1 Networks and OSI Layers

1.1 History

ARPANET: Created in 1969-1990, mainly had 4 sites for universities. Developed TCP/IP.

International Network Working Group: 1972, proposed packet switching (ie. splitting data into packets and then sending them independently). Rejected for international standardisation.

OSI Model: Competitor to ARPANET group, published as an international standard in 1984 however slow development stopped it from taking hold.

NSFNet: Created in 1986 to provide researches access to sites in USA.

CERN: Developed their own TCP/IP based network, which would lead to the creation of the world wide web. Standardised TCP/IP as the protocol stack of choice, and only used OSI as the name for protocols rather than the protocols themselves.

Internet Standardization: Designed without consideration for an adversary on the network, however difficult to make changes on everything that uses TCP/IP so some protocols may still be insecure eg. DNS. Additionally, due to OSI and TCP rivalry, other parties may reject perfectly good solutions simply because the other party came up with it.

TCP/IP: Main protocol stack used. It implements a protocol then standardizes it.

OSI: Used as an abstraction or model for TCP/IP now. First decides the function then implements it.

Models: Benefits of having a model.

- Interoperability: Allows different stakeholders to communicate with each other. Eg. Google browser interacting with a web application.
- Developing: Allows a reference model to develop with, or expected interfaces so that we can abstract concepts. Eg. No matter what form of communication, we know that there should be a critical layer, even if we don't know how it is carried out.
- Reference model can simplify the design process (again by having an interface to abstract complications)
- Also good engineering practice to have an abstract reference model, a reference model and corresponding implementations for validation purposes.

1.2 Network Models

Network Model: Models the network as a stack of layers. *It is just a model.*

Layers: Each layer offers services to the layer above it. Layers can communicate between each other, eg. passing information up or down.

Physical Layer: The very bottom of the network model. It tends to be a wire or something else physical that allows communication.

Protocol: Allows inter-layer communication and exchanges. Communicates between objects on the same layer. They are also a set of rules which govern the format and meaning of packets that are exchanged by peers within a layer. Eg. send a packet, resend etc.

Services: Interfaces between layers, it provides a service to the layer above it.

TCP: Connection oriented service. It has functions connect, use and disconnect. Similar to a telephone service in that it has to call, receiver responds and once the phone call is over it disconnects.

UDP: Connectionless oriented service. It passes a message by routing it through intermediate nodes. It is similar to sending mail or a text message in that it doesn't require a connection from the other end to send and doesn't have to disconnect after sending. Once a packet is sent, the protocol is complete. Both protocols has its own benefits and drawbacks.

1.3 OSI Model

OSI Model:

- A layer should be created where a different abstraction is needed.
- Each layer should perform a well defined function.
- The function of each layer should be chosen with a view toward defining internationally standardized protocols.
- The layer boundaries should be chosen to minimise the information flow across interfaces (encapsulation, all necessary data and its functions should generally be in the same layer)
- The number of layers should be large enough that the distinct functions can be separated, but small enough to not over complicate the architecture.

Application Layer (7): Interprets packets, deals with things such as http, web and file downloads. Can access all the layers below it.

Presentation Layer (6): Deals with the presentation, such as choosing whether to use ASCII or unicode for text.

Session Layer (5): Not commonly used.

Transport Layer (4): Ensures reliable transport for end to end protocols(travels between the first and last objects in a layer).

Network Layer (3): Able to work point to point (objects next to each other in a layer) and end to end. It also names and identifies network entities.

Datalink Layer (2): Send frames (packets for the datalink layer). It can turn a stream of bits into packets/frames, do bit verification (and ask again to validate bits). It functions through the point to point protocol, however packets goes down through the physical link and not across layers.

Physical Layer (1): Anything physical that is able to communicate information from one location to another.

TCP/IP Model: Has 4 layers. Application, Transport, Internet (similar to network) and Host-to-Network (combination of Data link and Physical layers).

Using protocols: Encapsulates data at each stage, and adds headers depending on the layer. Passing messages between locations involves wrapping and unwrapping those layers. Similar to a birthday package being sent, where at each layer different information is added.

2 The Application Layer

2.1 HTTP and HTML

Hypertext: Creation and use of linked content. Eg. a page linking to another page.

Client: Provides browser based access to pages.

Server: Daemon based (continuously running) content delivery of pages.

URL: Uniform Resource Locator, has the protocol, DNS name and file name. Also can be known as URI/IRI but they have differences. Provides and address for a resource, and can be absolute or relative. Also can take arguments which are passed in through the URL, isch as path, query, fragment etc.

HTTP: HyperText Transfer Protocol. Defines everything needed for a web.

HTTP Protocol Overview:

- Client initiates TCP Connection to a server (a socket is created)
- Server accepts TCP Connection from client
- HTTP messages are exchanged between the browser (HTTP client) and Web server (HTTP server). For example a file, or HTML document.
- TCP connection is closed.

Non-Persistent Connection: Requires 2 'response times', one for TCP connection and one for a HTTP request for an object as well as the file transmission time. Has to open and close a connection for every object we want to download/access.

Persistent Connection: The server will leave the connection open after sending a response. This allows us to reuse the connection for subsequent messages and requests. It also has the added benefit of being able to send a request as soon as it need to, rather than waiting for the previous request to finish (pipelined).

HTTP Steps:

1. Browser determines the URL
2. Browser makes a TCP connection
3. Send the HTTP request for the page
4. Server sends back page as the HTTP response
5. Browser fetches other URLS as required
6. Browser displays the page progressively (as content arrives)
7. TCP connections are then released

HTTP Request methods:

HTTP Method	Safe	Idempotent	Cacheable
GET	Yes	Yes	Yes
HEAD	Yes	Yes	Yes
POST	No	No	Yes/No
PUT	No	Yes	No
DELETE	No	Yes	No
CONNECT	No	No	No
OPTIONS	Yes	Yes	No
TRACE	Yes	Yes	No
PATCH	No	No	No

Idempotent: Multiple identical requests have the same effect.

Safe: Only for information retrieval, calling this method should not alter the state.

HTTP Response Codes:

Code	Meaning	Example
1xx	Information	100 - Server agrees to handle client's request
2xx	Success	200 - Request succeeded; 204 = no content present
3xx	Redirection	301 - Page moved; 304 = cached page still valid
4xx	Client Error	403 = Forbidden page; 404 = page not found
5xx	Server Error	500 = Internal server error; 503 = try again later

HTTP Headers: Has a lot of examples, but they provide additional information to HTTP requests. Examples are cookies, expiration, and if the page has been modified since a date.

Client Side Processing: Has plugins and extensions which execute inside the browser, such as Adblock. It can also have helpers which is a separate program that is instantiated in the broswer, but access the local cache (your computer). An example is a word file redirecting to open on your machine.

Static Pages:

- Accept TCP Connection from client (browser)
- Identify the file requested
- Get the specified file from the local storage (disk, RAM)
- Send file to client
- Release the TCP Connection

Multithreaded Web Server: Serves dynamic pages. It has a front end which redirects requests to different modules or servers.

Dynamic Pages:

- Resolving name of Web page requested
- Perform access control on the Web page
- Check the cache
- Fetch the requested page from disk or run program
- Determine the rest of the response
- Return the response to the client
- Make an entry in the server log

Web Cache: Can return data without making a request to server if it is in the cache. Also does basic validation such as making sure it hasn't been modified or the data is not too old.

Web Proxy: Used for caching, security and IP address sharing. Can limit request types, and uses the cache to return data instead of making lots of requests to commonly referenced servers (such as Google).

Cookies: Stores a small amount of information on the users computer, and passes that information back to the front end (as front end doesn't store information about sessions).

HTML: Hypertext Markup Language, responsible for formatting and contents of a page.

Dynamic Content: Can be done client side or server side by running scripts. Java, Javascript, PHP.

2.2 DNS, Mail and Streaming

DNS: Domain Name System. DNS is behind the technology that maps a domain to an IP address. It comprises of 4 elements.

1. Domain Name Space: DNS uses a tree structured name space to identify resources on the internet
2. DNS Database: Each node/leaf in the name space tree has a set of information that is contained in a resource record (RR). The collection of RRs is organized into a distributed database.
3. Name Servers: Sever programs that hold information about a portion of the domain name tree structure and the associated RRs.
4. Resolvers: Programs that extract information from name servers in response to client requests.

IP Address: Unique numerical identifiers, similar to a phone number where the DNS can be considered as an address book. Conceptually, IP addresses should only map to one socket/jack on a computer, however it is often not the case.

Domain Names: Case insensitive, can have up to 63 chars per constituent (dot), and up to a total of 255 chars per path. It can also be internationalised, however using unicode can cause security problems for humans (such as an * in unicode and ASCII looks the same to humans, but is interpreted differently on a computer which makes issues such as spoofing arise).

Absolute Domain Names: Ends in a '.' which denotes the root of the tree. Eg. '.com.' shows that com is at the top of the tree.

Relative Domain Names: Ends in a constituent. Means that the address is relative to something (most often a database). Eg. '.com or .com.au'

Top Level Domains: Typically used for a purpose, however later expanded to be more commercialised. Eg. 'edu' for educational institution which is restricted, and 'net' for network providers which is not restricted so allows companies like ABC to have a .net domain.

Resource Records: What is stored in the database.

Type	Meaning	Value
SOA	Start of Authority	Parameters such as primary name server and the responsible party for this domain
A	IPv4 address of a host	32 bit integer that stores IP address associated with the domain
AAAA	IPv6 address of a host	128 bit integer for IP (better than IPv4 but not as widely used yet)
MX	Mail exchange	Priority (1 - access first, 2- access next), domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name, where the address should redirect to
PTR	Pointer	Alias for an IP Address, similar to CNAME but can have multiple eg. pointer pointing to other pointers
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides the service, name of services required
TXT	Text	Descriptive ASCII Test, can be used for guide or documentation that is retrieved from database

DNS Registrar: Used to register a name or domain. Need to provide names, IP addresses of name server etc, and the registrar inserts two RR into the server.

Name Server Zones: DNS namespace is divided into overlapping zones. They are arranged in a hierarchical manner extending from a root server, and can be responsible from controlling domains in their zones, or passing it onto another zone.

Root Name Servers: Forms the authoritative cluster for enquiries. The root servers are contacted by a local name server that cannot resolve names.

Top Level Domain DNS Servers: Level down from the root server. Responsible for com, org, net, edu etc as well as country domains like uk, au,jp.

Authoritative DNS Servers: Organizations DNS servers, providing authoritative hostname to IP mappings for organizations servers (eg. Web, email). Can be maintained by the organization or a separate service provider.

Local DNS Server: Typically each ISP has a default name server which handles DNS queries. If a value exists in the cache, it will return it, otherwise moves up and down the query hierarchy to get an answer.

Resolving a Query: A recursive query mode is used. If an answer is unknown, it will query up the hierarchy to a root, which then queries down until it is resolved. Queries are also subject to times to avoid long response times. Eg. local $\rightarrow$ auth server $\rightarrow$ root server $\rightarrow$ response: edu $\rightarrow$ queries edu, response: unimelb $\rightarrow$ sends response back down to local.

HOSTS File: Has hard code mappings to IP addresses. Useful for localhost on a computer, or using a fake address for ad blocking.

DNS Spoofing: Impersonating other websites.

DNS Flooding: Making small requests to a website, which then sends back a large amount of info to someone else.

SMTP: Simple Mail Transfer Protocol. Uses TCP to reliable transfer email messages from a client to server. Has three phases, handshake, transferring messages then closing the connection. Also limited to 7-bit ASCII.

User Agent: Allows the basic functions of composing, reporting, displaying and deleting emails. It also encapsulates transport related information in the headers of an email. It has a body which is where the message is sent. User needs to provide message and destination.

MIME: Multipurpose Internet Mail Extensions. An addition to SMTP that adds additional message headers, extending the functionality. Has MIME version, content description, id, transfer encoding and content type.

MIME Content Types: The type and subtype of the content allows MIME to know how to interpret files. Eg. images have gif, jpeg, png formats.

Message Access: Where SMTP deals with the sending of mail to a server, we need another protocol to read the email. Examples are local mail, POP3, IMAP and HTTP.

Remote Mail: Where local mail can be sent easily as the connection will always be open, remote mail doesn't have that same guarantee. As it can have intermittent connection or may not be open at the time of sending, we need a different protocol to handling receiving and reading mail.

POP3 Post Office Protocol, it has three states. The main issue with this protocol is that once a message is read, it can no longer be reread as it deletes once done.

- Authorization: `USER/PASS` which authorizes the user
- Transactions: `LIST` which shows mail
- Update: `RETR/DELE` which retrieves, reads and then deletes mail.
- `QUIT` which is part of the update state, but ends the connection

IMAP: Internet Message Access Protocol. The main benefit of this is that it keeps user state across sessions, which retains mailbox contents. It however requires more computing power which limits its use.

Streaming: Uses websockets over HTTP over TCP for streams. It makes a request once, and receives data in a stream of data packets.

RTCP/RTSP: Real Time [Control/Streaming] Protocol. Packets are sent with the correct spacing times needed to render the playback correctly. RTCP can delay and adapt video coding rate to the capacity, whereas RTSP provides play/record/pause services.

3 Transmission Control Protocol (TCP)

Transport Layer: Provides services to the Application (or Session) layer using services from the Network layer.

Transport Service Primitives: Basic functions used in the transport layer.

Primitive	Packet Sent	Meaning
LISTEN	(None)	Block until something tries to connect
CONNECT	CONNECTION REQ	Actively attempts to establish a connection
SEND	DATA	Send information
RECEIVE	(None)	Block until DATA packet arrives
DISCONNECT	DISCONNECTION REQ	This side wants to release the connection

TCP: TCP provides a protocol by which applications can transmit IP datagrams within a connection-oriented framework, thus increasing reliability.

- TCP transport entity manages TCP streams and interfaces to the IP layer
- TCP entity accepts user data streams, segments them into chunks and then sends them as a seperate datagram. Each datagram has an IP and a TCP header (containing information like order of bytes).
- The recipient TCP entities reconstruct the original byte streams from the encapsulation. From the datagrams it receives, it will remove duplicates and piece together back the bytes from the order (labelled in datagram).

TCP - Service Model: Both the sender and receiver creates sockets. For a TCP service to be activated, connections must explicitly be established between a socket at the sending host, and a socket at a receiving host.

Socket: A socket is a kernel data structure, which is named by the *5-tuple* of the IP address and port number of both the sender and receiver, as well as the protocol.

(Sender IP, Sender Port, Receiver IP, Receiver Port, Protocol)

TCP- Connections:

- Full duplex: Receives data in both directions simultaneously.
  (Half duplex is when one side sends data while the other only receives)
- End to end: Exact pairs of senders and receivers (i.e nothing in between)
- Byte streams: Message boundaries are not preserved, unlike message streams which mark things like header etc. The user or recipient of the stream is responsible for interpreting the boundaries (eg. end of message or end of header).
- Buffer capable: TCP entity can choose to wait and buffer data before sending it. Eg. If we have 1 byte but require a 40 byte header, then we might want to wait for more data to come before sending it.
- **PUSH flag**: Indicates that a transmission not be delayed and should interrupt the receiver.
- **URGENT flag**: Indicates that a transmission should be sent immediately (given priority above data in progress) and that the receiver should send it to the application out-of-band (outside of the byte stream). This is usually used as an alarm.

TCP - Properties:

- Data is exchanged between TCP entities in segments (packets for layer 4). Each segment has a 20-60 byte header, plus zero or more data bytes. Zero data bytes is used for an ACK segment, where we have only a header and an empty body.
- TCP decides on the size of the segment, given two constraints. The IP payload must be under 65,156 bytes, and the MTU (Maximum Transfer Unit) is generally 1500 bytes.

$$1500 \text{ bytes} < \text{Payload} < 65{,}156 \text{ bytes}$$

- Sliding window protocol: A protocol used to ensure reliable data delivery without overloading the receiver. Controls how much data is being sent at a time as well as the order it is sent in.

TCP - Header:

Name	Description
Source port	Sending port
Destination port	Receiving port
Sequence Number	If **SYN** = 1: Initial sequence number If **SYN** = 0: Accumulated sequence number of the first data byte of this segment (part of packet or stream)
Acknowledgement Number	If **ACK** = 1: The next sequence number that the sender of the **ACK** is expecting.
Data Offset	Size of the TCP Header (20-60 bytes)
Flags	Single bit flags (**SYN, ACK, RST, FIN** etc)
Window Size	Size of the receive window: How much data the sender of this segment is willing to receive.

Reliable Connections: We want to ensure that only one connection is established, even if some set-up packets get lost. We also want to establish initial sequence numbers for sliding windows (negotiate a random starting point, so that old data streams don't interfere with new ones).

Three Way Handshake: A solution that aims to achieve the establishment of reliable connections. Both the sender and receiver exchange information about which sequencing strategy each will use, and agree on it before transmitting data.

- **Normal Operation**

  1. Host 1 sends a connection request SYN(SEQ = x) to Host 2
  2. Host 2 receives the request and responds with SYN(SEQ = y, ACK = x + 1) which sends its own sequence number and acknowledges Host 1's sequence number
  3. Host 1 responds with (SEQ = x + 1, ACK = y + 1), incrementing its own sequence number after sending data and acknowledging Y's request.

- **Simultaneous Connection Attempts**

  1. Host 1 sends SYN(SEQ = x)
  2. Host 2 sends SYN(SEQ = y)
  3. Host 2 receives Host 1's initial connection request and responds with SYN(SEQ = y, ACK = x + 1)
  4. Host 1 receives Host 2's initial connection request and responds with SYN(SEQ = x, ACK = y + 1)
  5. Host 1 and host 2 have now agreed on respective sequence numbers and can start sending data.

Synchronisation: SYN is used for synchronization during connection establishment. Sending a SIN or FIN causes the sequence number to be incremented by 1.

- Connection request has SYN = 1, ACK = 0
- Connection reply has SYN = 1, ACK = 1
- SYN is used in both the connection *request* and the connection *reply*. The ACK bit distinguishes between the two.

Sequence Number: The first byte of the segment's payload.

Acknowledgement Number: The next byte the sender expects to receive. If there is a missing segment, it will stop incrementing, even if the later segments have been received.

TCP - Retransmission: Each segment has an associated retransmission timer (RTO). If a packet is lost, it will retransmit until received or the connection ends. If the timer expires before an ACK is received, the segment is resent.

Fast Retransmission: When the receiver receives a segment with a sequence number higher than expected, it will send the ACK of the number it is expecting (will not increment even if it has received a higher numbered segment). This will be a duplicate of the previously sent acknowledgement (DupACK). After receiving 3 DupACKs, the sender resends the lost segment.

TCP - Closing: The FIN flag is used to signify a request to close a connection. Each FIN is directional, so once it is sent and acknowledged no further data can be sent from the sender to the receiver. The sender of the FIN can however, continue to receive data.

1. Host 1 sends a FIN request
2. Host 2 sends an ACK of the FIN request, and sends its own FIN request
3. Host 1 responds with an ACK of Host 2's FIN request
4. Connection is closed. (Needs 1 FIN and 1 ACK for each direction)

TCP - Hard Close: A RST flag can be used to signify a hard close of a connection. It basically states that the sender is closing the connection, and will not listen for any further messages. It can be used to close a connection, however FIN is preferred as it allows cleanup of other packets and has an order. RST is usually used as a respond to invalid data or some other error.

3.1 Sockets

Purpose: A process uses a socket to send and receive data. Sockets are used as the 'doorway' leading in/out of the application.

Format: The address of a socket is the 5-tuple consisting of the (Protocol, Source IP, Source Port Number, Destination IP, Destination Port Number).

Interface: Allows sockets to be ported from one OS to another. The format of a socket is similar to reading or writing to a file, and the name given to a socket is a file descriptor. The API is also implemented as system calls eg. `connect(), read(), write(), close()`

Socket Primitives:

State	Description
SOCKET	Creates a new communication endpoint
BIND	Associate a local address with a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Passively establish an incoming connection (block until then)
CONNECT	Actively attempt to establish a connection
SEND	Send some data over a connection (`write()`)
RECEIVE	Receive some data from the connection (`read()`) May need to be called multiple times until all data is received
CLOSE	Release the connection

Socket States:

State	Simplified Name	Description
CLOSED	Idle	No connection is active or pending
LISTEN	Pass. est.	The server is waiting for an incoming call
SYN RCVD	Pass. est.	A connection request has arrived; wait for ACK
SYN SENT	Act. est.	The application has started to open a connection
ESTABLISHED	Established	The normal data transfer state
FIN WAIT 1	Act. disc.	The application has said it is finished
FIN WAIT 2	Act. disc.	The other side has agreed to release
TIME WAIT	Act. disc.	Waiting for all packets to die off
CLOSING	Act. disc.	Both sides have tried to close simultaneously
CLOSE WAIT	Pass. disc.	The other side has initiated a release
LAST ACK	Pass. disc.	Wait for all packets to die off

Using Sockets in C:

- `bind()`: Binds a socket to an address
- `listen()`: Waits for a connection
- `accept()`: Accepts a connection
- `read()`: Reads in data
- `write()`: Sends data
- `close()`: Closes connection
- `select()`, `pselect()`, `poll()`: Waits until one of several files is ready to read/write

3.2 TCP Sliding Window

Sliding Window Size: Controlled by the receiver, and determines the amount of data the receiver is able to accept. Both the sender and receiver independently of the application.

Process: See diagram in Week 8 L2, Slide 19.

1. Sender sends 2KB of data, SEQ = 0
2. Receiver has 2KB of data in a buffer (4KB). Receiver sends back ACK = 2048 and WIN = 2048
3. Sender sends another 2KB packet, SEQ = 2048
4. Receiver now has 4KB in buffer. Receiver sends back ACK = 4096, WIN = 0
5. Sender is blocked from sending as WIN = 0 (no space in buffer)
6. Receiver processes packet 1 and sends ACK = 4096, WIN = 2048
7. Sender is unblocked from sending and can now send more packets.

Blocking: When the window is 0, the sender is blocked from sending data. They can however send URGENT data or a 'zero window probe'.

Zero Window Probe: A 0 byte segment that causes the receiver to re-announce the next expected byte and the window size. This was designed to prevent a deadlock (where buffer has since been empties, but sender cannot send anything to check).

Send Window: What data the sender is able to send, unacknowledged segments and unsent data that can fit into the receive window.

Receive Window: Amount of data the receiver is willing to receive, the window size in ACK.

Window Format: See diagram in Week 8 L2, Slide 33.

- Before Window: Bytes acknowledged. Consists of data that has already been sent and successfully acknowledged.
- In Window: Bytes in flight or bytes the receiver is willing to receive. The sequence of bytes that are currently being sent to the receiver, or being prepared to be sent next.
- After Window: Bytes that the receiver is not ready to receive. Falls outside the size of the window, so we have to wait for the window to slide up before we can start sending them.

Segment Loss: In a sliding window, if a segment is lost, then we can use fast transmission (3 DupACKS) to resend.

Go-back-N: A type of flow control for missing packets. When a packet is lost, go back to the point it was transmitted, and restransmit everything from that point onwards.

Issues: Resends packets that have already been sent
Advantages: Receiver doesn't need to reorder packets, and store out of order packets.

Selective Repeat: Another type of flow control for missing packets. It only retransmits the lost packet.

Issues: Packets arrive out of order, and the receiver has to store and reorder them before forwarding to the application.
Advantages: No unnecessary packets are retransmitted. Only helpful if loss is common.

Persist Timer: Prevents deadlocks from occurring. A timer is started when the window = 0, if by the time the timer = 0, and the window is still 0, it will send a ZeroWindowProbe.

ZeroWindowProbe: Sender asking the receiver to report back its buffer size. Returns ZeroWindowProbeACK, ACK = last packet sent, Window: current window size.

Congestion Control: When networks are overloaded, congestion occurs and can potentially affect all layers. TCP's methods reduces the rate that data is sent and receives, and helps to reduce congestion.

Congestion Control Window (CWND: Provides an additional window that is dynamically adjusted based on network performance to aid efficient transfer. If the network traffic is high, the window is decreased, and if it is low, the window is increased. The congestion window size is maintained by the sender (the sliding window is controlled by receiver but used by sender).

Incremental Congestion Control: Has a slow start to packet sending rate, which later grows exponentially.

1. To start, CWND = 2 * max segment size
2. If segment is acknowledged, CWND++ (adds another window)
3. Each window full of acknowledgements doubles the congestion window
4. Keeps incrementing until it has reached a timeout or a threshold. (Returns to Step 1 after)
5. Eg. Start of with 1 request, which sends 1 packet.
   Returns 1 ACK, which now sends 2 packets.
   2 ACKS are returned, which now send 4 packets each.
   8 ACKS are returned, which continue to send 4 packets (reached max)

Congestion Control Optimization: Similar principle to incremental congestion control, however alters how it deals with reaching a threshold. Rather than reducing the rate back down to 1 after hitting the max rate, we can reduce it by half of the threshold instead. This allows us to skip the slow start phase. This also uses DupACKs for a fast recovery.

Selective Acknowledgements (SACK): Provides greater ability to track segments in flight, allowing up to 3 ranges of bytes received to be specified. Eg. ACK: 1, SACK: 6, 2-4. From the data given, we know we need to retransmit packets 2 and 5.

Explicit Congestion Notification (ECN): Allows IP layer to indicate that congestion is occurring without dropping the segment by setting the ECN flag. It will send a notification through the flag before dropping a packet, and lets the router know to slow down. Sender acknowledges this by setting the CWR flag (Congestion Window Reduced Flag) and reacts as if a segment has been lost, without actually losing a segment.

Equations for Windows:

- W increases once per window

$$W+ = \frac{1}{W}$$

- When a loss occurs, W is halved

$$W- = \frac{W}{2}$$

- Average increase in window size, where p = probability of packet loss

$$\frac{(1-p)}{W} - p\frac{W}{2}$$

- Equilibrium (Equating average to 0)

$$W \approx \sqrt{\frac{2}{p}}$$

- Rate given Round Trip Time (RTT) = T

$$\frac{W}{T} \approx \frac{1}{T}\sqrt{\frac{2}{p}}$$

Macroscopic Model: From the generalised equations above, we can make two insights.

1. RTT unfairness can arise. For a given packet loss rate, longer RTTs get less rate. I.e if p is constant, then if T is large, the overall rate for more trips is slower.
2. If RTT is small, then TCP forces tha packet loss rate to be high. I.e if RTT is fixed and small, then to maintain the same rate of $\frac{W}{T}$, the probability loss p will have to increase.

4 Presentation, Session and Transport Layers

4.1 Presentation Layer

Services:

- Encryption
- Compression
- Data conversion (from CR/LF to LF, .doc to .docx)
- Mapping between character sets (ASCII etc)

Presentation Layer: Not really widely used, a lot of its previous services have been offset to the application layer as it is simple to implement, are not common (required by all applications) and the application layer is not in the kernel.

4.2 Session Layer

Services:

- Authentication
- Authorization
- Session restoration (between log off, websites etc)

Examples:

- Remote procedure call (RPC)
- Point to point tunnelling protocol (PPTP)
- Password Authentication protocol (PAP/EAP)

4.3 Transport Layer

Role: Provides services needed by applications, using the services made available by the network layer. Essentially creates an interface between the application layer and network/internet layer. Currently the network provides the ability to deliver packets from host to host, but has no guarantees on the reliability.

Application Requirements:

- Data as a stream of bytes
- Data from one application is not mixed with that for another
- Data arrives reliably, or we know when a packet has been lost
- Data arrives in order
- Data doesn't arrive faster than we can handle

Connection Oriented Communication: Both sides establish a connection, can send data, then close the connection. Eg. TCP. Similar to a phone call, in that someone dials and other end picks up and they have to actively end the call.

Connectionless Communication: Data is transferred from a sender to a receiver. Eg. UDP. Similar to a text message, where there is data sent to a destination, however the destination is not required to initiate anything to receive the data.

Encapsulation: Abstract representation of messages sent to and from transport entities. Each layer adds a header and wraps the data. Eg. Can start off with data in transport layer, where the transport layer adds a header. Later it moves up to the network layer, which wraps the data given from the transport layer and adds a network header.

Terminology: Not universal, but at least used for this subject.

- Segments: Sent at transport layer (TCP)
- Datagrams: Sent at transport layer (UDP)
- Packets: Sent at the internet/network layer
- Frames: Sent at the link/data link layer

Reliable connection orientated service: Want to provide a perfect connection between two nodes, that hides ACKS, congestion control and lost packets. For higher layers, they just receive the data in order.

Unreliable connection-less orientated service: Provides multiplexing between different processes (has a different layer for addressing processes)

Addressing: Specifies the location where it should connect to. The full address is the 5-tuple.

Port Allocations: Ranges from 0 - 65525. Comprises of well-known ports (0-1023), registered ports (1024-49151) and dynamic ports (49152 - 65535).

Multiplexing: Combining multiple distinct streams into a single shared stream. Can be shortened to 'MUXING'. Many inputs $\rightarrow$ 1 input

Demultiplexing: Splitting distinct streams out from a single shared stream. Can be shorted to 'DEMUXING'. 1 input $\rightarrow$ many inputs.

5 User Datagram Protocol (UDP)

User Datagram Protocol (UDP): Provides a protocol whereby applications can transmit encapsulated IP datagrams without establishing a connection. UDP transmits in segments consisting of a header followed by the payload.

Advantages: Unlike raw IP, UDP allows us to specify ports for source and destination pairs. Requires both the source and destination ports, the source allows it to send replies, and the destination allows it to route to where it should end up.

Strengths: Allows for multiplexing and demultiplexing, and has no delay in recovering lost packets.

Weaknesses: No flow control, error control, or retransmission of bad segments.

Uses: Used where applications require a precise level of control over packet flow/error/timing. Otherwise TCP takes care of a lot of these for you.

UDP Header: Contains a source port, destination port, UDP length and UDP checksum. The checksum can tell us if a packet is corrupted, but not if it is lost (so it might not be needed if the link layer protocol is reliable).

IP header: Contains the source header, Destination address, Protocol number and UDP length.

5.1 Remove Procedure Calls

Remote Procedure Calls: Allow calling procedures on a remove server as if they are local to the client. Eg. making database queries to a server (that is not local but can act like it is)

Client stub: Operates in the client address space

Server stub: Operates in the server address space

Marshalling: Converts the in-memory data structure to a form that can be stored and transmitted. I.e. turns your programs variables into a portable format.

Unmarshalling: Converts the stored or transmitted data into an in-memory data structure. I.e takes the structure given and converts it to be executed on the server.

Issues: With languages like C, issues can arise with pointers (unable to pass addresses between different address spaces), and unknown array sizes(can't automatically detect the size of arrays unlike in python). It is also unable to deduce parameter types and cannot share global variables between a server and client.

Protocols: Both UDP and TCP can be used for RPC. UDP is good if it is idempotent (running multiple times will not change the outcome, unlike a bank statement). TCP is good if non-idempotent operations are required.

5.2 Real Time Transport Protocol (RTP)

RTP: Multiplexes several streams into a single stream of UDP segments. Used in the presentation layer.

RTP Header: Has payload type (eg. mp3) which can vary, sequence number and timestamp (relative to the start of stream).

Real Time Transport Control Protocol (RTCP): Used with RTP, and handles feedback, synchronization and UI, like pausing and rewinding.

Feedback: Handing delays, bandwidth and congestion. RTCP can adapt the rate that data is sent to suit network conditions.

Synchronization: Where different streams use different clocks or have different drifts. RTCP adjusts the playback to match accordingly.

UI: RTCP can add additional features to RTP by adding names in a call, or pause/play functionality.

RTP Playback: If there is a variation in the rate that packets are sent, eg, there is a large gap and by the time the playback has reached the packet it is not available. The application can choose how to handle it, by waiting for the packet to arrive or skip the packet altogether.

6 Internet/Network Layer

Internet Layer: To get data from the source all the way to the destination. May be multiple hops (point to point) rather than end to end. As a result, traffic must be routed efficiently, and nodes need to be given names.

Internet: A network of networks. It is a sublayer on top of the network layer.

Hop: An entire network.

Network Layer: Similar to the internet layer, however has links instead of networks. Can provide connectionless services, which is called a datagram network, or connection oriented services, which is called a virtual circuit network.

Store and Forward Packet Switching:

1. Host H1 wants to send a packet to H2
2. Host H1 transmits packet to the closest router A
3. The packet is buffered while arriving, and the checksum is verified
4. If valid, the packet is stored until the outgoing interface(node) is free
5. The router forwards the packet to the next router in the path
6. steps 3 - 5 until it arrives at the destination H2

6.1 Datagram Network

Packet forwarding: A routing/forwarding table is used to decide on the route for a packet. At each node it lists the destination and which node to send the packet to.

Given a node A, which has connections to B and C. The routing table is as follows.

Destination	Forwarding
A	-
B	B
C	C
D	B
E	C
F	C

6.2 Virtual Circuit Network

Packet forwarding: In a connection oriented network, the paths are predetermined before the packet is sent, and routing tables are not changed often.

Given a node A, which has two hosts H1 and H3 connect to it, and that connects to a node C which connects to E.

Node A

Host	Incoming Packet Number	Destination Host	Outcoming Packet Number
H1	1	C	1
H2	1	C	2

Node C

Host	Incoming Packet Number	Destination Host	Outcoming Packet Number
A	1	E	1
A	2	E	2

6.3 Connection Oriented Vs Connectionless

Issue	**Datagram Network**	**Virtual Circuit**
Type	Connectionless	Connection-oriented
Addressing	Each packet has a full source and destination Large header may be an issue	Each packet contains a short VC number Small header is a benefit
State	Routers do not hold state information about connections	Each VC requires a router table space, which makes router reboots a problem
Routing	Each packet is routed independently (easy to change)	Routing is defined at setup (with a backup route)
Quality of Service	Difficult	Easy if we have enough resources
Congestion Control	Difficult	Easy if we have enough resources
Link failure recovery	Simple	Extra work

Multiprotocol Label Switching: Widely deployed Virtual Circuit (connection-oriented) Network Layer protocol. Primary purpose of MPLS is quality of service, as it prioritises traffic, has reliable connectivity with known parameters and has service agreements for network performance.

Quality of Service: Depending on the needs of your own network or a shared network, different services can be prioritised. For example YouTube would have streaming prioritised, while a home network might just have it optimised for web browsing.

7 Internet Protocol

Internet Protocol: Designed with a number of principle in mind.

- Something that works OK is better than an ideal standard 'in progress'
- Keep it simple
- Be strict when sending and tolerant when receiving eg. browsers can still display invalid HTML.
- Make clear choices - don't have different approaches in a standard
- Avoid static options and parameters
- Think about scalability

Responsibilities:

- Responsible for moving the packets through the various networks from a source to a destination host
- Multiple paths through network (important for redundancy)
- Routing algorithms are used to determine the best path
- Nothing is guaranteed, just does its best

7.1 IPv4

IP Header:

Field	Usage
Version	Protocol Version 4
IHL	Header length in 32 bits words; min 5, max 15
Differentiated services	6 bits for service class, 2 bits for congestion control (ECN)
Total length	Including payload, max of 65,535
Identification, DF, MF, Fragment offset	Used in the handling of fragmentation
TTL	Countdown of hops, at zero packet is discarded
Protocol	Transport layer service (UDP/TCP)
Source and Destination	IPv4 address
Options	Rarely used and poorly supported

IPv4 Addresses: A 32 bit number, which is expressed in decimal notation. Each byte is shown as a decimal separated by a period. Min is 0.0.0.0 and Max is 255.255.255.255. Due to irresponsible assignment of addresses, IPv4 address allocations are basically exhausted.

Address Types:

- Unicast: One destination (Normal addresses)
- Broadcast: Send to everyone
- Multicast: Sent to a particular set of nodes (eg. streaming of a live event)
- Anycast: Send to any one of a set of addresses (eg. database queries)
- Geocast: Send to all users in a geographic service (eg. earthquake warnings)

Classes: No longer used, however originally IP addresses were allocated based on classes. If there was a small network but large amount of users, they would be allocated to 1.0.0.0 to 127.255.255.255 etc. Due to this method of assigning via classes, it was very wasteful, eg. a network with 260 nodes didn't fit in class A and had to be assigned to class B with 65,536 addresses.

Classless InterDomain Routing (CIDR): Each interface/route explicitly specifies which bits are the network field. Networks with 260 nodes only need 9 bits for the host field.

CIDR Prefix: A contiguous block of IP address space, assigned to a network. Prefixes are written as the lowest IP address, followed by a slash and the size of the network portion.

CIDR Suffix: Contains the hosts for a network.

Network mask: Used for efficient routing on the internet. Routes are based on the network mask, and once it arrives at the destination, the network number can be extracted by using a network mask. Network number = network mask (bitwise AND) IP address.

Route aggregation: Performed automatically, and the longest matching prefix is selected for a route.

Special IP addresses:

Networks	Private Address Range	Prefix and Mask
Home	10.0.0.0 - 10.255.255.255	10.0.0.0/8 (255.0.0.0)
	172.16.0.0 - 172.31.255.255	172.16.0.0/12 (255.240.0.0)
	192.168.0.0 - 192.168.255.255	192.168.0.0/16 (255.255.0.0)
Broadcast	Network.255.255	
Loopback	127.Anything	

7.2 IPv6

Field	Usage
Version	Protocol Version 6
Differentiated services	6 bits for service class, 2 bits for congestion control (ECN)
Flow Label	Pseudo-Virtual Circuit Identifier
Payload length	Bytes after the 40 byte header
Next header	Used to specify additional headers or protocols (TCP/UDP)
Hop limit	Same as TTL
Source and Destination	16 bytes IPv6 address

IPv6 Addressing: Written as 8 groups of 4 hex digits. Can be optimised by stripping *one* group of consecutive 0's. Also compatible with IPv4 by appending the IPv4 address to the back of the address.

7.3 Network Address Translation (NAT)

Responsibility: Due to the scarcity of IPv4 addresses, this method was used as a way to handle the limited addresses. Rather than giving multiple IP addresses, each customer/home was assigned one public IP address, and then the hosts or networks internal to them could all use the same private IP addresses. These private addresses could not be used on the public internet, and must be translated to a public IP address if it is heading out of the network.

Local Area Network (LAN): Where internal/private IP addresses are used for communicating with hosts on the same network.

Process:

1. A packet originates from a device on the network (laptop etc). These are all connected to a router on the network.
2. The information is sent to a router, which has its own internal IP address and port number. The router is aware of which devices are connected to it.
3. The router then forwards this packet to a NAT box which converts the private IP address of the router, into the public IP address and port number of the company or network.
4. The translated packet is then forwarded to an ISP router, which forwards it through to the internet.

Notes on Process: When a packet enters the NAT box from the local network, the NAT translation table stores the index of entry with the original IP address and the original source number. This is used to traceback any responses from the packet.

Criticisms:

- Breaks end to end connectivity, an interface in the private network can only receive packets once it has sent a packet out and created a mapping in the NAT box.
- Layering violations by assuming nature of payload. It assumes that it contains a port number to work (ok for TCP and UDP) but otherwise might have to snoop on FTP messages.
- Violates IP architectural model which states that every interface on the internet has a unique IP address. Not really an issue, but different from the original design of IP.
- Changes internet from being connectionless to pseudo connection oriented. As the NAT stores the connections made to it, if it crashes those connections are lost and we have to reestablish them again.
- Also has a limited number of outgoing connections to $2^{16} \approx 65000$. For larger companies can be an issue.

Advantages: Has a significant security advantage as someone has to establish a connection first to receive a packet, which can be a benefit when dealing with unsolicited packets. Also has the ability to use *Carrier Grade NAT*, which gives customers private addresses. I.e A private address can have more private addresses.

7.4 Subnets

Subnetting: Allows networks to be split into several parts for internal use whilst acting like a single network for external use.

Example: A university with a /16 prefix can subnet its network. /17 can be assigned to computer science (which is half the allocation), /18 can be assigned to arts (which is a quarter of the allocation).

Subnetwork: When a packet arrives from the internet, the router can use the subnet masks (bit-wise AND) to find which subnet it should sent the packet to, without knowing all hosts on the subnet. Eg. If a packet comes from X.X.X.1, we can send it immediately to the host beginning with 1. If there are more bits, we go down until it can be identified.

8 Fragmentation

Issue: Although IP packets can have a maximum size of 65,535 bits ($2^{16} - 1$), most networks cannot handle those sizes. All networks have a maximum size for packets due to limitations such as error handling, OS, hardware etc.

Fragmentation: Divides packets into fragments which allows network gateways to meet size constraints.

Goal: The host wants to transmits packets as large as possible, as the workload would be reduced on their end. However, not all networks will support such a large packet size. As a result we want a packet large enough that headers aren't considered a big load (eg. 1 byte with 40 byte header is non efficient), but small enough that the network can process it.

Maximum Transmission Unit (MTU): The maximum size that each link in a network can carry.

Path MTU: The maximum size for the path through the network, i.e. the minimum size for all links. Eg. if link A can accept 5 bytes, and link B can accept 10 bytes, then the path MTU is 5 bytes. This can change if the route is dynamic (eg. we remove link A and go through link C instead).

Transparent Fragmentation: Fragments are sent to a router, which then performs reassembly before sending it off to the next router. If the next router cannot handle the reassembled packet, it will fragment according to the next router's MTU. Eg. From A the packet is split into 5, when it reaches B it is reassembled, however it is too big for C. B splits the packet into 3 (based on C's MTU) and sends it as 3 fragments. This is more efficient in terms of packets, however does create more work for each router.

Nontransparent Fragmentation: Reassembly is performed at the destination host. The packets remain the same size throughout the route, so the number of fragments is based on the Path's MTU rather than the routers.

Fragmentation Header:

- Identification bits: used to identify a packet (same for all fragments of the same packet)
- MF: More fragments bit. Will be 1 unless it is the last packet.
- DF: Don't fragment, if 1 won't fragment
- Fragment offset: Must be on an 8 byte boundary (multiple of 8), unless it is the last fragment. For example, we can have FO = 185, which means a fragment of size 1480 is sent (+ 20 for header).

Disadvantages: Overhead from fragmentation (20 bytes) is incurred from the point of fragmentation to the host. If a single fragment is lost, we have to resent the entire packet. Overhead on hosts performing reassembly is higher than expected.

Path MTU Discovery: Aims to avoid some of these disadvantages. Each packet is sent with the DF bit set (Don't fragment). If a router cannot handle the packet size, it sends an ICMP (Internet Control Message Protocol) to the sender host, telling it to fragment the packets to a smaller size. At first packets may be dropped, but then the host will learn the optimal size quickly and reduce subsequent fragmentation.

1. Source sends packet as is (non-fragmented)
2. Router 1 receives packet, however it can only handle packets of size 1400, so returns an ICMP and drops packet
3. Source sends packets as size 1400
4. Router 1 accepts packet and forwards to Router 2
5. Router 2 can only handle packets of size 1200, so returns ICMP to source and drops packet
6. Source sends packet as size 1200
7. Router 1 and 2 accept the packet, and Router 2 forwards it to the destination
8. Destination rejects packet as it can only handle size 900 and sends ICMP back to source.
9. Source sends packet as size 900
10. Packet reaches destination with no issues
11. Source sends more packets as size 900 with no issues

IPv4 Fragmentation: IPv4 allows for either nontransparent fragmentation, or path MTU discovery. IPv4 minimum accept size is 576 bytes, and if this is not satisfied, then applications have to handle packet delivery by themselves.

IPv6 Fragmentation: IPv6 expects hosts to discover the optimal path MTU. Routers will not perform fragmentation in IPv6, however the sender and source and perform fragmentation if they wish. IPv6 minimum accept size is 1280 bytes.

9 Internet (Network) Layer

Packet Forwarding: The process of delivering a packet according to each router's routing table. This maps destination addresses to outgoing interfaces.

Routing: The process of deciding on the route rather than forwarding a packet. A routing algorithm is used to decide which output line an incoming packet should be transmitted on. These are run on a router to create a routing table.

Routing Algorithm:

- Correctness - that a valid route is found between all pairs of nodes
- Simplicity - easy to understand and modify
- Robustness - a router crash should not require a network reboot, should be able to route around a broken node
- Stability - a stable algorithm should reach equilibrium and stay there, don't want flapping routes
- Fairness
- Efficiency
- Flexibility to implement policies

Fairness vs. Efficiency: Can conflict with each other. For example if we have 2 routes, and one is faster than the other the most efficient method would be to just use the faster route. The fair solution would be to give equal time to both of the routes (assuming they can't be used at the same time).

Delay vs. Bandwidth: Aims to optimise either packet delay or the network throughput. We can use a simple approach of minimizing hops which tends to reduce bandwidth but improve delay. An alternative is to use weighted paths which give a value to a path based on distance or if we want to use it or not.

Non-adaptive Routing (static):

- The routing does not change (even in errors/broken links)
- Does not adapt to the network topology
- Calculated offline and uploaded to the router when started
- Does not respond to failure
- Mainly used where there is a clear or implicit choice. E.g. home networks likely only have one router, so if that router is down, then we cannot route around it regardless.

Adaptive Routing:

- Adapts to changes in topology and potentially traffic levels
- Can also adapt to own requirements, such as we expect more traffic at a certain time of day, so we take this router, but change the route in off peak times
- Optimises some property: distance, hops, estimated transit time etc.
- May get information from adjacent routers, or all routers in the network

Flooding: Simplest example of adaptive routing.

- Guarantees shortest distance and minimal delay
- Used as a benchmark in terms of speed
- Extremely robust - if a path exists, flooding will find it
- Highly inefficient - generates a lot of duplicate packets
- Have to have a method of discarding packets (TTL or through network)
- Process is that we start from a node, which sends out packets to all neighbour nodes. The neighbour nodes will send out more packets to all nodes it is connected to and hasn't received a packet from until we have the topology of the network.

Optimality Principle: Not always applicable for BGP, however assumes that there is an optimal path, and other optimal paths are within that path. A set of optimal routes from all sources will form a tree which has a root at the destination.

Shortest Path Algorithm: Uses a labelled graph to find the shortest path between nodes. The most famous example is Dijkstra's algorithm.

Dijkstra's Algorithm: Divides nodes into three groups: unseen, open and closed. We visit unseen or open nodes which have the current lowest value, and explore its neighbours. After a node is visited it will be marked as closed (cannot revisit).

Unseen: Not a neighbour of a node which we have processed.

Open: We have visited a neighbour of this node, but not this node yet. We are aware of at least one path to it.

Closed: We have visited this node, and we know the best path to it. Cannot be revisited (in the algorithm).

Dijkstra's Algorithm Process:

1. Visit the source node, *'open'* all of its neighbours and set their labels as the distance to them.
2. Examine adjacent nodes to the 'working node', calculate the distances and update labels if it has improved (lower cost to visit from another node)
3. Examine all tentative/open nodes in the graph, then pick the one with the lowest distance and visit it.
4. Repeat until all nodes are visited or the destination is found (depending on purpose)

Link State Routing: A distributed algorithm that replace Distance Vector Routing. Has 5 steps to the process.

1. Discover its neighbours and learn their network address
2. Set the distance or cost metric to each of its neighbours
3. Construct a packet containing all it has just learned
4. Send the packet to, and receive packets from, all other routers
5. Compute the shortest path to every other router

HELLO packet: Used to discover neighbouring routers. When a router starts up it sends a HELLO packet on each interface and the router on the other end responds with its own ID.

Link State Routing Costs: Can be set automatically, manually or using delays (ECHO packet). It can also be engineered based on traffic.

Link State Packet: Consists of an ID, sequence number, age, a list of neighbours and their respective costs. Building a packet is easy, however deciding when to build them is difficult.

Sending Packets: In link state routing, flooding is used to send packets to other routers. More specifically, reliable flooding which uses acknowledgements to guarantee every other router receives the packet. Routers also compares the packets it receives to check that the SEQ is higher than the last packet it has received. If it is, it will forward the packet otherwise it will discard it.

Distance Vector Routing: Instead of a router sending packets, nodes announce the distance from themselves to each destination. When they receive a new announcement of others' distances, they update their own estimates and make a new announcement. This is however less efficient and has issues.

Autonomous Systems (AS): Collections of routers under the same administrative control.

Routing Protocols: Exist for both internal routing (within a network) and external routing (between ASes or different networks).

Border Gateway Protocol (BGP): Routing between networks can be based on a lot more than just the shortest or most efficient route. It often also involves politics. As a result Bellman's principle will not apply, as though it might be the shortest path, it could route through a network that the client doesn't want.

- Companies may not want others using their networks for free
- Competing ISPs might not want to carry other ISP's traffic on their networks
- Not carrying commercial traffic on academic networks
- Prefers a provider because it is cheaper
- Not wanting to send traffic through a particular company or country

Border Gateway Protocol (BGP): Based on customer or provider agreements, either one pays to send or receive traffic, or there is a mutual agreement to carry each others traffic. Providers also only advertise their routes to their own network to avoid carrying other people's traffic. Most paths taken using BGP are valley free, most of the time it originates then goes up to a higher network and only comes back down when it reaches its destination.

10 Internet Control Protocols

Data plane vs Control plane: An alternative model to the stack of protocols. One plane is in charge of decisions and one is in charge of moving data. There can also be a management plane for setting policies.

Data plane: Responsible for handling data, for example in the network layer, this would be in charge of forwarding packets.

Control plane: Responsible for deciding how data will be handled. In the network layer, this would be choosing routes that the packets will be forwarded to.

Internet Control Protocols: Protocols used by the internet layer to manage functionality.

10.1 Internet Control Message Protocol (ICMP)

Messages:

Message Type	Description
Destination Unreachable	Packet could not be delivered
Time exceeded	Time to live field hit 0
Parameter problem	Invalid header field
Source quench	Choke packet (send packet more slowly)
Redirect	Teach a router about geography
Echo and echo reply	Checks if a machine is still alive
Timestamp request or reply	Same as echo but with a timestamp
Router advertisement	Finds a nearby router

Traceroute: Uses the TTL field to send out packets and detemine the paths and times of a route. For n amount of hops in a given destination, we can discover each router's IP address by sending a packet out to the same destination. We can increment the TTL from 1 to n, so that we can get the IP address of a router at x hops.

1. Traceroute sends out a packet with a TTL of 1
2. Packet reaches router, then is decremented and expires. Router returns a time exceeded message, which has its IP address in it.
3. Traceroute sends out another packet with a TTL of 2, and repeats step 2 until n number of routers have been discovered.
4. Sender can use this information to determine a path and the timings of a route the packet will take.

10.2 Dynamic Host Configuration Protocol (DHCP)

DHCP: Automated way of handling IP address allocation. Has security concerns, but is more efficient than manually configuring each host.

Process:

1. Network has a DHCP server for issuing IP addresses
2. Host sends out a DHCP DISCOVER packet
3. DHCP Server receives this packet and responds with a DHCP OFFER packet which contains an available IP address
4. IP addresses are typically issued on a lease, and after it expires the server can reclaim it and re-issue it. Hosts can also requests for a renewal before the lease expires.
5. Used to set a number of parameters, like default gateway, DNS servers addresses and time addresses

Media Access Control (MAC) Address: Used as the DHCP won't know where to send the response to before issuing the IP address. It exists in the link layer, and is also known as the physical layer address. It is a globally unique identifier for the interface, is usually hard coded by the manufacturer, and is between 48 and 64 digits long.

10.3 Address Resolution Protocol (ARP)

ARP: The link between the internet layer and the physical network layer. It allows a host to translate an IP address to a physical address (MAC).

Process:

1. Broadcasts an Ethernet packet within its local network to ask who owns the target IP address
2. Broadcast arrives at every host on the network, the owner will respond with its MAC address
3. The low level sending is done via MAC addresses, so this protocol is run often, even to find out how to communicate with the nearest router.

Proxy ARP: Rather than all routers being woken up for every ARP request that may or may not be relevant to it, we can direct our requests to a proxy, which contains the information on all MAC Address mappings. Once the packet reaches the proxy, the proxy forwards it to the relevant router, which will wake up and respond to the ARP request.

Benefits: Incredible simple.

Issues: Not efficient, and has security issues such as lack of authentication, ARP spoofing and intercepting ARP messages to associate attackers MAC address with a different host IP address.