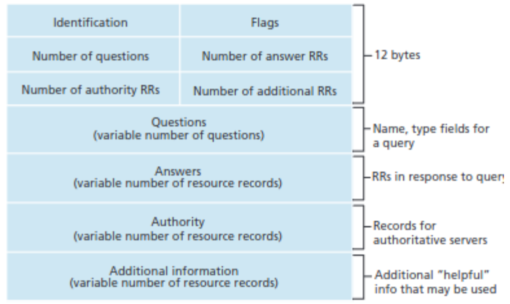
Email systems consist of user agents, mail servers, and the simple mail transfer protocol (STMP). When an email is sent, the user agent sends the email to the user's mail server, which then sends the email to the recipient's mail server, which then stores the email until the recipient's user agent requests it. STMP uses TCP with persistent connections, and pushes the email to the recipient's mail server, unlike HTTP which pulls objects from a server. An email header looks like this:

```
From: alice@aServer.com
To: bob@anotherServer.com
Subject: Hello
```

POP3 is a protocol used by the recipient's mail client to retrieve the email from the server. A user logs in with `user <username>` and `pass <password>`, then can list emails with `list`, retrieve an email with `retr`, and delete an email with `dele`. Once the user has finished, all their actions are applied at the same time with `quit`.
IMAP is a protocol used by the recipient's mail client to retrieve the email from the server. It is more powerful than POP3, as it allows the user to create, delete, and rename mailboxes, and allows the user to search for emails.
Today, most email clients are sent using a web browser, so emails are requested from servers using HTTP.

2.1.3 DNS

The **Domain Name System** is responsible for translating application layer hostnames to network layer IP addresses. It is a distributed database implemented in a hierarchy of DNS servers. When a request is made, the DNS client calls a local DNS server, which then calls a root DNS server, which then calls a top-level domain DNS server, which then calls an authoritative DNS server. The authoritative DNS server then sends the IP address back to the client. The local server will cache the IP address of the response, but will also cache the addresses of the various servers, meaning that the root servers are very rarely contacted. DNS resolution uses both recursive requests (the client asks the server to resolve the request) and iterative requests (the client asks the server to tell it the next server to contact). A hostname may map to many IP addresses, in which case the server responds with them in a random order. As the client usually picks the first one, this acts as a simple load balancing system.
A DNS request has this format:


Identification
Flags
Number of questions
Number of answer RRs
Number of authority RRs
Number of additional RRs
12 bytes
Questions (variable number of questions)
Name, type fields for a query
Answers (variable number of resource records)
RRs in response to quer
Authority (variable number of resource records)
Records for authoritative servers
Additional information (variable number of resource records)
Additional "helpful" info that may be used

And a record has the format `<name, value, type, ttl>`. The **type** states what the record is, `A` means `name` is a hostname and `value` is the corresponding IP address, `NS` means `name` is a domain and `value` is the server knows the next step in resolving that domain, `CNAME` means `name` is an alias for `value`, `MX` means `name` is a domain and `value` is the mail server for that domain. The **TTL** is the time the record can be cached for.

2.1.4 P2P and BitTorrent

Peer-to-peer applications communicate directly with other peers without requiring a central server. P2P applications are used for file distribution, streaming, and telephony, as they are well suited to sharing large files.
If we have a system with N hosts wishing to download an F bit file, peer[i] having an upload rate of u_i and a download rate of d_i, along with a server with an upload rate of u_s, then for a client server architecture the time for all clients to download the file is bounded by

$$D_{CS} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{d_{\min}}\right\}$$

In a P2P system, each peer who gets a part of the file can distribute it onwards, and the download time is bounded by

$$D_{P2P} \geq \max\left\{\frac{F}{u_s}, \frac{F}{d_{\min}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

D_{P2P} equals that only when the server sends bits one at a time to each client.
BitTorrent is a popular P2P file distribution protocol. A file is broken into chunks, typically 256 KB each, and all peers for a file form a torrent. Each torrent has a tracker, which keeps a list of all active peers. When a new peer (we'll call Alice) joins a torrent, it contacts the tracker, which selects a random subset of peers and sends their IPs to Alice who attempts to open concurrent TCP connections with all the IPs it received. Each successful connection is counted as a "neighbouring peer", and over time some of those peers will leave while others join and establish connections, leading to Alice's neighbouring peers fluctuating over time. Every neighbour will have a different subset of chunks of the file, so Alice will periodically request a list of the chunks each of its neighbours has. Alice then determines which chunks are held by fewest of her neighbours and requests them first (rarest-first), this means that each chunk of a file should end up roughly equally distributed.
Alice will also receive requests for chunks for each of her neighbours, and she selects the four that are sending her the highest rate of bits, and reciprocates by sending them their requested chunks - these neighbours are known as **unchoked**. She recalculates the top four every 10 seconds. Every 30 seconds, she also selects one random neighbour (in this case Bob) and sends it chunks - this neighbour is **optimistically unchoked**. If Alice makes it into Bob's top four uploaders and he will reciprocate, and may also make it into Alice's top four. This means that peers will gradually match up with other peers that are capable of uploading at comparable rates, and that a new peer will get chunks that it can trade with its neighbours. All other neighbouring peers receive nothing from Alice, and are **choked**.

2.1.5 QUIC

QUIC is an application layer protocol built on top of UDP, and provides error and congestion control, along with security. Integrating all of these allow it to set up reliability and authentication in 1 RTT, instead of requiring 2 RTTs that are required for TCP + TLS. In addition, QUIC can multiplex multiple streams over a single connection.

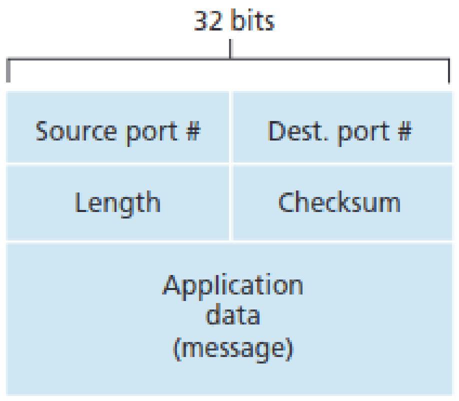
2.2 Transport layer

Multiplexing is the process of taking data from multiple applications and sending it over a single link. UDP uses a tuple of `(destination IP, destination port)` to identify which application the data is for, while TCP uses a tuple of

```
(source IP, source port, destination IP,
destination port)
```

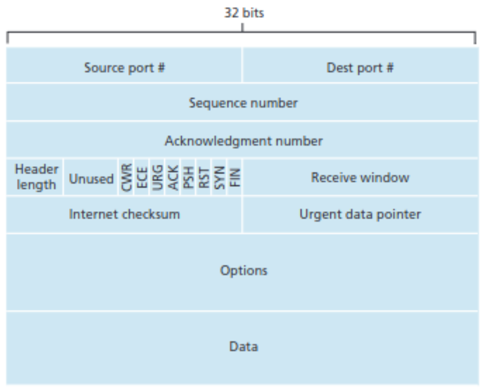
2.2.1 UDP

User Datagram Protocol is a connectionless protocol that simply packages a bytestream into segments and sends it to the destination. It provides no guarantees of delivery, order, or duplicate packets. It performs simple error checking, and is used for applications that can tolerate some loss, such as streaming media, DNS, and SNMP. A UDP segment has this format:



2.2.2 TCP

Transmission Control Protocol provides a delivery service that guarantees that a byte stream sent using it will arrive in order with no duplicate packets. It is a **connection oriented service**, meaning that the client and server exchange transport layer level control information before the application layer messages start to flow. TCP creates a segment up to the **maximum segment size (MSS)** which is determined so that the TCP segment when encapsulated in an IP datagram can fit within the largest link-layer frame that can be sent by the sending host. Ethernet has a **maximum transmission unit (MTU)** of 1500 bytes, and the TCP+IP headers are usually 40 bytes, so the MSS is typically 1460 bytes. A TCP segment has this format:



The **receive window** is the amount of data that the receiver is willing to accept, and is used to prevent the sender from overwhelming the receiver. The **options field** is usually empty, but can be used to negotiate a window scaling factor for for high speed use cases or for timestamping. The 4 bit **header length** is the length of the header in 32 bit words, usually when there are no options it is 5. The flags CWR and ECE are for congestion notification, **URG** is used if there is an urgent pointer in the **urgent data pointer** field, though this is rarely used, **ACK** is set if the segment is an acknowledgement for a successfully received segment, **PSH** is used to push data to the application layer immediately, again rarely used, **RST** is used to reset the connection, **SYN** is used to establish a connection, and **FIN** is used to terminate a connection.

Sequence and acknowledgement numbers: TCP uses cumulative acknowledgement, so the acknowledgement number is the next byte the receiver expects to receive. The sequence number is the byte number of the first byte in the segment. TCP usually, but isn't required to, cache data that is received after a gap in the sequence numbers, and only deliver it to the application layer once the gap is filled.

Retransmission timers: TCP uses a retransmission timer with a variable timeout period. The period is calculated using a sample RTT based off of one currently transmitted but not yet ACKed segment, resulting in a new value approximately once every RTT. This is used to calculate an estimated RTT, using an **exponential weighted moving average (EWMA)** using the following formula:

$$\text{EstimatedRTT} = (1-\alpha)\cdot\text{EstimatedRTT} + \alpha\cdot\text{SampleRTT}$$

The recommended, but not required, value for α is 0.125. The variability of the sample RTT is also calculated, using:

$$\text{DevRTT} = (1-\beta)\cdot\text{DevRTT} + \beta\cdot|\text{SampleRTT}-\text{EstimatedRTT}|$$

If there is little variation of sample RTT values, then DevRTT will be small, while if there is a lot of variability it will be large. The recommended value of β is 0.25.
The timeout value is then set using both these values:

$$\text{TimeoutInterval} = \text{EstimatedRTT} + 4\cdot\text{DevRTT}$$

This means that the timeout stays close to the estimated RTT, but if DevRTT grows then so does the timeout period. An initial TimeoutInterval of 1 second is recommended, and if a timeout occurs then the timeout is doubled until a segment is received, after which the timeout is recalculated using the above equation.

Flow control prevents the sender from overflowing the receive buffer of the receiver. The receiver advertises a **receive window** in each segment, which is the amount of data it is willing to accept. Each time the receiver sends a segment, it includes the current amount of space it has in its receive buffer. The sender uses this value to ensure that the number of unacknowledged bytes in transit is never more than the receive window, ensuring that the receiving buffer never overflows. In the case that the receiver's buffer is full, and it gives the sender a window size of 0, and the receiver has no segments it wants to send, the sender will continue to send 1 data byte segments to the receiver until the receiving application starts reading from the buffer again and the recipient will start responding with non-zero window sizes.

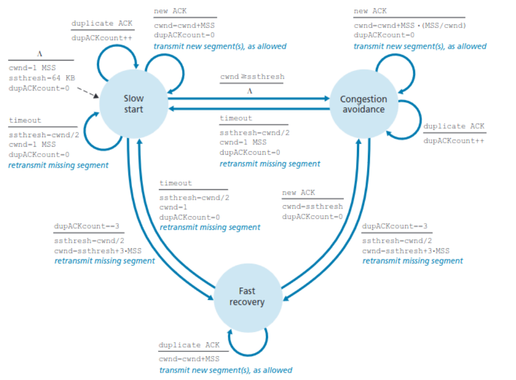
Connection setup: The **three-way handshake** is used to establish a connection. The client sends a segment with the SYN flag set and a randomly chosen initial client sequence number `client_isn` to the server, this is the SYN message. The server responds with a segment with the SYN and ACK bits set, a random initial server sequence number `server_isn`, and the acknowledgement number set to `client_isn+1`, this is the SYN-ACK message. The client then sends a segment with the ACK bit set, the sequence number set to `client_isn+1`, and the acknowledgement number set to `server_isn+1`, this is the ACK message. The connection is now established, and data transfer can begin.

Connection teardown: either party can end the connection at any point by sending a segment with the FIN bit set. The other party responds with an ACK, then its own FIN, which the first party responds to with an ACK. The connection is now closed.

Congestion control: TCP performs congestion control by maintaining a congestion window cwnd alongside the receive window rwnd ant the maximum amount of data in flight at any time is `min(cwnd, rwnd)`. A **loss event** occurs when either a timeout occurs or 3 duplicate ACKs are received. If rwnd is large enough it can be ignored, so we will focus only on cwnd.
The TCP congestion control algorithm consists of 3 parts:

1. **Slow start**: initially, TCP ramps up cwnd, starting from 1 MSS, and increases it by 1 MSS every time an ACK is received. If a loss event occurs, cwnd is set back to 1 MSS and the process repeats, except TCP also maintains a variable `ssthresh` (slow-start threshold), which is set to half the cwnd value at the time the loss event occurred. Once cwnd reaches `ssthresh`, slow start ends and enters congestion avoidance mode.
2. **Congestion avoidance**: now, cwnd is around half the value at which congestion last occurred, so TCP must be more careful. Therefore, cwnd is increased by 1 MSS every RTT, usually by increasing the window by *MSS*/number of bytes in flight for each ACK. If a timeout occurs, the response is the same as in slow start, `ssthresh` is set to half of cwnd, cwnd is set to 1 MSS, and slow start mode is reentered. If 3 duplicate ACKs are received, then `ssthresh` is set to cwnd, and cwnd is halved (+ 3 MSS for the duplicate ACKs received), and it enters fast recovery mode.
3. **Fast recovery**: in this mode, cwnd is increased by 1 MSS for every duplicate ACK received for the missing segment that caused fast recovery to start. Once that segment is ACKed, TCP returns to congestion avoidance with cwnd set to `ssthresh`, returning to the state it was in.

The state machine for TCP congestion control is as follows:

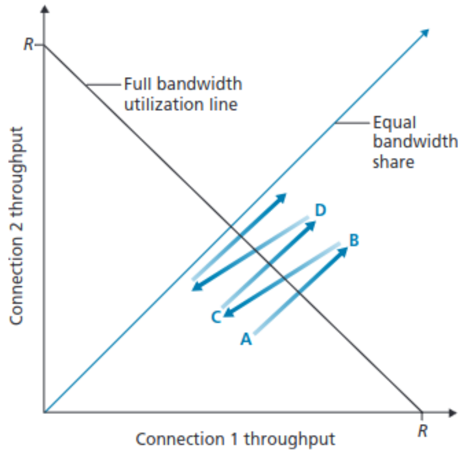


TCP uses **additive increase, multiplicative decrease (AIMD)** for congestion control.
If a router is experiencing congestion, it may set the ECN bit in TCP ACKs passing through it, which causes the receiver to halve its cwnd.

TCP throughput: if w is the window size, and when a loss event occurs the window halves to get W, then the throughput of a TCP connection is given by:

$$\text{Average throughput of a connection} = \frac{0.75\cdot W}{RTT}$$

Fairness: if multiple TCP connections pass through the same bottleneck link, they tend to share the bandwidth fairly if they both have the same RTT. If they don't the connection with the smallest RTT will claim a larger portion of the available connection. This is the mechanism by which it converges on fairness:

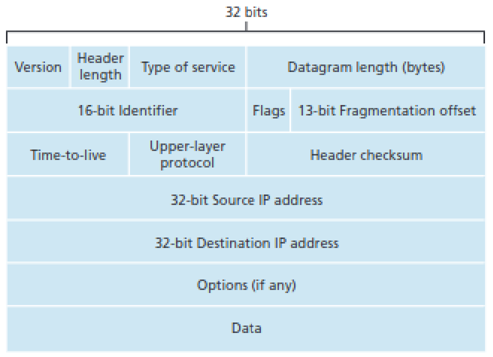


2.3 Network layer

The network layer routes packets between hosts. It runs on every host and router on a network. The network layer consists of two components: the **control plane** determines the best routes across the network, and the **data plane** forwards packets based on those routes. The internet's network layer is the **Internet Protocol (IP)**, which provides only a "best effort service", with no guarantees of packets being delivered.

2.3.1 IPv4

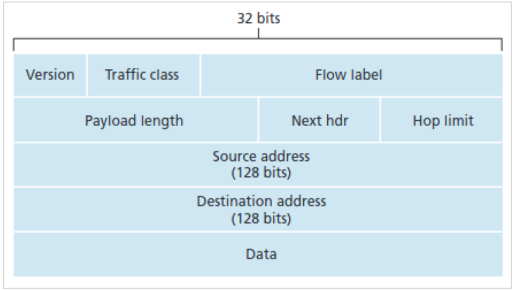
The **IPv4** header has this format:



The minimum header length is 20 bytes, but it may be longer if there are options set. **TTL** is the number of hops until the datagram gets dropped. This is decremented each time a router processes it, preventing a datagram circulating endlessly. **Protocol** is the transport layer protocol that the datagram is carrying, with 6 being TCP, 17 being UDP, and 1 being ICMP. If a datagram travels across a physical link with a smaller MTU than the datagram, it will be fragmented. The **fragment offset** is the offset of the fragment in the original datagram, and the **identifier** is the same for all fragments of a datagram. The **header checksum** is calculated over the header only, and is recalculated at each router due to the TTL changing. **type-of-service bits** allow different types of IP datagrams to be distinguished from each other. This is also where the ECN bits are set
Addressing: every interface between a physical link and a host/router has a 32 bit **IP address**, which is unique across the entire internet. This allows for only around 4 billion IP addresses, so routers use **network address translation (NAT)** to allow multiple devices to share a single IP address by having the router present itself to the outside network as though it is only one device, and each request being sent from a device within the internal network gets assigned an arbitrary port and its sender address changed to be the router's address. When responses arrive, the router uses the destination port to map that message to an internal IP, and sends the packet on to there. A device behind a NAT router will have an internal IP of 10.0.0.x, which is a reserved IP block for internal addresses. A network is divided into **subnets** by dividing it along each router. Every interface in the same subnet has the same subnet mask (prefix), e.g. `223.1.1.0/24`, which means that the first 24 bits are the same for all IPs in that subnet. This means that an external router will only need to consider what subnet a packet needs to be routed to, while an internal router only needs to look at the last bits of the IP address.
DHCP is a method to automatically assign IP addresses to devices on a network. The DHCP address assignment protocol occurs in four steps: 1. **DHCP server discovery**: when a new host connects, it first must find a DHCP server. It does this by sending a UDP packet to port 67 of the IP broadcast address (255.255.255.255), and sets the source address to 0.0.0.0. 2. **DHCP server offer(s)**: when a DHCP server receives a packet on port 67, it then responds with an offer of an IP address, network mask, and a duration for that IP address to remain valid, again sent to the broadcast address this time on port 68. 3. **DHCP request**: the client selects one of the offered IP addresses, and responds to the broadcast address with that IP, as well as echoing back the configuration parameters 4. **DHCP ACK**: the server responds with an ACK message confirming the parameters.

2.3.2 IPv6

An **IPv6** header has this format:



IPv6 uses 128 bit addresses, which allow for 2^{128} addresses. The header is fixed to always be 40 bytes long, so we only need to provide the **payload length**. IPv6 packets cannot be fragmented, instead if an overly large datagram arrives at a router, it drops it and sends a "Packet Too Big" ICMP message back to the sender. There also no checksum, which is left to the link and transport layer protocols. Instead of an options field, the **next header** field can be set to an options field, which contains options then itself states the contained protocol. IPv6 datagrams can be tunnelled through IPv4 if a router doesn't support IPv6, by encapsulating the IPv6 datagram in an IPv4 datagram.

2.3.3 Data plane

A router consists of:
Input ports: physically terminates the incoming physical link, a queue of messages waiting to be routed, and finally performs the lookup to determine where the packet should be routed to. **Switch fabric**: connects all input ports to all output ports, and is used to move a datagram from an input to an output. **Output ports**: contains a queue of messages waiting to be transmitted, and the physical transmitter used to send messages onto the next physical link. (Note that sometimes there will be an input and an output port on a physical link, if the physical link supports 2 way communication). A **routing processor**: performs control plane functions such as maintaining the routing tables.
When an packet arrives at the router, the router table finds the longest matching prefix in the routing table and applies that behaviour to the packet. Packets are switched using one of several approaches:
Switching via memory: the oldest approach. The router is simply a computer, and when a packet arrives it triggers an interrupt, then the routing processor copies the packet to memory, determines how to route that packet, then copies it back out to the output buffer. In this scenario, if the processor has a bandwidth of B bps, then the overall throughput must be less than $B/2$. In addition, only one packet can be routed at a time, as only one memory read or write can occur over the shared system bus. **Switching via a bus**: here, the input port prepends an internal routing label to the packet indicating the correct output link. This is then written to a bus connecting all inputs to all outputs, and only the indicated output stores the packet. Here, the switching speed of the router is limited by the bus speed. **Switching via an interconnection network**: by connecting a network of busses with toggleable connections at each intersection multiple packets can be routed in parallel. In the example below, a packet arriving at B or C could be simultaneously routed to outputs X or Z, alongside the current $A \rightarrow Y$ routing occurring.

2.3.4 Control plane

The control plane determines how packets are routed across the network. There are two approaches: **per router control**: a routing algorithm runs on every router, and so routers talk to each other to determine what the best routes are. **logically centralised control**: a single routing controller determines the best routes, and tells all routers on the network what their forwarding tables should be.
Routing algorithms determine "good" paths from senders to receivers for some definition of "good", generally the shortest or fastest path. Routing algorithms can be **static** if routes are computed once and rarely change, or **dynamic** if routes are recomputed as the network changes. They are **load sensisitive** if they dynamically vary the routes based on the current network load.
A **link-state algorithm** has every router send a message to every other router with the state of its links, and then each router constructs a graph of the network and uses Dijkstra's algorithm to find the shortest path. Each router should advertise its link costs at random intervals to prevent route oscillations. The **distance-vector (DV) algorithm** is iterative, asynchronous, and distributed, and scales better than link state as routers only need to send messages to their neighbours. Each router maintains a distance vector, which is the cost of the shortest path to every other router. The router then sends its distance vector to its neighbours, who then update their distance vectors based on the received vectors. Distance vectors are updated by finding the minimum cost to each destination, based on which neighbour has the lowest cost from the current router to that one added to the cost from that router to the destination based on the received distance vector. The **Bellman-Ford equation** is used to update the distance vector, and is:

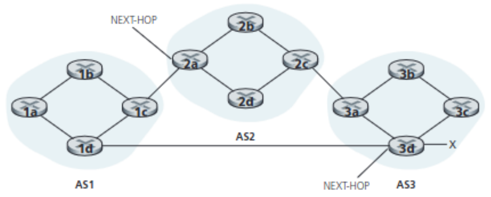
$$d_x(y) = \min_v\{c(x, v) + d_v(y)\}$$

It is possible for loops to form if link costs increase, where one node x routes through y to get to z, while y routes through x to get to z. In the above case, when $c(x, y)$ goes up to 60, then y recalculates its shortest path to 6, but this is based on z saying it has a $c(z, x) = 5$ path available. z then gets this update, and recalculates $c(z, x)$ to 7, and so on until the costs have counted up to be more than 50 (44 iterations).

This is known as the **count-to-infinity problem**. This can be partially fixed by having z lie to y and say it has no route to x because its route to x goes via y (the **poisoned reverse**). This is not perfect though, as larger loops can still form.
DV and LS algorithms differ on a number of key metrics: **Message complexity**: LS requires each node to be sent the cost of every link in the network, or $O(|N| + |E|)$ messages, and when a link cost changes this must all be sent again. DV only sends messages between neighbours at each iteration, and these messages need only to be sent after a link cost changes, significantly reducing the number of messages needed. **Speed of convergence**: LS takes $O(|N|^2)$ to calculate all routes, while DV converges slowly and can have routing loops until the network converges. **Robustness**: single corrupt routers can influence DV more significantly, as every router depends on every other routers' communication, causing incorrect values to diffuse through the whole network.

2.3.5 Large scale routing

Routers are organised into **autonomous systems (ASes)**, with every router within an AS running the same routing algorithm, and one global routing protocol managing all inter-autonomous system routing.
Open shortest path first (OSPF) is an intra-AS routing algorithm which uses link-state information flooding and Dijkstra's algorithm. Individual link costs must be configured by the system administrator, giving them control over the routed flows. Advantages include: **security**: link state updates may be authenticated, ensuring that only trusted routers can participate in the OSPF protocol. **Multiple same-cost paths**: when multiple paths to a destination have the same cost, then both may be used. **Hierarchy within ASes**: an OSPF AS can be broken down into smaller hierarchical areas, with each running its own OSPF link-state algorithm.
The internets inter-AS routing system is the **border gateway protocol (BGP)**. BGP uses routing tables in the format `(x,I)`, where `x` is a prefix and `I` is one of the router's interfaces. BGP determines prefix reachability by having each router broadcast all the prefixes that are reachable from itself, and the path from it to those prefixes. Using this example network:



Router 3a would send an **external BGP (eBGP)** message to 2c saying `AS3 x` (to reach subnet x go via AS3). 2c would then send **internal BGP (iBGP)** messages to all the other routers in AS2 to tell them to go via AS3. 2a would then send a eBGP message to 1c, which would tell its AS there is a path to subnet x along `AS2 AS3 x`. In addition, there would be a path `AS3 x` from 1d. If there are multiple shortest paths, then the route is filtered by a manually set local preference, then the shortest AS path, then the closest exit point (**hot potato routing**).
Routing policy may be set so that an AS won't transport packets that came from another AS unless the companies managing those ASes have a peering agreement, or to prevent undesired through traffic for ISPs who peer with multiple other ISPs.
BGP allows for IP anycast, where if multiple servers have the same IP address, BGP routing will ensure that a request goes to one of the nearer servers. This is fine for stateless protocols, but can cause issues for stateful protocols.

2.3.6 Software-defined networking (SDN)

SDN uses a logically centralised controller to manage the network. The controller communicates with the routers using the **OpenFlow** protocol, which allows the controller to determine the forwarding tables of the routers. This allows for more flexible and dynamic routing, and allows for easier network management. The routers have a flow table, which can match a number of fields, then perform certain actions, including: **forwarding** the packet to specific ports or the controller, **dropping** the packet, or **modifying fields** in the packet.

2.4 Link layer

The link layer transports data between neighbouring nodes on the network. The link layer protocol is generally implemented in a **network interface card (NIC)**, with a single special purpose chip that implements the protocol in hardware, along with software drivers to interface with the card.
The link layer provides **error detection and correction**.