CO212 - Networks and Communications

14th January 2020 **Week 1, Lecture 1**

Evolution of the Internet

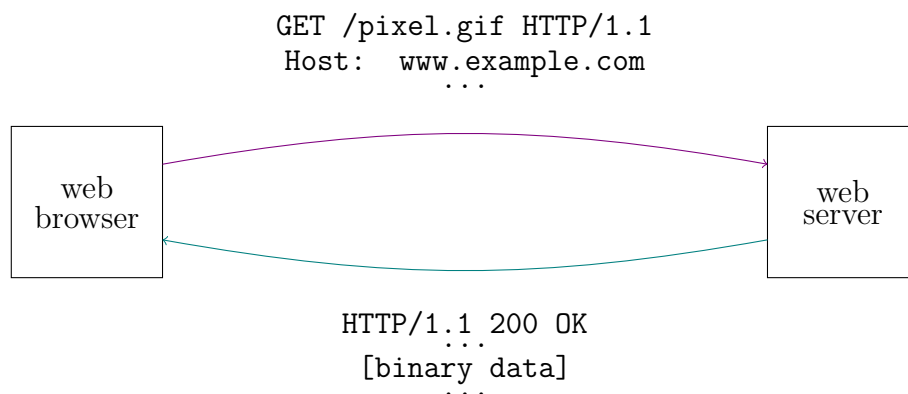Literally only writing this part so I have something for the first lecture.

- (1969 - October) first message sent on ARPANET; "login", crashed after "l" and "o" were sent
- (1971) universities in West and East coast of USA connected
- (1980) London connected

14th January 2020 **Week 2, Lecture 1**

World Wide Web (WWW)

This is an example of an **application** on the internet, based on HTTP (HyperText Transfer Protocol). A web browser (the client) sends a request to the web server over a pipe, which can be any form of connection between the two devices (can also be the same device), which in turn sends back a response.

```
GET /pixel.gif HTTP/1.1
 Host:  www.example.com
          ...
```

```
+----------+                    +----------+
|   web    |  ───────────────▶  |   web    |
| browser  |  ◀───────────────  |  server  |
+----------+                    +----------+
```

```
 HTTP/1.1 200 OK
       ...
   [binary data]
       ...
```
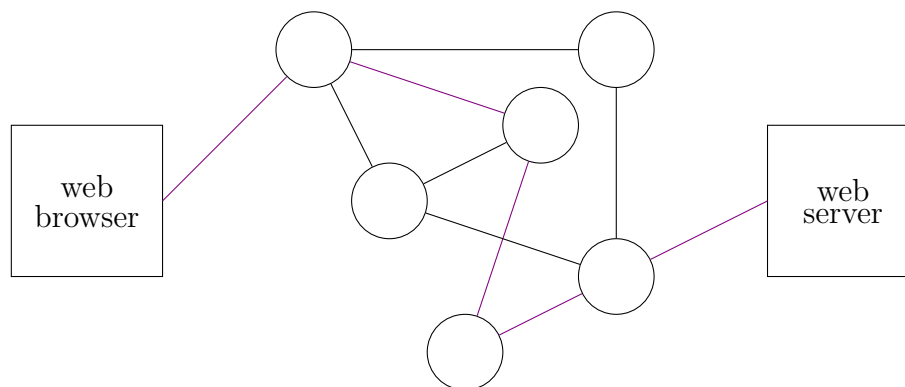
Layers

- **application layer**

  Any software written for the internet is on the application layer.

- **transport layer**

  In the **transport layer**, packets leave your (client) machine to the server, and the server sends back packets to your client. This layer divides a (big) message into smaller chunks, and sends them to the other side (re-ordered) to be presented to the recipient.

- **network layer**

  The **route / path** (sequences of switches a packet goes through) each packet takes can be different from the others, and is often the most optimal route available. This is done on the **network layer**, which routers are a part of.

- **data link layer**

  Our devices are linked to the network on the **data link layer**, via network interface controllers (NICs). Examples of this include Ethernet, fiber optic network cards, as well as wireless devices such as WiFi access points, and USB dongles for 4G. A communication link is any connection between packet switches and / or end systems.

- **physical layer**

  Finally, on the **physical layer**, there are various forms of communication media, including fiber-optic cables, twisted-pair copper wire, coaxial cables, and wireless local-area links (802.11, Bluetooth, etc).

16th January 2020 — Week 2, Lecture 2

Circuit Switching

Old phones used circuit switching, which creates a connection between the two points, which is used for the entire communication. This isn't used for the internet as the failure of one node in the circuit would lead the the entire communication dropping - whereas a different route would be calculated in packet switching.

Compared to packet switching, it has an expensive setup phase, but will need very little processing once the connection is established. However, it is inefficient for sharing resources - if a node is used as part of a circuit, it cannot be used by another connection for a different circuit. The resources are blocked once a connection is established (hence it is an inefficient way to use the network). On the other hand, packet switching has no setup cost, but has a processing cost, as well as space overhead, for every packet. It has a processing cost for forwarding the packets, as well as space overhead as there must be redundant data for each packet, such that it is self contained. It is specifically designed to share links, hence it allows for a better utilisation of network resources.

Protocols

A protocol is a set of rules (an agreement between communicating parties on how communication is to proceed), run by end systems as well as packet switches. It must be unambiguous, complete (includes actions and / or responses for all possible situations), and also define all necessary message formats. The phases are as follows;

- **handshake** — establishes identities and / or context
- **conversation** — free-form exchange
- **closing** — terminating the conversation

The internet protocol stack is based on the 5 layers briefly covered in the previous lecture. Some examples of design issues that can be encountered are as follows;

- **addressing** — how to denote the intended recipient
- **error control** — how to detect (and possibly fix) transmission errors, e.g. checksums
- **flow control** — ensure data travels through communication media without issues
- **multiplexing / demultiplexing** — conversion of data into binary, and parallel communications
- **routing** — which route is chosen

Most network layers are either connection-oriented, where a connection is first established, data is exchanged, and the connection is finally released, or connectionless, where data is marked with its destination.

The TCP/IP (4 layer) stack consists of application, transport, internet, and network access (which combines data link and physical). On the other hand, the OSI (7 layer) model consists of the application layer, presentation, session, transport, network, data link, and physical.

application
presentation
session
transport
network
data link
physical

application
transport
network / internet
data link
physical

application
transport
internet
network access
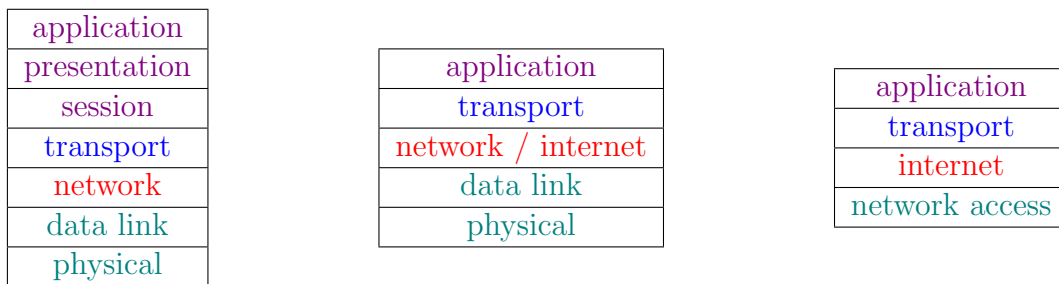
A **service** is a set of primitives that a layer provides to the layer above it, whereas a **protocol** is a set of rules that prescribe the layout and meaning of packets. In a protocol stack, layer k puts its entire packet as data into a layer $k-1$ packet, the latter may add a header and / or a trailer. This may have to be split across several lower level packets (**fragmentation**). An example of protocol layering is as follows;

Note that the connection between the two machines can have multiple nodes in between, that can read up to a different physical layer. For example, if there was a link-layer switch after the source machine, it can read up to the link layer (layer 2), remove and add headers / trailers, and then send it on to the next device. The next device may be a router for example, which can read up to the network layer (layer 3), and do the same.

The data from the layer above is known as the SDU (service data unit), and the SDU combined with a header, added by the current layer, is known as the PDU (protocol data unit).

16th January 2020   **Week 2, Lecture 3**

Protocol Layers

The types of protocols on each layer are as follows;

- **application layer**

  Protocols on the application layer defines functionality and message formats. Some examples are as follows;

  - **traditional**   name services (DNS), sending email (SMTP), file transfer (FTP), web (HTTP(S))
  - **modern**

    includes middleware protocols to support distributed systems with special protocols to handle replication, fault tolerance, caching, etc.
  - **high-level**   special application-level protocols for e-commerce, banking etc.
  - **peer-to-peer**   BitTorrent, old-Skype (awful API)

- **transport layer**

  These protocols generally offer connection-oriented (TCP) as well as connectionless (UDP) services, which have varying degrees of reliability. These often provide a network interface to application via sockets. It's also important to note there is a difference between reliability and security; the former guarantees that the data is sent, whereas the latter ensures that it is encrypted in some form. This layer also provides flow control; mechanisms to ensure fast senders don't overwhelm slow receivers.

- **network layer**

  In this layer, the protocols generally describe how routing is performed, such as determining which computers / routers are in the network, the best route between two points, how to handle faults (such as a device going down), as well as handling congestion (when a router is overloaded and packets are dropped).

- **data link layer**

  In this layer, we need to detect bit transmission errors. This can be done by adding redundancy bits in frames to detect errors - for example adding a parity bit to every 7 bits, where a 1 indicates an odd number of 1s, and a 0 indicates an even number of 1s, or adding a checksum (cyclic redundancy check) which should match the bits before it.

  It also specifies how many computers can share a common channel, with the medium access control sub-layer (MAC). A well known protocol is the Ethernet protocol.

- **physical layer**

  This describes the transmission of raw bits, in terms of the physical mechanical and electrical issues. For example, when two computers are connected with a wire, -3V may indicate to a binary 1, and +4V may correspond to a binary 0. It may also specify the number of times the voltage can be changed per second.

  For example if the voltage can be changed 20,000 times per second, it indicates that the maximum transfer rate is 20,000 bits per second, which is 20 Kbps (20 kilobits per second).

Units

- 1 Byte = 8 bits   note that a byte has an uppercase B, whereas a bit has lowercase b
- 1000 Bytes = 1 KB   (Kilobyte)
- 1024 Bytes = 1 KiB   (Kibibyte)

Typically we use powers of 10 for networks, but as long as we are consistent in what we use it is fine.

Quantifying Data Transfer

- **bandwidth**

  the amount of information that **can** get into the connection in a time unit

- **throughput**

  the amount of information that **actually** get into the connection in a time unit - at steady-state, we assume zero accumulation of traffic therefore the input and output throughputs are equal
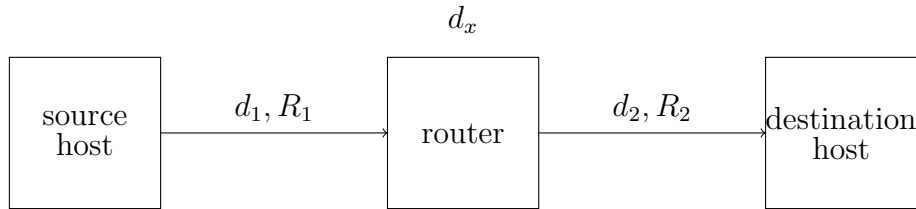
- **latency**

  the time it takes for one bit to go through the connection

Note that for the following formula, we use these values;

- t_0 — the time the first packet leaves the source
- t_1 — the time the first packet reaches the destination
- t_2 — the time all the packets reach the destination
- L — the size of the packet in bits
- latency (propagation delay) — $d = t_1 - t_0$ (generally $\frac{\text{distance}}{\text{wave propagation speed}}$)

  note that this is half the RTT (round-trip time)

- throughput (link bandwidth) — $R = \frac{L}{t_2 - t_1}$ (generally $\frac{\text{transferred bits}}{\text{duration}}$)

- packetization (transmission delay / store-and-forward delay) — $\frac{L}{R}$

  time it takes for the entire packet to be received after the first bit is received

- transfer time (propagation delay + transmission delay) — $\Delta = d + \frac{L}{R}$

However, it's important to note that the connections are (almost) never direct, and therefore will have additional delays at each router. Note that our bandwidth is also bottlenecked by the lowest bandwidth.

$$d_x$$

```
┌──────────┐   d_1, R_1   ┌────────┐   d_2, R_2   ┌────────────┐
│  source  │ ───────────> │ router │ ───────────> │destination │
│  host    │              │        │              │   host     │
└──────────┘              └────────┘              └────────────┘
```

$$d_{\text{end-end}} = \begin{cases} d_1 + \frac{L}{R_1} + d_x + d_2 & R_1 < R_2 \\ d_1 + d_x + \frac{L}{R_2} + d_2 & R_1 \geq R_2 \end{cases}$$

$$= d_1 + d_x + d_2 + \frac{L}{\min(R_1, R_2)}$$

The router delay, d_x has two components; the processing delay d_{proc} which is the processing time (checking for bit errors and determining the output link), as well as d_q, which is the queueing delay - the time waiting at the output link for transmission, which depends on how congested the router is. We can quantify the **traffic intensity** as follows;

$$R = \text{link bandwidth}$$
$$L = \text{packet length (bits)}$$
$$a = \text{average packet arrival rate}$$
$$\frac{La}{R} = \text{traffic intensity}$$

$$\frac{La}{R} \approx 0 \qquad \text{small average queueing delay}$$

$$\frac{La}{R} \to 1 \qquad \text{large average queueing delay}$$

$$\frac{La}{R} > 1 \qquad \text{more working arriving than can be serviced, infinite delay}$$

21st January 2020 **Week 3, Lecture 1**

Clients and Servers

We can distinguish between the two roles in a pair of communicating processes, in a connection-oriented model;

- **client** initiates communication

- **server** waits to be contacted

On the other hand, some applications have processes that act as both the client and the server, which is referred to P2P (or peer-to-peer) architecture.

End System Applications

Internet applications are processes on the end systems. They must have a way of addressing each other, either via the internet (such as chat servers), or they can directly communicate (such as FTP) - but this depends on the protocol in use. Note that a single end system (or host) can run multiple programs, which run multiple processes, all of which connect through a network API provided by the operating system. Each process is addressed within its host by a **port number**. When an application wants to communicate with another application, the OS opens a socket, which allows data to be transferred between the two machines.

client application

1. create a socket C by connecting to server application (connecting to host H on port P)

2. read and write data to socket C

3. disconnect and destroy C

server application (on host H)

1. create a socket S by accepting connection on part P (port is often called a server socket)

2. read and write data to socket S

3. disconnect and destroy S

The server application can open more sockets to server multiple clients at a time. A DDoS (distributed denial of service) attack works by opening many sockets, preventing the server from serving legitimate requests.

The World Wide Web

Invented in 1989 (formally defined in 1991) by Sir Tim Berners-Lee. Based on the idea of hypertext and hyperlinks (based on a proposal by William Tunnicliffe in late 1960s). Commonly used terminology is as follows;

- **document** a webpage is called a document

- **object** any called within a document (images, stylesheets, etc.)

- **Uniform Resource Locator (URL)** specifies the address of an object

- **browser** also called user agent - client used to access documents

- **web server** application that makes documents and objects available through HTTP

Protocol

In general, the request and reply would include the following;

request

- protocol version
- URL specification
- connection attributes
- content / feature negotiation

reply

- protocol version
- reply status / value
- connection attributes
- object attributes
- content specification
- content

A protocol should always include a version number, as it allows the protocol design to change. HTTP/2 will replace HTTP/1.x in the next few years (hopefully), as the former is able to fully multiplex a connection since all content is binary, and can use a single TCP connection for parallelism.

The URL contains the host name, which determines where the requests goes, by mapping to a network address. The request consists of a request line, such as `GET /path/to/index.html HTTP/1.1`, zero or more header lines, an empty line, followed by the object body (which can be empty). The request line contains a method, such as;

- `GET`   retrieve the object identified by the URL
- `POST`   allows for submission of data to the server
- `HEAD`   similar to `GET` but only receives headers
- `PUT`   requests the enclosed object to be stored under the given URL
- `DELETE`   deletes the given object
- `OPTIONS`   requests the available communication options for the given object

The status code in a server response is generally as follows;

- `1xx`   informational
- `2xx`   successful
- `3xx`   redirection (e.g. object has temporarily or permanently moved)
- `4xx`   client error (e.g. malformed request, unauthorised, object not found, method not allowed)
- `5xx`   server error (e.g. internal server error, service overloaded)

23rd January 2020   Week 3, Lecture 2

How HTTP uses TCP

HTTP uses TCP as it is essentially a file transfer protocol, which needs to be connection-oriented. The first version of HTTP uses a TCP connection for each object, which was an inefficient use of both the network and the operating system. HTTP/1.1 introduced persistent connection, which allows for an existing connection to be used to issue multiple requests (either sending a request, waiting for a response, sending the next request and so on, or through pipelining) - which will eventually close with a timeout. Pipelining allows the client to send all its requests without waiting for a response, and the server delivers them in response.

Web Caching

A proxy is a server which acts as an intermediate between the client and the destination server. This can be used for caching by storing a copy of the content, which reduces load on the origin server, and also allows for lower latency. Data isn't cached for an extended period of time as it can lead to stale data (where old content is served to a user, even after the content is changed on the origin server). Proxies can also protect the clients by providing anonymity, as well blocking malicious content through the use of a single firewall on the proxy. However, this also acts as a single point of failure - if the proxy fails then the clients may not be able to connect.

A `HEAD` request could be used to see if an object has been updated, which is less expensive than retrieving the entire object with a `GET` request. The request can also include `Cache-Control: no-cache`, which indicates it does not want cached objects, thus requiring proxies to go to the origin server, or `Cache-Control: max-age=20`, which only gets a cached object if the cache is less than 20 seconds old.

On the other hand, the reply can also include `Cache-Control: no-cache`, which informs the proxy not to cache the object, or `Cache-Control: maxage=100; must-revalidate`, which specifies to the proxy that it must revalidate the object after 100 seconds.

Sessions

Note that HTTP is a **stateless** protocol, which means that responses have no memory of past requests. However, HTTP allows higher-level applications to maintain **stateful** sessions, via the use of cookies. The `Set-Cookie` header is sent from a server, informing the client to store the cookie as a session identifier for that site. On the other hand, the client sends a `Cookie` header, which tells the server which session the request belongs to. This can be useful for storing identifying a user on a page (allowing for personalised pages). However, this can be also be used to track users for profiling and targeted advertising - leading to privacy issues.

Dynamic Web Pages

Servers often generate pages on-the-fly, instead of only serving statically stored pages. CGI (common gateway interface) allows you to identify a program and its parameters in a URL, which then starts a process to execute the program and return any results as a regular web page. On the other hand, servelets in Java maintain state (whereas CGI is stateless), and the webserver contains an instance of the JVM.

Another approach is for the web page to incorporate interpretable code, which is executed when the page is being processed on the client-side (via JavaScript). It's important to distinguish this from server-side processing in something such as PHP, which the user should not see.

IP and Hosts

Each end system is identified and addressed by its IP address, which is 32 bits in IPv4, or 128 bits in IPv6. This is easy to process by a computer, as it can easily work in powers of two, but not practical for people to use.

Host names are used to create human-readable "aliases" for IP addresses. Originally, before 1983, all mappings were in the `hosts` file, since there weren't many different hosts. However, as more hosts became present, DNS (Domain Name System) was developed, which provides a distributed lookup facility.

There are 13 root DNS servers, which know where the top-level servers are located. The top-level domain servers are each associated with a top-level domain. However, knowing where to connect to requires a large amount of communication between servers. This can therefore be a bottleneck for applications, and also be a critical point of failure. To circumvent this, we can also cache DNS lookups.

This improves performance as we do not have to do as much communication, and it also reduces the overall load on the DNS infrastructure.

However, this can be an issue if enough DNS servers advertise an incorrect lookup, causing subsequent requests to point to an incorrect IP (DNS cache poisoning).

DNS query types are as follows;

- `A` maps a host name to its address, **name** is a host name, and **value** is its IP address
- `NS`

    a query for a name server, **name** is a domain name, and **value** is the authoritative name server for that domain

- `CNAME`

    a query for a canonical name, **name** is a host name alias, **value** is the primary host name

- `MX`

    a query for the mail exchange server **name** is a host or domain name, and **value** is the name of the mail server handling incoming mail

The DNS protocol is connectionless, and runs on UDP port 53. UDP (best effort) is used since it only involves two network packets (request and response), setting up and closing a TCP (reliable) connection every time would be wasteful. If it fails, it can just try again.

Round Robin DNS is a load balancing technique, as it responds to DNS requests with a list of IP addresses instead of a single IP address. The IP at the top of list is returned a set number of times before it is moved to the bottom of the list (the IP that was previously second in the list is now at the top). If load balancing is used, the TTL should be small, as we want this to constantly change depending on server load.

23rd January 2020 Week 3, Lecture 3

Content Distribution Networks (CDNs)

We have the following options when we want to provide a streaming service from millions of videos to many simultaneous users;

1. single, large "mega-server"

    - single point of failure, and point of network congestion
    - long path to distant clients (slow)
    - multiple copies of video sent over outgoing link

    This solution does not scale to a large amount of users.

2. store multiple copies of videos at multiple geographically distributed sites

    - **enter deep** push CDN servers into many access networks (close to users) used by Akamai (216000+ servers, 120+ countries, 1500+ networks)
    - **bring home** smaller number of large clusters in PoPs near (not within) access networks used by Limelight (80+ PoPs - Points of Presence)

A CDN DNS can select a good CDN node by picking the CDN node closest to the client, or a CDN node with the shortest delay to the client. However, it's important to note that the CDN doesn't know the IP address of the client, only the address of the local DNS which may not be fully accurate. An alternative is to let the client decide by giving a list of CDN servers. The client can then select the "best" based on the lowest RTT.

Electronic Mail

While email was able to achieve asynchronous communication, one-to-many communication, as well as multimedia content it had a number of limitations. Privacy and security was an issue, as there was initially no authentication - hence messages could be modified or forged, and messages could be read by others. Additionally, it was unreliable as there were no delivery guarantees, and had no reliable acknowledgement system.

The user agent is the client the user uses to read, compose, reply, send and forward messages. The mail server can do the following;

- accept messages for remote delivery (delivers message to remote destination server with transport protocol)
- accept messages for local delivery (saves messages in local persistent mailbox)
- allows user agents to access local mailboxes (to allow for retrieval and / or deletion of messages)

Simple Mail Transfer Protocol

It's important to note that the 'S' in SMTP does not stand for secure (no authentication). This is a connection-oriented protocol (TCP), on port 25. As it is a very simple protocol, it can be left unsecured - this is often targeted by spammers and phishers who use unsecured mail servers to send mail without using their own resources.

The general format of using SMTP is headers, followed by an empty line, and then content. A single dot is used to end the email, and QUIT is used to exit. SMTP is completely oblivious to the contents of a message, other than the **received** header, which is added by each receiving SMTP server. This can be used to trace the origins of an email.

The initial message format had many limitations - it only supported 7-bit text content, and was essentially only usable for the English language. The MIME (multipurpose internet mail extensions) format defined useful extensions on top of SMTP, which includes the following types;

- text/plain   normal ASCII message
- text/html   HTML-formatted message
- image/jpeg   contains only an image file
- multipart/mixed   consists of multiple parts

Post Office Protocol (POP3) and IMAP

The user's mailbox is often stored on a different machine than the user agent, but we need remote access to incoming (and outgoing) messages. However, POP3 is insecure (at least on port 110), due to the transmission of credentials in plain text. It also implicitly assumes the retrieved mail is deleted at the server, which isn't useful if people wanted to access mail from different clients.

The internet message access protocol (IMAP) solves this, by storing messages on a server, and requiring the client to be online to read mail.

Dark Web

TOR (The Onion Router) hides the user behind a series of machines (proxies), and also allows for access to `.onion` sites (as well as regular ones). This can be used for privacy purposes, as well as for circumventing censorship. The exit nodes of TOR can be owned by law enforcement agencies, allowing for a user to be compromised if they were to subpoena all intermediate proxies.

Not that this shouldn't be confused with the **deep web**, which is typically just not indexed on the surface web.

27th January 2020   Week 4, Lecture 1

Transport Layer

The transport layer provides both reliable connection-oriented services (TCP - transmission control protocol), as well as unreliable connection-less services (UDP - user datagram protocol). This provides for logical communication between application processes, and only runs on end hosts (not routers / switches). TCP data are called segments, whereas UDP data are called datagrams.

We also assume that the underlying layer is working, with every host having a unique IP address, and that IP is a "best-effort" delivery service. This means that it has no guarantees on the integrity of data (or packet) transmission, nor does it guarantee the order of delivery of packets or segments.

Data Encapsulation

For each layer, we have the following terminology - we can't refer to everything as packets;

- application layer   data
- transport layer   TCP segments or UDP datagrams

    (TCP only) segmentation can be done when the segments are too large - if the UDP datagrams are too large, it cannot be sent

- network / internet layer   IP datagrams (or packets)

    similarly, fragmentation can be done when the packets are too large

- data link layer   frames
- physical layer   bits

Ports

Note that it's possible for a client (one IP) to communicate with the same host (one IP) via multiple applications, such as HTTP and SMTP. Each application on ta host is identified with a unique port number - they can be thought of as cross-platform process identifiers. A socket consists of two pairs of `ip_address` + `port_number` + `TCP/UDP`; such as

$$146.179.40.24:80 \text{ TCP} \Leftrightarrow 192.168.1.1:7155 \text{ TCP}$$

The first 1024 ports (0 - 1023) are reserved, and cannot be used to form a connection as a client (unless it is done as a superuser).

Transmission Control Protocol

TCP is the Internet's primary transport protocol. It is a connection-oriented service, and endpoints initially perform a handshake to establish a connection. This is a full-duplex service, thus both endpoints can send a receive at the same time. Some definitions for TCP are as follows;

- **TCP segment**   "envelope" for TCP data

- **maximum segment size (MSS)**

  maximum amount of application data transmitted in a single segment (does not include headers) - typically related to MTU to avoid network-level fragmentation

- **maximum transmission unit (MTU)**   largest link-layer frame available to the sender host

  path MTU discovery (PMTUD) is used to determine the largest link-layer frame that can be sent on all links from the sender to the receiver

The TCP header consists of the following fields;

- source and destination ports   16-bit each (identifies applications)

- sequence number   32-bit (used to implement reliable data transfer)

  These numbers are not associated with the segments, but instead are associated with the bytes in the data stream. It indicates the sequence number (the place) of the first byte carried by the TCP segment. When the connection is initialised, a random ISN (initial sequence number) is decided upon to avoid accidentally receiving leftover segments.

- acknowledgement number   32-bit (used to implement reliable data transfer)

  Represents the first number not yet seen by the receiver (one higher than the sequence number of the last bit received). These can be cumulative, and typically TCP implementations acknowledge every other packet;

```
A                                                                        B
|              [seq#=1200,...], size(data) = 1000                        |
|----------------------------------------------------------------------->|
|                                                                        |
|               [seq#=2200,...], size(data) = 500                        |
|----------------------------------------------------------------------->|
|                                                                        |
|                    [seq#=...,ack#=2700]                                 |
|<-----------------------------------------------------------------------|
|                                                                        |
|                            ⋮                                            |
v                                                                        v
```

Note that because a TCP connection consists of a full-duplex link, there are two streams, hence two different sequence numbers (see the example below, of an "echo" application). Acknowledgements are "piggybacked" on data segments.

```
A                                                                        B
|                    [seq#=100,data="C"]                                 |
|----------------------------------------------------------------------->|
|                                                                        |
|                 [ack#=101,seq#=200,data="C"]                           |
|<-----------------------------------------------------------------------|
|                                                                        |
|                 [seq#=101,ack#=201,data="i"]                           |
|----------------------------------------------------------------------->|
|                                                                        |
|                 [ack#=201,seq#=102,data="i"]                           |
|<-----------------------------------------------------------------------|
v                                                                        v
```

- receive window   16-bit (size of window on receiver end)

- header length / offset   4-bit (size of TCP header in 32-bit words)

- optional fields   variable length (may be used to negotiate protocol parameters)

- `URG` flag   1-bit (informs receiver some data is marked as urgent)

- **ACK** flag   1-bit (value contained in the acknowledge number is a valid acknowledgement)

    See **acknowledgement number** above, and the **three-way handshake** section.

- PSH flag   1-bit (solicit receiver to pass data to application immediately)
- RST flag   1-bit (used during connection setup and shutdown)
- **SYN** flag   1-bit (used during connection setup and shutdown)

    See the **three-way handshake** section below.

- **FIN** flag   1-bit (used during connection shutdown)

    A client sends a TCP segment with FIN set, and the server responds with ACK, and then the server sends a FIN TCP segment, client then responds with ACK. Closing a connection is simpler than initialising one.

- checksum   16-bit (used to detect transmission errors)

Three-Way Handshake

The three steps are as follows;

1. **client** sends a TCP segment with SYN set, and an initial sequence number
2. **server** responds with another TCP segment with SYN set, as well as ACK, the first unseen client sequence number, and the ISN for the server
3. **client** reserved with a TCP segment with ACK set, the first unseen server sequence number, and the client's new sequence number


```
client                                                              server
         [SYN,seq#=cli_init_seq]
  |------------------------------------------------------------------->|
         [SYN,ACK,ack#=cli_init_seq+1,seq#=srv_init_seq]
  |<-------------------------------------------------------------------|
         [ACK,seq#=cli_init_seq+1,ack#=srv_init_seq+1]
  |------------------------------------------------------------------->|
                              ⋮
```

User Datagram Protocol

UDP only provides the two most basic functions of a transport protocol, which are application identification (multiplexing and demultiplexing), and an integrity check via a CRC-type checksum. There is no flow control, no error control, nor any retransmissions. The datagrams cannot be larger than 65K, there is no segmentation, and the router will just drop it. The 65K consists of 20B IP header, 8B UDP header, 65507B of data, coming to a total of 65535 bytes. In practice, only 500 to 1000 bytes are used (the smaller the datagram, the more likely it will arrive intact).

Instead of just using IP, it adds port numbers on top of it, allowing us to differentiate between applications. This is a connection-less protocol, therefore there is no need to connect (just send the data) but each datagram packet must carry the full address and port of the recipient. The UDP header consists of the following fields;

- source and destination ports   16-bit each
- length of data   16-bit
- checksum   16-bit

Any application where we care more about speed, we can use UDP, since it is generally faster (due to the lack of connection establishment), and also has a smaller packet overhead.

Berkeley Socket Interface

The **Berkeley** socket interface is as follows;

- `SOCKET`   create a new communication endpoint
- `BIND`   attach a local address to a socket

  the client and server each bind a transport-level address and a name to the locally created socket
- `LISTEN`   announce willingness to accept `N` connections

  server starts listening on this socket, thus telling the kernel it will wait for connections from clients
- `ACCEPT`   block until a remote client wishes to establish a connection

  this blocks the current thread, thus it is a synchronous operation

  from here the server can accept or select connections from clients
- `CONNECT`   attempt to establish a connection

  a client connects to the socket, it needs to provide the full transport-level address to locate the socket
- `SEND`   send data over a connection
- `RECEIVE`   receive data over a connection

  now the client and server communicate through these operations on their respective sockets
- `CLOSE`   release the connection

  end the communication, must be closed otherwise connections may be continued, and may run out of ports

Note that the part in violet would not be present for UDP.

30th January 2020   **Week 4, Lecture 2**

TCP vs UDP

The scenario is as follows; movie player app is being built that allows (paying) users to stream public domain movies. Since the users are paying, we should be using TCP, as it guarantees quality of service, whereas UDP does not. The difference in performance (TCP being slower than UDP) can be circumvented by pre-buffering (sending some data in advance for the client to store locally). This however doesn't work for live events - we can either switch to UDP, or introduce a delay between the feed and the live event.

QUIC

QUIC (Quick UDP Internet Connections) is a new layer 4 protocol, implemented by a Google engineer in 2012. Originally, it was designed for general purpose, but now it is being used for HTTP (layer 5). This is still a draft, but is actively being used by some web servers.

Finite-State Machines

A finite-state machine (FSM) is a mathematical abstraction, where states are represented as nodes, and transitions are directed edges between states, labelled with events. They are also known as finite-state automaton (FSA), deterministic finite-state automaton (DFA), and non-deterministic finite-state automaton (NFA) - multiple edges with the same label from one node. This can be used to specify protocols, where the states represent the state of a protocol, and transitions are characterised by an event / action label (event typically consists of an input message or timeout, whereas an action typically consists of an output message).

$$\frac{\text{input / timeout (event)}}{\text{output (action)}}$$

Examples of this are in the slides, for the TCP state machines for both the client and server.

Reliable Data Transfer

TCP adds a reliable channel on top of an unreliable channel (IP) which is "best effort" delivery. The service provided by the transport layer sits on top of the service provided by the network layer.

Another issue is transmitting data through a noisy channel. There is a chance no packets will be lost, but a a bit may be modified during the transmission (flipped). The stages of dealing with this are as follows;

- **error detection**   receiver must know when a received packet is corrupted

    Some strategies for detecting errors are as follows;

    - sending redundant information

        sending information twice, report an error if two different messages received - inefficient
    - error-detection codes

        parity bit, adding a single bit at the end that is the XOR of all other bits in the message, if the recomputed XOR (by receiver) is different, then an error occurred

- **receiver feedback**   receiver must be able to alert sender that a corrupted packet was received

    `ACKs` and `NACKs` are also protected with an error-detection code, with corrupted `ACKs` being treated as `NACKs`. This may possibly generate duplicate segments, however sequence numbers allow the receiver to ignore these data segments.

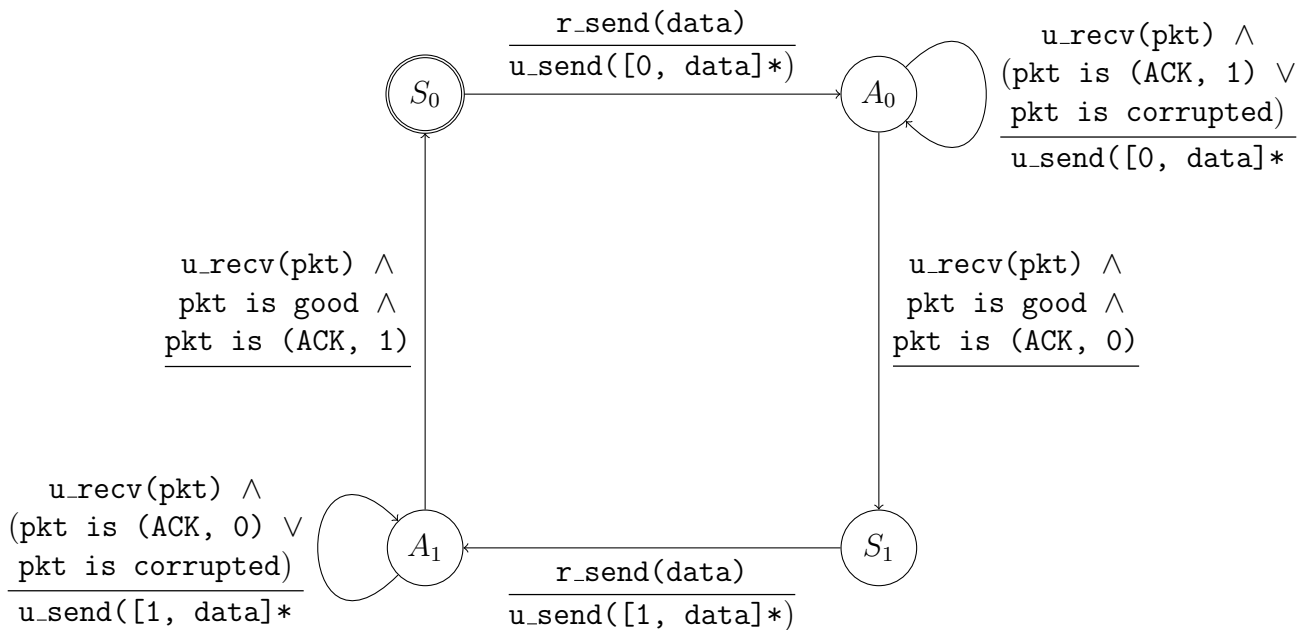- **retransmission**   the sender must retransmit corrupted packets

Note that in the diagram below, `[data]*` indicates a packet containing `data` and an error-detection code, and A represents the acknowledgement state.
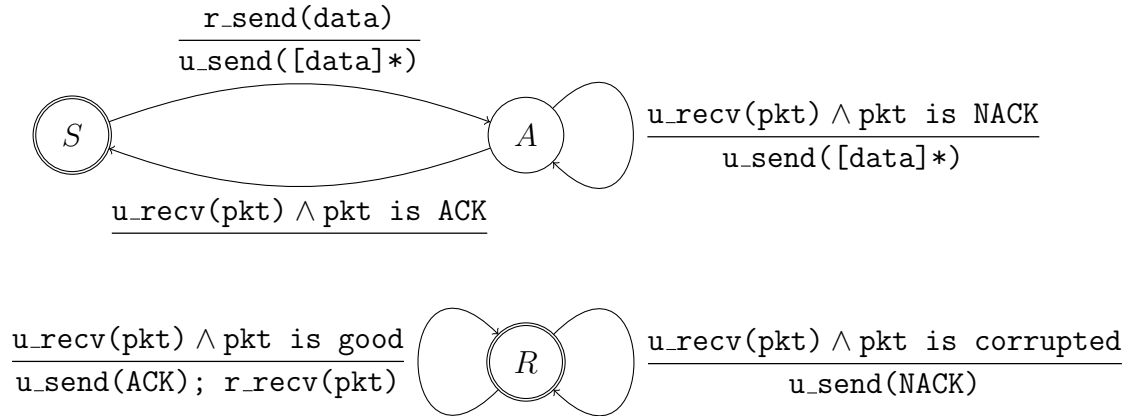
This is a synchronous (or stop-and-wait) for each segment, as the sender must receive a positive acknowledgement before it can take more data from the application layer. However, this can have issues when the `ACK/NACK` packets have issues. A way around this is for the sender to add a sequence number to each packet, thus allowing the receiver to determine whether a packet is a retransmission. If the sender does not hear an acknowledgement, due to some failure, it assumes a `NACK`, and retransmits. However, if the receiver did actually send an acknowledgement, it would ignore the packet, since it knows it is a retransmission. It then resends the `ACK`, and doesn't process the retransmitted.

Since this is a stop-and-wait protocol for each segment, only one bit is needed for the sequence number - as all we need to do is distinguish between the next segment and the retransmission of the current segment.

However, now that we have sequence numbers, we don't need both `ACK` and `NACK`, since we can convey the same semantics by sending an `ACK` for the last good packet it received.

u_recv(pkt) ∧
pkt is good ∧
snd seq_num(pkt) is 1
―――――――――――
u_send([ACK, 1]*)

R_0

u_recv(pkt) ∧
pkt is corrupted
―――――――――――
u_send([ACK, 1]*)

u_recv(pkt) ∧
pkt is good ∧
seq_num(pkt) is 1
―――――――――――――――――
u_send([ACK, 1]*); r_recv(pkt)

u_recv(pkt) ∧
pkt is good ∧
seq_num(pkt) is 0
―――――――――――――――――
u_send([ACK, 0]*); r_recv(pkt)

u_recv(pkt) ∧
pkt is good ∧
snd seq_num(pkt) is 0
―――――――――――
u_send([ACK, 0]*)

R_1

u_recv(pkt) ∧
pkt is corrupted
―――――――――――
u_send([ACK, 0]*)

Acknowledgement Generation

The following are cases of segment arrival, and how the protocol handles them;

- arrival of in-order segment with expected sequence number (all data up to expected sequence number already acknowledged)

    delayed ACK, wait up to 500ms for another in-order segment, if it doesn't arrive, send ACK for just this segment

- arrival of in-order segment with expected sequence number (one other in-order segment awaiting ACK, from above case)

    immediately send cumulative ACK for both segments

- arrival of out of order segment with higher-than-expected sequence number (gap detected)

    immediately send duplicate ACK - the sender will know to fill in the gap, as it would've already seen this acknowledgement

- arrival of segment that fills a gap in the received data

    immediately send ACK if segment starts at lower end of gap

Additionally, it's important to note that lost packets can easily be treated as corrupted packets. Since I don't really want to draw the FSM again, add a start_timer() after every u_send(...), and an additional transition loop to A_0 and A_1, which sends the same data, but the input is timeout.

There's also a small section after this on the alternating bit protocol, which isn't used in practice.

30th January 2020 $\hfill$ Week 4, Lecture 3

Congestion Detection

If the queue of one or more routers between the sender and receiver overflow, we call it **congestion** - this has the visible effect of segments being dropped. The server can assume the network experiences congestion when it detects a segment loss; either on a timeout, hence no ACK, or multiple acknowledgements, which is equivalent to NACK.

TCP we've described so far is referred to as TCP Reno. Another popular implementation is TCP Vegas, which aims to detect congestion before losses occur. It does this by predicting imminent packet loss via observing the RTT - the longer it is, the greater the congestion in the routers. However, flows

with small RTTs are advantaged, compared to the ones with large RTTs, as their window can grow faster. TCP CUBIC circumvents this by making the window increase as a function of time rather than RTT.

Congestion Window and Congestion Control

The congestion window, W, is maintained by the sender, and limits the amount of bytes that the sender pushes into the network before it blocks to wait for acknowledgements.

$$\text{LastByteSent} - \text{LastByteAcked} \leq W = \min(\text{CongestionWindow}, \text{ReceiverWindow})$$

This gives a resulting maximum output of roughly

$$\lambda = \frac{W}{\text{RTT}}$$

The phases are as follows;

- **slow start**

    The initial value of W is the maximum segment size (MSS), which is quite low for modern (high-speed) networks. To result in a good throughput, TCP increases the sending rate exponentially for its first phase. It increases W by 1 MSS on every (positive) segment acknowledgement (doubles each time - the first `ACK` increases it by 1 MSS, the second `ACK` will be after sending 2 segments, hence it increases by a further s2 MSS, and so on), until it reaches some slow start threshold (ssthresh), or until congestion occurs. If W is greater than (ssthresh), then use congestion avoidance (if they are equal, either could be used).

- **congestion avoidance**

    TCP then increases W linearly in this phase.

    $$W = W + \text{MSS} \cdot \underbrace{\frac{\text{MSS}}{W}}_{\text{increment}}$$

    This happens until congestion is detected.

- **Additive-Increase / Multiplicative-Decrease (AIMD)**

    - at every good acknowledgement, increase W in accordance with congestion avoidance $\left(\frac{\text{MSS}^2}{W}\right)$
    - at every packet loss event, TCP halves the congestion window

        There are two possibilities for packet loss. TCP provides reliable data transfer by using a timer to detect lost segments. If there is a timeout, without an `ACK`, it means that there is likely a lost packet, hence a retransmission is required. This timeout interval, T, must be larger than the RTT to avoid unnecessary retransmissions, however it shouldn't be too far from RTT, as we want to detect and retransmit lost segments as quickly as possible. TCP sets its timeouts as follows, where (1) is the estimated RTT, and (2) is the variability estimate;

        $$T = \underbrace{\overline{\text{RTT}}}_{(1)} + 4 \cdot \underbrace{\overline{\text{DevRTT}}}_{(2)}$$

        The timeout is controlled by continuously estimating the current RTT.

        The agreement is that 3 duplicate `ACK`s (hence 4 identical `ACK`s overall), are interpreted as a `NACK` in **fast retransmit**. While both timeouts and `NACK`s signal a loss, they say different things about the status of the network, hence TCP should react differently. A timeout indicates congestion, whereas the duplicate `ACK`s suggests the network is able to still able to deliver segments along that path.

        In the formulae below, assume the current window size is $W = \overline{W}$;

* timeout

  In the event of a timeout, set $W = \text{MSS}$, and run slow start until W reaches $\frac{\overline{W}}{2}$ (which is the new ssthresh), and then proceed with congestion avoidance.
* NACK

  Similarly set ssthresh to $\frac{\overline{W}}{2}$, but set $W = \frac{\overline{W}}{2}$. We then continue to run congestion avoidance, which increases W linearly - **fast recovery**.

Selective Repeat

The sender only transmits packets it suspects were lost or corrupted. However, if we were to use a TCP flavour that uses this, we'd need powerful routers in the path, as it requires storage and processing.

- sender maintains a vector of acknowledgement flags
- receiver maintains a vector of acknowledged flags

    maintains a buffer of out-of-order packets
- sender maintains a timer for each pending packet
- sender resends a packet when its timer experiences
- sender slides the window when the lowest pending sequence number is acknowledged

window size (W)
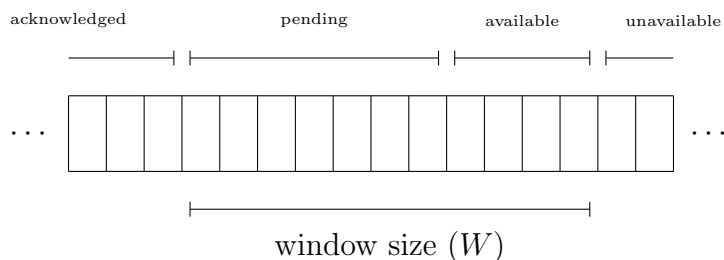
Flow Control vs Congestion Control

The former aims to not overflow the receiver, whereas the latter aims not to overflow the network. An example of flow control is as follows;
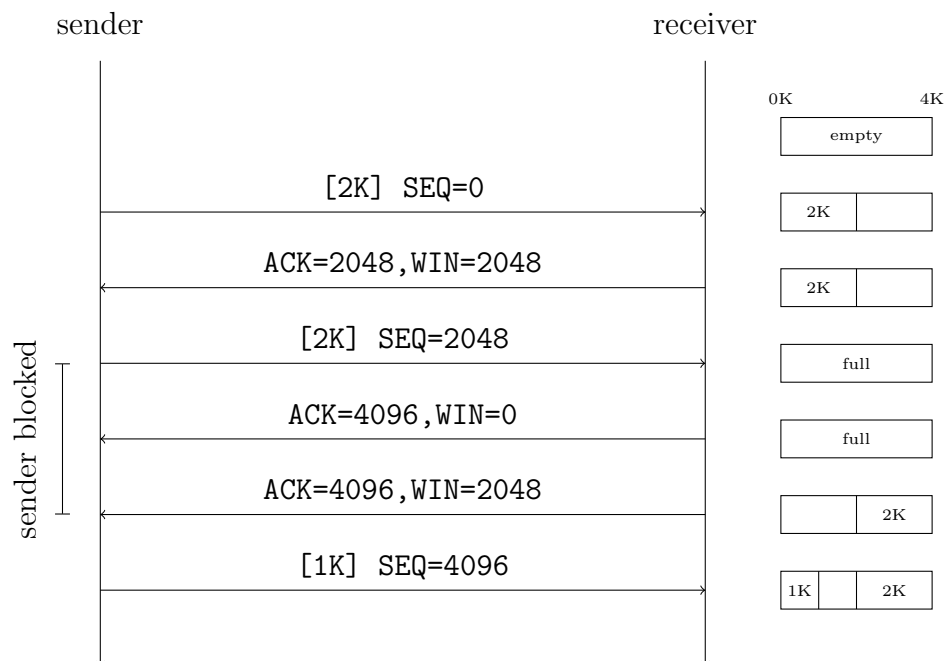
The receiver sends the maximum number of bytes that may be sent (ReceiverWindow) along with the acknowledgement - a size of 0 is legal, and it means no more messages can be sent, except a 1-byte segment to ping receivers.

Wireless TCP

TCP assumes that IP is running across wires. Therefore, when packets are lost, it assumes that this is due to congestion and slows down, whereas in wireless environments, packets are usually lost to channel reliability issues (where TCP should actually try harder).

One solution it to split TCP connections to distinguish between wired and wireless IP. Another solution would be to allow the base station to do some retransmission, without informing the source (if it still fails, the source is then informed).

Network Usage

We can quantify network usage with the utilisation factor;

$$\frac{\text{how much we actually used the network}}{\text{how much we could have used it}}$$

For example, if we were to rent a 24Mbps line, and download at 2MBps, we are at around 67% utilisation (note the conversion from bits to bytes). This may be due to congestion control and / or flow control, as it may not be possible for all nodes in the route to support the amount of traffic you wish to send.

If we have all the data, we can use the following formula for utilisation;

$$U = \frac{\frac{L}{R}}{\text{RTT} + \frac{L}{R}}$$

For example, if we have a RTT of 30ms, a channel with transmission rate $R = 1$Gbps, and a packet size $L = 1000$bits, we have a utilisation of $U \approx 0.027\%$.

4th February 2020 Week 5, Lecture 1

H/P/V/A/C

- **hackers**

  Originally meant highly competent computer engineers who explore different ways of using / combining things. Can be divided into the following;

  - white hat   a researcher who informs the company of problems before they go public
  - grey hat   an analyst who does not inform anyone unless they get paid
  - black hat   a malicious individual who abuses the findings to cause harm

- **phreakers**

  Phone hackers - not as prevalent anymore, due to the telephone network being almost fully digital. Playing the right frequencies allowed for access to administrative interface for a phone network.

- **virii (creators)**

  Create viruses, for curiosity or money. Popular viruses include ransomware, spyware, and Trojans (remotely controlled botnet, and profit from renting). A rootkit is injected into the root account, running as the superuser, thus having more control.

- **anarchists** "hacktivists"
- **crackers**

  Originally meant wannabe hackers (script kiddies), who use the tools of others to infiltrate systems.

Methods and Tools

Some commonly used methods are as follows;

- **credential reuse / stuffing**   leaked `email:password` combinations tested on other services
- **network monitor / packet sniffing**   reading messages not intended for your NIC
- **code / SQL injection**   running your own code in someone else's system
- **session / cookie hijacking**   use another session without credentials
- **wardriving**   searching and using unsecured WiFi spots
- **dumpster diving / trashing**   physically checking trash for information
- **clickjacking**   forcing users to click on hidden links
- **bat and switch**   legitimate-looking ads leading to malicious destinations
- **spoofing**   IP (layer 3), MAC (layer 2), DNS poisoning

Some commonly used tools are as follows;

- **rootkit**   runs as an administrator (can reinstall itself)
- **keyloggers**   allows attackers to record all keypresses
- **trojans**   allows attackers to remote control an entire system (similar to *TeamViewer* or VNC)
- **TAILS**   The Amnesic Incognito Live System - forgets on reboot
- **Kali Linux**   large collection of security and pentesting tools
- **Metasploit**   identifies systems and their vulnerabilities (similar to nmap)

Laws and Standards

Basically just a long list of laws (and their years), and organisations.

- Computer Misuse Act (1990)   most intrusions fall under this
- Copyright, Designs, and Patents Act (1988)   piracy
- Criminal Justice Act (2003)
- Data Protection Act (2018)
- Defamation Act (2013)   cyber-bullying
- Disability Discrimination Act (2005)
- Freedom of Information Act (2000)   can be used against companies
- Obscene Publications Act (1964)
- Protection of Children Act (1978)   1999 version still being amended
- Regulation of Investigatory Powers Act (2000)

Note that an intruder is trialled in both the country they reside in, as well as the laws of the country where the involved hosts are physically located.

- IANA (Internet Assigned Numbers Authority)   sets these standards, and assigns port numbers
- ICANN (Internet Corporation for Assigned Numbers and Names)   names for corporations
- IETF (Internet Engineering Task Force)   attempt to set standards straight
- ISOC (Internet Society)   push governments to update laws
- EFF (Electronic Frontier Foundation)   push rights (free speech and equality) online
- W3C (World Wide Web Consortium)
- ISO (International Organisation for Standardisation)

Examples

Some examples include Heartbleed, KRACK, and some other legal / ethical grey areas. GDPR forces companies to inform their users of hacks within 72 hours.

Network Security Issues

We want control access over a channel between a user and a resource. We need a way to ensure only specific users can access a resource (access control), as well as a way to authenticate the user (and resource). Additionally, confidentiality also has to be ensured to limit access to information owned by users, as well as integrity (actions of a user should not be able to affect the overall integrity of a resource). Users also cannot deny communication took place (non-repudiation) - blockchain is an example. To achieve this we need;

- access control
- security policy
- secure channel (users and data are authenticated, information exchanged is confidential)
- monitoring / logging / auditing

Access Control

```
┌───────────┐  request   ┌─────────┬──────────┐
│ principal │───────────▶│  guard  │ resource │
└───────────┘            └─────────┴──────────┘
```

Assuming a secure channel, the guard controls which principals can access the resource, where principals are allowed to be located (access within networks), and what requests (actions) principals are allowed to make.

This is done via the use of a **firewall**, which controls access to the network by providing a security gateway between the internal and external networks.

- **application-level gateway**   runs on the host, and only protects that host (such as iptables)

  This is in a logical connection, allowing it to monitor traffic. This means that it can block / filter / report based on application level message content (as well as scan for data leaks, viruses, etc). Data can also be rewritten, therefore we do not want this to be compromised. If the data is encrypted, it may ask for the ability to read it, thus decrypting it, and then re-encrypting it for the client.

- **proxy server**   runs on network, and can protect entire LAN

  Proxies can filter incoming / outgoing traffic in different modes;

    - **normal**   client is aware (and needs to be set up)
    - **transparent**   client is unaware (local router takes care of everything)
    - **reverse**   runs on receiving side, impersonates servers (CDN load balancing)

  SOCKS proxies can be used for any protocol, whereas HTTP is for web.

- **circuit-level gateway**   takes over host's communication as a non-caching proxy (such as Tor)
- **stateless packet filtering**   checks source / destination IP addresses and ports
- **stateful packet filtering**   checks contents of current and previous packets

  It can also perform user authentication. Connections can be dropped based on destination, incorrect packets, time, or volume. This is done with inspection to the layer 4 ports, layer 3 IP addresses, and the layer 2 MAC addresses.

The **Bastion** host is the host that all public connections arrive to. It should expect to be attacked, and perform auditing / logging on a secure OS.

The is set by using **access control lists**, which contain packet filtering rules. The firewall scans through the list of rules, until it finds one that matches, and performs the actions. The rules contain a number to uniquely identify them, a direction (incoming or outgoing), an action (block or allow), as well as two IP and port pairs for the inside and outside. Usually, the last rule is to block everything else.

Other Systems

- IDS (Intrusion Detection Systems)   detects intrusions (such as identifying a DDoS attack) doesn't stop the intrusion, only informs the system
- IPS (Intrusion Prevention System)   actively prevents intrusions (such as blocking `SYN` floods) either includes an IDS, or works with an IDS
- NGFW (Next Generation Firewall)   a (stateful) firewall with an IPS/IDS system
- UTM (Unified Thread Management)   similar to NGFW but more capabilities (e.g. antivirus)

DMZ (Demilitarised Zone)

This is the part of the network accessible to external hosts. All sensitive servers should lie behind the DMZ. The router uses NAT (Network Address Translation) to get external messages to the correct internal host - if we want to expose an internal host without putting it in the DMZ, port forwarding can be used.

Port forwarding lets the router know that packets for certain ports should be directly forwarded to an internal host's port. For example, if we tell the router to forward any packet at port 12345, it should be forwarded to the NAT-based LAN IP of an internal host at port 80, accessing `http://publicip:12345/` is actually accessing `http://internalhost:80/` indirectly, since the internal host does not have a public IP.

Getting Around Firewalls

Methods of getting around a firewall include ssh tunnelling, which uses another server to act as a proxy between services; VNC can also be used in a similar way. Another method is to spoof the software MAC address - if a firewall was to allow a certain address through. Similarly spoofing an IP could be attempted, but a stateful firewall will most likely catch this.

Security Policy and Logging

Each company or organisation needs to define its Network Security Policy, which includes (but is not limited to);

- intranet / internet access rights
- allowed use of software (whether the user should be allowed to install software) / hardware (whether the user should be allowed to use their own hardware)
- risk assessments
- training

Systems are likely already keeping logs, and these can be used to identify system or network issues, as well as being used evidence of criminal activities (if the log files haven't been deleted). Anything useful identified by an audit should be permanently stored.

Cryptography

Cryptography is the process of writing something in code (encode or encrypt), with a specific algorithm, whilst retaining the ability to retrieve it afterwards (decode or decrypt). The latter is only possible in the case of encryption (see **hashing** later on). We assume that the physical channel is open to attack from the enemy, who can read and alter any bit pattern.

Encryption
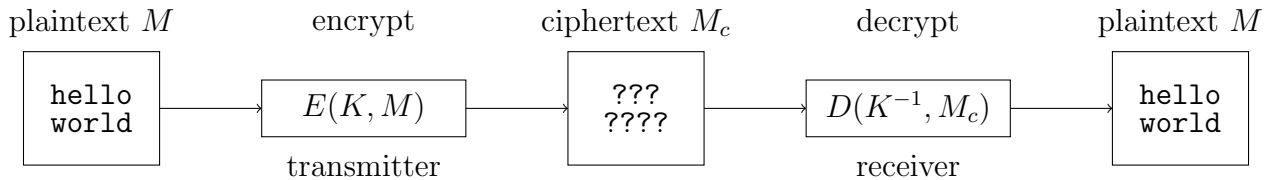
Only certain users should have K (the encryption key), and K^{-1} (the decryption key). If someone were to intercept M_c, without K^{-1}, the attacker can only find M by enumerating all possible K^{-1} via a brute-force attack. Another property is that it should be difficult to obtain the values of K and K^{-1} with just M and M_c.

- secret key (symmetric) encryption has $K = K^{-1}$, hence the same key is used to encrypt and decrypt the file, thus the key must be very carefully distributed otherwise all files can be decrypted

- public key (asymmetric / public-private key pair) encryption has $K \neq K^{-1}$

  K_t is the private-key of host H_t, and is only stored / used by that host. On the other hand, K_t^{-1} is the public-key of the same host, and is freely distributed. Therefore anything encrypted with K_t^{-1} can only be decrypted by the host H_t. This also means anyone can decrypt something encrypted by H_t with K_t - if it is successful, it authenticates the message as coming from H_t.

  In this example, let H_t have its own private key K_t, as well as H_r's public key K_r^{-1} (and vice versa for H_r). H_t first signs the message with its own private key, $E(K_t, M)$. It then encrypts the message with the destination's public key, $E(K_r^{-1}, E(K_t, M))$ - ensuring only H_r can read the message. Now, **only** H_r can obtain the signed message as follows;

  $$D(K_r, E(K_r^{-1}, E(K_t, M))) = E(K_t, M)$$

  Additionally, we can validate that the sender was actually H_t with H_t's public key K_t^{-1};

  $$D(K_t^{-1}, E(K_t, M)) = M$$

We can compare the two approaches as follows;

public key (asymmetric)	secret key (symmetric)
1. owner of private-key does not need to disclose its value to communicate	1. owner of private-key needs to disclose its value in order to communicate
2. more secure	2. less secure
3. slow encryption / decryption	3. faster encryption / decryption
4. example: RSA	4. example: AES

Diffie-Hellman Key Exchange

An approach for this, to establish a new key on the spot, is the *Diffie-Hellman key exchange*, which can then be used for encryption by symmetric algorithms. This can potentially be defeated by powerful systems, but is still used by TLS (formally SSL). The (simplified) algorithm is as follows;

1. Bob and Alice agree on a public value g (generator), and a large prime number p

2. Bob chooses a secret value b, and Alice chooses a secret value a (these are not shared) - both values must be lower than p

3. The secret values are used to calculate their public values (which are then exchanged);

   - bob: $x = g^b \bmod p$
   - alice: $y = g^a \bmod p$

4. They use each other's public value to calculate the shared secret key - which can be used to communicate securely;

   - bob: $y^b \bmod p = (g^a \bmod p)^b \bmod p \equiv g^{ab} \bmod p$
   - alice: $x^a \bmod p = (g^b \bmod p)^a \bmod p \equiv g^{ab} \bmod p$

5. Eve, the eavesdropper, cannot guess / derive the shared secret key, even if they have access to g, p, x, y - since they don't know either of the secret values.

An example of this is as follows;

$$a = 4$$
$$b = 5$$
$$g = 6$$
$$p = p$$
$$(\text{Bob}) \; x = (g^b \bmod p)$$
$$= 6^5 \bmod 97$$
$$= 7776 \bmod 97$$
$$= 16$$
$$(\text{Alice}) \; y = (g^a \bmod p)$$
$$= 6^4 \bmod 97$$
$$= 1296 \bmod 97$$
$$= 35$$
$$(\text{Bob}) \; y^b \bmod p = 35^5 \bmod 97$$
$$= 52521875 \bmod 97$$
$$= 61$$
$$(\text{Alice}) \; x^a \bmod p = 16^4 \bmod 97$$
$$= 65536 \bmod 97$$
$$= 61$$

Key Server (Kerberos)

Another approach is to have a trusted key server, which the host already knows how to communicate with safely.

Kerberos (Needham and Schroeder) is a user authentication system which knows your password. It also knows the password of the user / resource you want to communicate with. It has a KDC (key distribution centre) that can provide tickets, allowing you to communicate with the desired user / resource, which expire after a predefined amount of time.

Hashing

When a message is hashed, the original message cannot be retrieved - it one way. A hash is a fixed size alphanumeric string, calculated based on a specific input - the hash function should always produce the same hash value with the same input. Since it should not be possible to derive the input from the hash value, this can be used for safely storing passwords.

11th February 2020 Week 6, Lecture 1

The scenario is as follows; an IoT research and design institute will receive government funds for a new device used by the police.

- web servers running Apache 2.4.27 with OpenSSL 1.0.1 on Red Hat Enterprise Linux 7.3
- many desktop computers available, running Windows Vista

    they are slow, so many staff members bring their own MacBooks to work

- reception desk on ground floor, with server room at back of floor secured by RFID card reader
- research team on first floor
- design team on second floor
- HR team shares third floor with CEO and CTO
- lower ground level used as storage for technical equipment and parking space
- all desktops have wired Ethernet

    CEO and CTO have company laptops, wireless AP added on top floor for LAN and Internet connection

- employees are planned to be provided with iPads, so they can use iCloud to work from anywhere
- installing more access points throughout building

Some identified security vulnerabilities are as follows;

- server room easily accessible (move to basement)
- use biometric scanner instead of RFID
- LAN and Internet connected means local network could be exposed
- wireless network can be accessed without being in the building (packet sniffing)
- sensitive data should not be stored on an external service (iCloud)
- OpenSSL 1.0.1 is vulnerable to Heartbleed
- Apache 2.4.27 (Optionsbleed) and RHEL 7.3 (Local Privilege Escalation Bug) are all broken (CVE can be easily searched)
- Windows Vista
- no building security overnight?
- employees should not be using personal devices
- third-party repairs for MacBooks may expose data
- visitor parking area also has technology
- insecure connections if working from home (VPN should be used)
- employees could be blackmailed or phished (social engineering)
- "evil twin" for wireless AP
- flood risk for ground floor server room

13th February 2020 Week 6, Lecture 2

This lecture is the Wireshark practical (enable Promiscuous Mode and Monitor Mode). Promiscuous mode works for both wired and wireless connections, and tells the NIC to not drop any packets (for a wireless network; it only focuses on the connected network). Monitor mode only works on wireless networks, which tells the NIC to start listening in on all networks - any WiFi network with a password will have encrypted data. Most NICs do not support this, or will need different drivers.

With Wireshark, we can add display filters to the captured packets to select the ones we are interested in, such as;

- `http.request.method == GET`
- `http contains "password"`
- `ip.src == 192.168.9.220`

13th February 2020 Week 6, Lecture 3

This continues on from the last lecture, with the Wireshark practical. Change the root password for a Raspberry Pi - and ensure new users haven't been created (or keys added).

<p style="text-align:center">"skrrrrr" - <code>root</code></p>

18th February 2020 Week 7, Lecture 1

Introduction to the Network Layer

This is also referred to as the Internet Layer, and contains the Internet Protocol (IP). At this layer, data travel in IP datagrams (or packets), and data are fragmented into packets, with a layer 3 header added in front of each.

A router (layer 3) typically has a line in port, which is connected to the ISP via Ethernet, a phone cable, or co-axial. There may also additional Ethernet ports to connect devices multiple devices to the local network, via the use of a switch (layer 2) within the router. A router is a fundamental component of the network layer (they are the nodes in the graph), and have a finite set of input / output connections. It has interfaces (or physical ports - not the same as layer 4 ports), which are individual network interface controllers, and therefore a router can have multiple IP addresses.
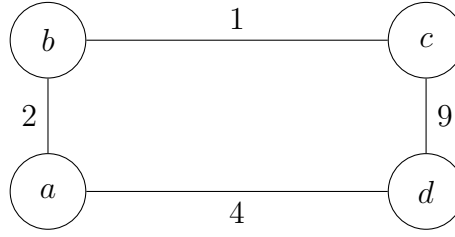
Routing

The network layer provides facilities to get data from a source to a destination, and may require making many **hops** at at intermediate routers along the way (frames in the data link layer are moved only within the same network) - this is routing. To do this, it must know the topology of the network to choose appropriate paths. Load balancing may be used to prevent congestion. This layer also has to deal with network heterogeneity, as there may be very old routers still in the network.

Datagram Networks and Forwarding

IP is connectionless (each message is routed through the system independent of others, there is also no connection setup or tear-down phase) and packet-switched (each datagram is forwarded from one router to the next based on forwarding decisions). This is a best-effort service, with no guarantees on delivery, maximum latency, bandwidth, in-order delivery nor congestion indication.

There are potentially multiple (possibly asymmetric paths) for the same source / destination. In forwarding, we have the destination of the datagram as the input, and an output port as the output. A simple design for this is a forwarding table.

Routers must cooperate to find the best routes between all pairs of nodes, and build a sink tree (or approximation) for each node. The sink tree of each node does not contain loops, and only contains the optimal routes towards that specific host.

Some strategies for routing are as follows;

- **shortest path routing**

  Shortest Path Routing (SPR) uses Dijkstra's Algorithm (see **CO150**), where the labels on arcs represent cost. At each step, we select newly reachable node with the lowest cost, and then add that edge to the tree built so far.

- **flood routing**

  In flood routing, an incoming packet is forwarded across every outgoing link (except the original source). We have to avoid too many packets in the network;

  - hop counter   discard a packet after it was forwarded across N routers
  - forward only once   avoid cycles, requires storing sequence numbers per source address
  - flood selectively   only forward in a direction that makes sense

  Flooding always chooses the shortest path because all paths are explored in parallel. This only makes sense to do when **robustness** is needed, such as in the discovery of unknown destinations.

- **distance vector routing**

  The previous protocols are static, meaning they do not take the current network load into account. DVR, or *Bellman-Ford* is a dynamic protocol that does the following;

  1. consider costs that direct neighbours are advertising to get packet to destination
  2. select neighbour who's advertised cost, added with own cost to get to that neighbour, is lowest
  3. advertise the new cost to neighbours

  The main idea is expressed by the Bellman-Ford equation;

  $$D'_u[v] = \min_{x\in\text{neighbours}(u)}(c(u,x) + D_x[v])$$

  We want to find the cost of the least cost path from u to v. This is the minimum sum of the cost between u and x, and the least cost path between x and v, where x is each direct neighbour of u. At the start, if we don't know anything, we can start with flood routing to discover neighbours.

An example of this is as follows;

(a)	a	b	c	d
D_a	0	2	∞	4
D_b	∞	∞	∞	∞
D_d	∞	∞	∞	∞
(b)	a	b	c	d
D_b	2	0	1	∞
D_a	∞	∞	∞	∞
D_c	∞	∞	∞	∞
(c)	a	b	c	d
D_c	∞	1	0	9
D_b	∞	∞	∞	∞
D_d	∞	∞	∞	∞
(d)	a	b	c	d
D_d	4	∞	9	0
D_a	∞	∞	∞	∞
D_c	∞	∞	∞	∞

(a)	a	b	c	d
D_a	0	2	3	4
D_b	2	0	1	∞
D_d	4	∞	9	0
(b)	a	b	c	d
D_b	2	0	1	6
D_a	0	2	∞	4
D_c	∞	1	0	9
(c)	a	b	c	d
D_c	3	1	0	9
D_b	2	0	1	∞
D_d	4	∞	9	0
(d)	a	b	c	d
D_d	4	6	9	0
D_a	0	2	∞	4
D_c	∞	1	0	9

(a)	a	b	c	d
D_a	0	2	3	4
D_b	2	0	1	6
D_d	4	9	9	0
(b)	a	b	c	d
D_b	2	0	1	6
D_a	0	2	3	4
D_c	3	1	0	9
(c)	a	b	c	d
D_c	3	1	0	7
D_b	2	0	1	6
D_d	4	6	9	0
(d)	a	b	c	d
D_d	4	6	7	0
D_a	0	2	3	4
D_c	3	1	0	9

However, there is a problem with DVR - good news propagates quickly, but bad news propagates slowly. Consider the following graph, and assume all costs are 1 (hence we can just look at the number of hops);
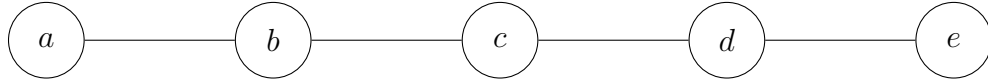
Consider the table below, where we have the distance from a;

b	c	d	e	comment
∞	∞	∞	∞	initial state
a comes online after being down				
1	∞	∞	∞	after 1 exchange
1	2	∞	∞	after 2 exchanges
1	2	3	∞	after 3 exchanges
1	2	3	4	after 4 exchanges
a suddenly goes down				
3	2	3	4	after 5 exchanges
3	4	3	4	after 6 exchanges
5	4	5	4	after 7 exchanges
5	6	5	6	after 8 exchanges
7	6	7	6	after 9 exchanges
7	8	7	8	after 10 exchanges
⋮				
∞	∞	∞	∞	

When a suddenly goes down, they keep increasing their route up to infinity (count-to-infinity problem) We can set infinity to be one more than the longest acceptable path.

- **link state routing**

This was published in 1979, aiming to replace DVR. The goal is to broadcast information about the network topology to all routers, and each router calculates the sink tree to other routers. Each router does the following;

1. discover direct neighbours and gets their network addresses

   routers know their local network interfaces, and therefore sends a `HELLO` packet through each interface - the routers on the other end responds with its address

2. calculates cost for sending packet to each neighbour

   an `ECHO` packet is sent through each interface and the RTT is measured (this gives a reasonable estimate of actual delay)

3. constructs LSA (link state advertisement) packet with all information

   done with flooding to send LSA to all other routers, once a router has LSAs from every router, it has complete knowledge of the network

4. sends that packet to all routers (as well as collecting that packet from all routers) - not limited to just neighbours

5. runs Dijkstra's algorithm locally to find shortest paths

This means that each router stores information about its neighbours, the entire network, and the generated routing table.

Both DVR and LSR are used, in order to support routers both old (DVR) and new (LSR) routers.

	distance vector routing	link state routing
network knowledge	local	global
computation	global	local
synchronisation	gradual	"instant" (after SPR calculation)

These protocols are used internally, and therefore a specific one can be chosen for a local network.

Hierarchical Routing

All of the previously discussed routing algorithms require each router to know about the others - this cannot scale, as it becomes too demanding in terms of memory and processing power. A solution to this is to introduce regions, and have separate algorithms for intra-region (within the region) and inter-region (between regions) routing. Two or three levels is generally sufficient. We no longer take an optimal route - we just know where to forward packets.

Broadcast Routing

If we want to send a message to (almost) every host on a network, we can take the following approaches;

- message each host individually   inefficient
- flood routing   accepted, if we can limit the flood
- multi-destination routing via a list sent with the packet

  router checks destination and splits list when forwarding to different interfaces - the message must contain all destinations.

- build sink tree at source and use that for multicast route

  must be spanning tree, and routers need to agree on trees

We can construct this spanning tree with reverse-path forwarding (RPF) broadcast. Every router forwards / broadcasts a packet to every adjacent router, except where it came from. A router only accepts a broadcast packet p originating at A if p arrives on a link that is on a direct (unicast) path between themselves and A.

Multicast Routing

We want to send a message to only a subset of nodes, and we need to know when a host enters or leaves a multicast group. The solution to this is to construct a spanning tree at each router, and use a group ID to prune paths that do not contain members of group.

The main issue with this is scalability - as we need to maintain a spanning tree per source. Another approach is to use core-based trees, and have a single spanning tree per group, with the root (core) near the middle of the group. For a multicast message to be sent, the host sends it to the core which then performs multicast along the tree.

20th February 2020 Week 7, Lecture 2

Internetworking

Internetworking is the process of linking different types of networks together. All of these different networks have different protocol stacks, and need to somehow communicate with each other. Instead of forcing all networks to run the same stack, we can construct gateways (routers) that interconnect different types of networks. Different types of networks include, but are not limited to;

- **PAN** (Personal Area Network) such as phone connected to PC via Bluetooth (1 user)
- **LAN** (Local Area Network) such as laptop connected to desktop via WiFi AP
- **MAN** (Metropolitan Area Network) such as displaying bus times without internet (city)
- **WAN** (Wide Area Network) such as the Internet

Additionally, we have the following terminology for devices;

- repeaters / hubs physical layer

  boosts signals by repeating everything received at one port to every other port
- switches and bridges data link layer

  creates interconnections by storing a table, and deciding where to forward based on MAC addresses
- multi-protocol routers / gateways network layer

  forwards (and possibly splits up) packets - makes decisions based on IP addresses (note that Transport Layer Gateways and Application Layer Gateways also exist)

The Internet

The internet is a collection of autonomous systems (ASes) connected by backbones.

An application offers a data stream to the transport layer, using either connection-oriented or connection-less services. The data stream on the transport layer is converted into an IP datagram for the network layer. These are routed through the internet, and routers pass datagrams to the underlying data link layer (LANs and WANs). After this, data is in the physical layer, transmitted through cables.

At the Internet Network Layer, the following needs to be provided;

- **IP** (Internet Protocol)

  this includes datagram format, fragmentation, addressing, and packet handling
- **ICMP** (Internet Control Message Protocol)

  this provides error handling, and signalling (such as `ping`)
- **DHCP** (Dynamic Host Configuration Protocol)

  this provides address configuration

Fragmentation

Similar to how TCP includes the sequence number in the header, the layer 3 (IP) header includes the fragment offset from the start of the IP datagram. Fragmentation occurs when the router encounters the case where the size of the input datagram exceeds the MTU (maximum transmission unit) of the output link.

Only the destination reassembles fragmented datagrams, as this pushes complexity out of network. Along the path, the datagram may have to be fragmented further. The fragmentation scheme must ensure the destination host can figure out if and when all fragments were received, as well as recognising two fragments of the same datagram.

Assume an initial (non-fragmented) datagram of size X. The sender assigns a 16-bit identifier to the datagram, and sets the fragment offset (offset in units of 8 bytes) to 0, which indicates this packet contains data starting at position 0 of the original datagram. Note that the fragment offset field is 13 bits, hence there can be $2^{13} = 8192$ (0 to 8191) units of 8B in one fragment. Note that all fragments should be multiples of 8B, except the last one.

However, the total length field is 16 bits, therefore it can only support a maximum size of 65535B. The IP header itself is 20B, meaning that there are 65515B available, which means that there are 8189 possible units of 8B, and an additional unit of 3B. If the more fragments flag is set to 1, more fragments are expected to follow.

An example of this is as follows - assuming an MTU of 512B, and a total packet size of 1020B;

identifier	fragment offset	more fragments	length	header length	total length
789	0	1	488	20	508
789	61	1	488	20	508
789	122	0	24	20	44

IPv4 Addressing

In IPv4, the addresses are in the format `XXX.XXX.XXX.XXX`, where `XXX` is from 0 to 255 - they are 32-bit, and thus there are 4 billion possible IP addresses. Each address is associated with an interface, not host, and therefore a host with multiple interfaces may have more than one IP.

The key idea is to assign addresses with the same prefix to interfaces belonging to the same organisation. Originally, this was done with classful addressing, however this is no longer used as it became apparent that it would quickly run out, due to companies buying more addresses than needed. Note in the table below, the leftmost bit is bit 0.

class	prefix	network bits	max networks	host bits	max hosts	range of host addresses
A	0	1–7	126	8–31	16,777,214	1.0.0.0 to 127.255.255.255.255
B	10	2–15	16,382	16–31	65,536	128.0.0.0 to 191.255.255.255.255
C	110	3–23	2,097,150	24–31	254	192.0.0.0 to 233.255.255.255.255
D	1110	multicast address				224.0.0.0 to 239.255.255.255.255
E	1111	reserved for future use				240.0.0.0 to 255.255.255.255.255

Nowadays, IP addresses are managed by ICANN to avoid conflicts. It delegates parts of address space to regional authorities, which give out IP addresses to ISPs and companies.

Subnet Masking

Hosts on the same network must share the same network address. The solution to this was to use a single network address for the entire organisation, and divide internally into subnet addresses and host IDs. This divides the address into the class, network, subnet, and host, thus introducing a three-level routing hierarchy;

1. external routers only consider the network address, and forward the packet to one of the routers of the organisation

2. subnet routers apply the subnet mask, and check if the destination is on their subnet, or if they should forward it to another subnet router

3. after host is found, router knows which interface to use to forward packet

All interfaces in the same subnet share the same address prefix. If we were to get rid of the classes entirely; we will have a notation called classless inter-domain routing (CIDR), which has the format `address/prefix-length`. For example, `123.1.1.0/24` means that all addresses share the same left-most 24 bits with `123.1.1.0`. This scheme is not limited to entire bytes.

For example; if we were to be asked for the range of addresses in `128.138.207.160/27`;

$$\overbrace{\texttt{10000000 \ 10001010 \ 11001111 \ 101}}^{\text{subnet}}\texttt{00000}$$

```
10000000 10001010 11001111 10100000
10000000 10001010 11001111 10100001
10000000 10001010 11001111 10100010
10000000 10001010 11001111 10100011
                  ⋮
10000000 10001010 11001111 10111110
10000000 10001010 11001111 10111111
```

We would have a range of `128.138.207.160` - `128.138.207.191`.

The mask notation is `address/mask`, and a prefix of length p corresponds to the following mask;

$$M = \overbrace{\texttt{11} \cdots \texttt{1}}^{p \text{ times}} \underbrace{\texttt{00} \cdots \texttt{0}}_{32-p \text{ times}}$$

Therefore we have the following conversions (example);

$$\texttt{128.207.160/27} = \texttt{128.207.160/255.255.255.224}$$
$$\texttt{127.0.0.1/8} = \texttt{127.0.0.1/255.0.0.0}$$
$$\texttt{92.168.0.3/24} = \texttt{92.168.0.3/255.255.255.0}$$
$$\texttt{195.176.181.11/32} = \texttt{195.176.181.11/255.255.255.255}$$

When we (bitwise) `AND` the address with the mask, we obtain the network address.

Longest-Prefix Matching

Routers attempt to match the maximum possible prefix in order to select the correct network. They (bitwise) `AND` the IP address with its subnet mask to obtain the network address. If they know where this is (based on the router's forwarding table), it will forward it out of the correct port. Otherwise, it depends on the algorithm (or a default port where packets for unknown networks go to).

ICMP and NAT

ICMP encapsulates control messages in IP packets, and each type is further specified by a code. This may be blocked on some routers, as it can be abused, thus a UDP implementation of `ping` can be used instead.

Network Address Translation (NAT) partially solves address shortage by assigning each company a single IP. This also allows for home connection to be shared among multiple devices. Within a company each device gets a unique local IP used for routing internal traffic, when a packet exits the local network, address translation takes place. The following ranges are declared as private to be used in local networks;

$$
\begin{array}{ll}
\texttt{10.0.0.0-10.255.255.255/8} & \text{16,777,216 addresses} \\
\texttt{172.16.0.0-172.31.255.255/12} & \text{1,048,576 addresses} \\
\texttt{192.168.0.0-192.168.255.255/16} & \text{65,536 addresses}
\end{array}
$$

NAT hides the local addresses in the intranet from the public addresses in the internet. Other special addresses include the following;

- `0.0.0.0/0`   default route, used for packet forwarding when no other applicable IP matches
- `0.0.0.0/8`   this host on the network (not sent)
- `127.0.0.0/8`   loopback (this host, again) but can be sent (`localhost = 127.0.0.1`)
- `169.254.0.0/16`   link local (something went wrong trying to acquire an IP address)

The use of NAT allows one IP to be rented, and then many internal IPs can then be allocated.

For example, if a connection is set up from `10.0.0.1` (local address), on port `X`, the router uses its public source address `138.76.29.7` on port `Y`, and also registers a mapping $X \leftrightarrow Y$. A reply sent to the router on port `Y` is then sent to `10.0.0.1` on port `X`.

Similarly, to allow for incoming connections, port forwarding can be done. This is a static mapping on the router, such as telling it to redirect all traffic on port 80 to `10.0.0.2` on port 8080.

While NAT has widespread use, it has a number of problems, and criticism;

- violates architectural IP modal, which states an IP address should **uniquely** identify a single machine worldwide
- changes Internet from connectionless to connection-oriented network (NAT gateway maintains mapping for each connection)
- violates rule of protocol layering by making assumptions about higher layers
- cannot easily support new transport protocols
- P2P protocols may require full connectivity - users behind NAT cannot be contacted directly by peers

IPv6 deals with all of this, by having many more addresses.

Dynamic Host Configuration Protocol

A host needs to determine its own IP address, and this can be done via DHCP (RFC 2131 and 2132);

1. each newly-booted machine broadcasts a `DHCPDISCOVER` packet when it connects
2. DHCP server (may be the router) replies with the assigned IP address

     may maintain static mappings, or assign different addresses each time the host connects

To prevent a host from keeping an address for ever, periodic refreshing is required (leasing). This is used for a router to receive an IP from the ISP.

Subnetting Calculations

Note that the first address in the subnet (the network address) cannot be assigned to a host. Additionally, the last IP address of the subnet is the broadcast address, and also cannot be used for a host. The first host address is typically assigned to the router (gateway), but some routers may take the last host address instead.

For example, with a IP address of `192.168.1.100`, and a subnet mask of `255.255.255.0` (CIDR `/24`), we have the following addresses;

- `192.168.1.0`   network address
- `192.168.1.1`   first host address (typically the router / gateway)
- `192.168.1.254`   last host address (hence 254 host address, out of a total 256 addresses)
- `192.168.1.255`   broadcast address

Longest Prefix Matching

We have the following routing table;

rule	network	port
1	`123.4.0.0/16`	1
2	`98.7.1.0/16`	2
3	`123.4.20.0/24`	2
4	(100...) `128.0.0.0/1`	3
5	`66.249.0.0/16`	3
6	`0.0.0.0/0`	4
7	`128.138.0.0/16`	4

Based on the routing table, the following addresses have these output ports;

address	port	comment
`123.4.1.69`	1	rule 1 (cannot match the 20 on rule 3)
`98.7.2.71`	2	rule 2
`200.100.2.1`	3	rule 4 (first bit is a 1)
`123.4.20.11`	2	rule 3 (can match rule 1, but rule 3 is longer)
`123.4.21.10`	1	rule 1 (cannot match the 20 on rule 3)
`128.138.207.167`	4	rule 7
`68.142.226.44`	4	rule 6 (default route - does not match anything else)

Hierarchical Routing

A flat network model is too simplistic. It doesn't allow for scaling, as transmitting routing information (as well as forwarding) is too expensive. Additionally, it has no administrative autonomy, as one organisation may want to run DVR protocol, whereas another may want to run LSR protocol (organisations want different routing protocols), and they also may not want to expose their internal network structures.

The Internet is organised in Autonomous Systems, which are independent administrative domains, with gateway routers interconnecting autonomous systems. There needs to be a distinction between routing within an autonomous system and between autonomous systems;

- **intra-AS routing protocol**   runs within autonomous system

  This determines internal routes between any combination of internal routers and gateway routers (in the same AS). This should provide optimal routes, via the use of one of the following;

- **RIP** (distance vector)   used in old ISPs and small organisations
- **OSPF - Open Shortest Path First** (link state)   used by larger organisations

  This is an abstraction of the network, by collecting the entire network into a directed graph - with the edges representing costs. This then computes the shortest path from every router to every other router.

  This allows for ASes on the Internet to be divided into areas. The routers connecting an area to the backbone are referred to as area border routers - responsible for routing packets outside an area. The backbone area routes traffic between other areas in AS.

- **inter-AS routing protocol**   determines routing at autonomous-system level

This has to deal with politics, such as some ASes should not be traversed, or are more expensive. **BGP** (border gateway protocol) is the Internet standard. This is used when traffic is leaving an AS, though the backbone, into another AS. This doesn't care about the best path - just any path. This routing is based on reachability information, as well as policies. It is a path-vector protocol, similar to how DVR announces distances, but instead paths are announced.

Routers advertise routes to networks, with destinations denoted by address prefixes. It may or may not forward an advertisement for a foreign network (as doing so means it is willing to carry traffic for that network). Additionally, prefixes may be aggregated (supernetting) - combining as follows;

$$\left.\begin{array}{l} \texttt{128.138.242.0/24} \\ \texttt{128.138.243.0/24} \end{array}\right\} \rightarrow \texttt{128.138.242.0/23}$$

$$\left.\begin{array}{l} \texttt{192.224.128.0/22} \\ \texttt{192.224.136.0/21} \\ \texttt{192.224.132.0/22} \end{array}\right\} \rightarrow \texttt{192.224.128.0/20}$$

The message structure consists of the following;

- ASN (Autonomous System Number)   unique identifier for each AS
- BGP attributes
  * `AS-PATH`   sequence of ASNs through which this advertisement was sent
  * `NEXT-HOP`
    specifies next interface (IP address) for forwarding packets towards advertised destination (this resolves the cases where the AS is reachable through multiple interfaces)
- BGP import policy   decides whether to accept or reject route advertisement
    router may not want to send traffic through a specific AS listed in `AS-PATH`

The count-to-infinity problem in DVR is solved with path inspection, where failed routes are spotted and the chosen route can be changed. For example, F receives the following information about the path they take to D from its neighbours;

- $B : BCD$
- $G : GCD$
- $I : IFGCD$
- $E : EFGCD$

$$
\begin{array}{cccc}
A & B & C & D \\
| & | & | & | \\
E & F & G & H \\
 & | & | & \\
 & I & J &
\end{array}
$$
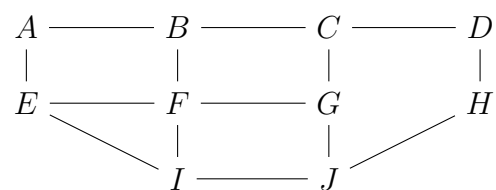
If G crashes, F will see that the routes advertised by I and E fail because they pass through G. Therefore F will choose $FBCD$ as its new route.

The routes are ranked according to;

1. **preference** value (depends on policy, and can be ∞)
2. **shortest** AS-PATH
3. **closest** NEXT-HOP router

Routers within an AS exchange their intra-AS routing information as expected. Gateway routers also discover inter-AS routing information, which is then propagated within AS. Both inter-AS and intra-AS routing information is used to populate forwarding tables. Destinations within the same AS are reached as usual. Destinations outside the AS are reached as follows;

- inter-AS information is used to decide is the destination x is reachable through the gateway G_x

- intra-AS information is used to decide how to reach G_x

- if x is reachable through multiple gateway routers $G_x, G_x', \ldots$, the following strategies can be taken;

  - use intra-AS information to determine the costs of paths to $G_x, G_x', \ldots$
  - **hot-potato** routing sends it through the closest gateway

IPv6

The current version of IP cannot support enough addresses, and therefore isn't flexible enough to support many users. IPv6 addresses this by supporting billions of hosts, improved security, reducing the size of routing tables, scoped multicasting, and many more improvements. It has some backwards compatibility, but adoption is an issue. The IPv6 header consists of the following;

- version   same as before
- traffic class
- flow label   sets up pseudo-connection between source and destination
- payload length
- next header
- hop limit
- source and destination addresses (128-bit each, also supports anycast address where a single address represents multiple interfaces)

Fragmentation is pushed onto the end-systems, and the header checksum is no longer used (since TCP and UDP still have checksums).

25th February 2020   Week 8, Lecture 1

Introduction to the Data Link Layer

This layer addresses data is transmitted successfully via the use of frames and their synchronisation (**framing**), as well as how a shared communication channel can be accessed by combining signals without data loss (**multiplexing**).

Ethernet

Ethernet is a protocol meant for LAN/MAN/WAN (mainly LAN). Originally, it started with a coaxial cable, achieving speeds of approximately 2.94Mbps, and currently it can achieve speeds of 100Gbps, through fiber optics. Common types of Ethernet cables include, with different levels of protection against electromagnetic interference (EMI);

- **UTP** (unshielded twisted pair)

  consists of pairs of twisted wires, Cat5e is the most common (at the moment), but versions Cat6a (wires are separated with plastic, preventing accidental damage), and Cat7a (not yet in production) also exist, and Cat8 is in development

- **STP** (shielded / screened twisted pair)

  similar to UTP, but there is a foil around all the cables to prevent shield from interference

- **FTP** (foiled twisted pair)   each individual wire is wrapped in foil

- **SFTP** (shielded and foiled twisted pair)   combination of the previous two

Ethernet cables come in three variations;

- **straight-through**   both sides have same order to cables

  Used to communicate between devices of different OSI layers. An example of this is a connection between a switch (layer 2) to a router (layer 3).
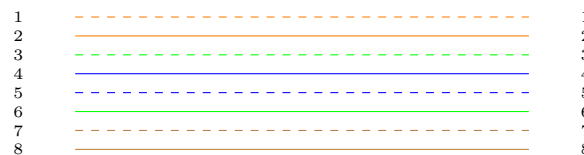
  Also known as MDI (media dependent interface).

- **crossover**   some are flipped

  Used to communicate between devices of the same OSI layer, such as from a switch to another switch (both layer 2).
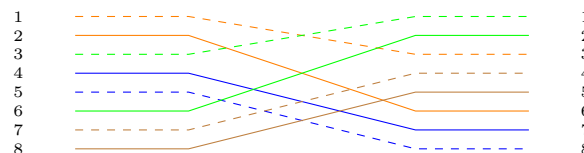
  Also known as MDIX (media dependent interface with crossover).

- **rollover / console**

  Used to directly connect to a networking device (such as to troubleshoot a router).
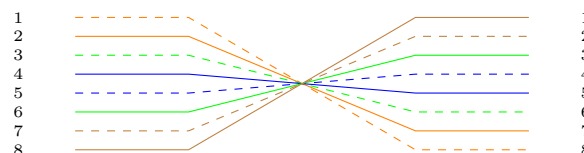
Note that to connect a computer (layer 3) to a router (layer 3), a straight-through cable is used, as we are actually connecting to the switch (layer 2) inside the router. Additionally, some modern devices can "fake-swap" between a straight-through and crossover, by using internal, software-based remapping (Auto-MDI/MDIX).

Frame

A frame consists of data from the layer above, with a header and footer (containing a CRC (cyclic redundancy check) checksum). The Ethernet II header consists of the following;

- preamble   64-bit
- destination and source MAC addresses   48-bit each
- Ethernet type   16-bit

This is followed by 46 to 1500 bytes of data. The checksum is the footer.

MAC Addressing

Each NIC conforming to IEEE 802 has a MAC address. A manufacturer of network devices is given a range of addresses - once they reach the end of this range, they return back to the start of the range. Two devices can have the same MAC address, as long as they are not on the same LAN, since after this point, a device is represented with an IP address. The address 48 bits, formatted as six colon-separated bytes in hexadecimal.

The first half of the address (24 bits) is the manufacturer specific OUID (organisationally unique identifier), and the remaining 24 bits is specific to the NIC. Bit 6 (starting with the leftmost bit being 0), is the U/L bit, which decides if the address is universally or locally administered, and bit 6 (I/G bit) decides if it is an individual or group address. An I/G bit of 0 indicates unicast, whereas 1 indicates multicast. A U/G bit of 0 indicates it is globally unique (OUI enforced), whereas 1 indicates it is locally administered.

The software MAC address can be changed (spoofed), which may confuse a basic layer 2 switch. A more advanced device may notice the issue, and notify an administrator.

Switch

A switch is smarter than a hub as it stores a table (FIB - forwarding information base) of where each MAC address resides, and thus forwards messages out of the right port (if it knows where). It remembers which port your MAC is connected to.

Sniffing a switched network can be difficult, as it knows which port frames should be sent out of. The only time someone else's frame would be seen is from a broadcast, or flood.

Two switching methods are as follows;

- **store-and-forward switching**

    whole frame must be received first, and then it can be forwarded (hence larger frames will take longer to forward) - this allows the switch to check the frame for errors

- **cut-through switching**

    it begins forwarding as soon as it has enough information (when the switch knows where to forward it) - the switch cannot check the frame for errors (as the CRC is at the end of the frame), thus the recipient needs to check

Wireless Access Point

Essentially acts as a wireless switch. It goes by the name 802.11, which uses 2.4GHz or 5GHz of the radio spectrum (which can be used without a license). This can also act as a bridge. Sniffing a wireless network is easier, as data is sent everywhere (hence encryption is very strongly recommended).

Issue with Hubs
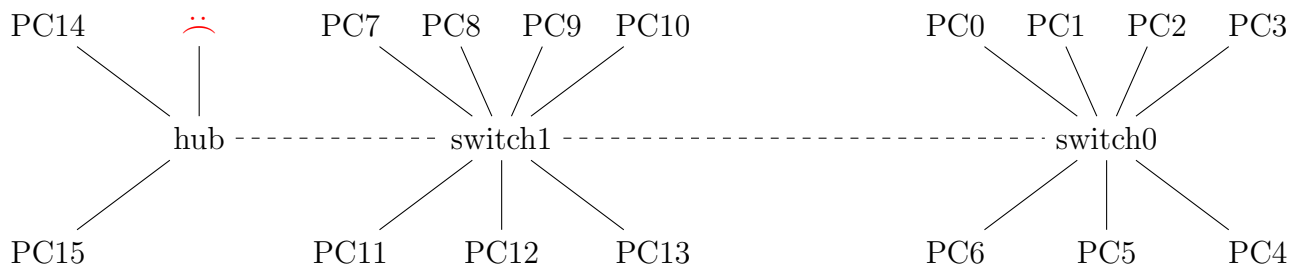
In the scenario above, the attacker, :(, can see all traffic sent to and from PC14, and PC15. They can also see all the broadcasts from switch 1, since it is sent to the hub. This is due to how hubs broadcast everything they receive - it also generates a large amount of redundant traffic.

27th February 2020   Week 8, Lecture 2

Switched Ethernet

A typical switched Ethernet topology has a single machine connected to each switch port. This avoids collisions on networks by separating collision domains.

Network Topologies

In the past, devices were connected directly to the cable (switches and Ethernet weren't available), and some computers had multiple NICs, allowing them to connect between hosts. Some topologies are as follows;

- **bus topology**

  This had a main coaxial cable (called the bus) connecting all hosts, where all the data travelled. A BNC coaxial T connector would be needed to add a new host. At the end of each side of the cable, a BNC terminator was added to signify the end of the network.

- **ring topology**

  Each host needed two NICs to achieve this, as all host were connected in a physical ring, which data flowed around. If a link was cut, the network died. To circumvent this, dual-ring was attempted, which means each device needed four NICs, which became too expensive for large networks.

- **token ring topology**

  This was similar to the ring topology, but instead passed around a token (forming a logical ring). An intermediate device, the MSAU (multistation access unit), was needed to achieve this (a predecessor to the switch). When data returned back to the original sender, it was removed from the network, and if a host died, it stopped passing the token to the dead host. Each host operates in the following modes;

  - **listen** mode

    Host H_n uses a single bit buffer to copy input bit stream from H_{n-1} to H_{n+1}, even if the frame is addressed to it. If it is the independent recipient of the frame, it keeps a copy of it.

  - **transmit** mode

    Host H_n reads bit stream from H_{n-1} into memory, and transmits frame from its memory to H_{n+1}. The whole frame does not need to fit on the ring, as the holder of the token can keep transmitting (only one device can speak, as it must hold the token).

Only one host will be in transmit mode (the sender), and once it drains off the frame F it sent (it has gone around the logical ring), it places the token T on the ring. Early release mode means that the token is placed on the ring before F arrives back (trusting that it has been sent around the ring). Until the token is placed back on the ring, all other hosts must be in listen mode - thus no collisions can occur.

After the token is sent, the host switches back to listen mode. If the next host doesn't need to transmit, it can just pass on the token, and once a host wants to transmit it can drain off the token T and send off its data frame F.

A token ring frame consists of the following fields (the token consists of the fields in violet);

  – start delimiter   1 byte
  – access control   1 byte

    This has a single bit to denote the presence of token, and priority bits to denote the priority level of the frame.

  – frame control   1 byte

    Used to generate different control frames. A special **active monitor** station must take responsibility for generating token and draining orphaned frames - any host must be able to take this function as a monitor can fail at any given time.

  – destination address   6 bytes
  – source address   6 bytes
  – data ≥ 0 bytes
  – FCS (checksum)   4 bytes
  – end delimiter   1 byte
  – frame status   1 byte

    Defines whether the address was found and the frame was copied (is set to 1 after data is copied).

This is followed by an interframe gap (IFG) between frames to separate them.

A host with a lower priority frame cannot take the token until all hosts with higher priority frames have taken it. It also supports a reservation scheme where a host in listen mode can increase reservation priority. The priority scheme can cause low priority data to be delayed indefinitely, but guarantees high priority data will be sent quickly.

- **star topology**

Most networks started using switches, which hosts directly connect to (WiFi networks are based on the star topology). This allows for any host to communicate with any other host without waiting for the token (as long as there is an agreed way to avoid interference). However, the central device is a single point of failure.

- **line topology**   similar to a ring, but does not meet at end
- **tree topology**   hybrid of star and bus
- **mesh topology**   some hosts are connected to some hosts (many NICs needed)
- **fully connected**   every host is connected to every other host (very expensive)

Leaving LAN

The MAC address allows a device to be unique within its network, but the Internet requires an IP address on top of the MAC address. However, switches are layer 2 devices, thus they must work with routers for an IP to be assigned (either statically or dynamically). This is linked to your MAC address, which is how the router informs the switch which host each packet is for. With the help of layer 4 port numbers, it can differentiate between devices.

IP addresses (network layer)

- specify addresses of hosts involved in end-to-end communication

- do not have to change as packets pass through a router

- can use IP address of any host on the Internet

MAC addresses (data link layer)

- specify addresses of host involved in communication on the same (sub)network

- usually change as frames pass through routers (as packets) - it takes the MAC address of the interface on the gateway when it leaves the LAN (this MAC address represents the entire LAN)

- can only use MAC addresses of hosts within the same subnet

Address Resolution Protocol (ARP)

Addressing hosts through IP addresses is not understood by NICs. A host on an Ethernet LAN only reads messages in frames that contain the host's hardware MAC address. ARP is used to find out the MAC address of a host given its IP address;

1. router asks each host on LAN if they have the requested IP address (encapsulated as ARP message in data link frame, and broadcasts)

2. each host then checks if it has the requested address - if it does, it sends a reply with its MAC address

3. router receives the ARP message, and usually caches the IP address along with the MAC address - the IP datagram is then forwarded to the correct host, encapsulated in data link frames

27th February 2020   Week 8, Lecture 3

Medium Access Control Sublayer

The data link layer can be divided further;

- LLC (Logical Link Control) sub-layer
- MAC (Media Access Control) sub-layer

In topologies like token ring, hosts are not expected to transmit at the same time, however with Ethernet and star networks, this is not the case - we may be on a broadcast channel. Examples of broadcast channels are shared connections, such as a shared wire (Ethernet), shared wireless (WiFi), or satellite links.

When we have overlapping frames at the recipient, we assume they are lost and must be transmitted. This is always true in wired LANs, however this may not be true in a wireless network, as one transmitter may be stronger than another (and prevails). Some strategies are as follows;

- **no control**

    A simple approach is to let stations retransmit after a collision. This is fine if the channel utilisation is low, but will be very inefficient if many devices are trying to get frames into the network simultaneously.

- **round-robin**

  Similar to the token ring topology where stations take turns using the channel. Only the station with the token can transmit - this slows down the network by a significant amount.

- **reservations**

  A station will need to reserve the channel before transmitting (slotted system). Consider N stations sharing a channel;

  - **TDM** (time division multiplexing)

    A station must wait for its turn / slot to transmit, and the transmission rate is limited to $\frac{R}{N}$, where R is the maximum channel rate.

  - **FDM** (frequency division multiplexing)

    A station must only use a limited frequency band, but allows multiple stations (on different bands) to talk simultaneously. The bandwidth is not fully utilised; $\frac{B}{N}$, where B is the total channel bandwidth. If the bands are too close together, interference is likely.

  As both these methods have drawbacks, we want to use dynamic channel allocation.

Dynamic Channel Allocation

Some schemes for dynamic MAC are as follows;

- **basic ALOHA**

  The ALOHA protocol was developed at the University of Hawaii in 1971. The (simple) idea is as follows;

  1. when a station has a frame to transmit, just send it
  2. when a collision occurs, stations wait a **random** amount of time before retrying

  This grants fair channel access, as any station has the same probability of transmission.
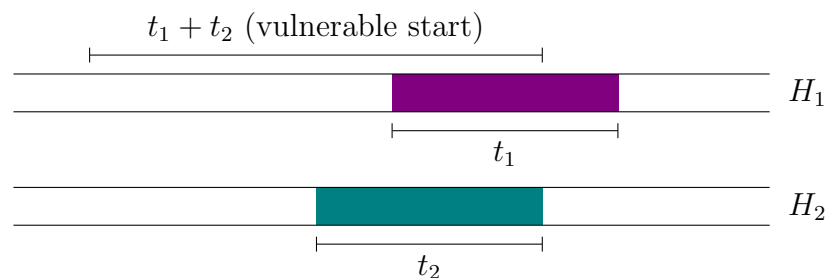
  However, this suffers from low channel efficiency, if there is much channel contention. This is due to the large vulnerability period for H_1 to start transmitting.

- **slotted ALOHA**

  In order to reduce the vulnerable period, slotted transmission is used (thus stations may only transmit at specific times). Stations are only permitted to send frame at the start of a fixed size slot - it assumes time synchronisation between stations.
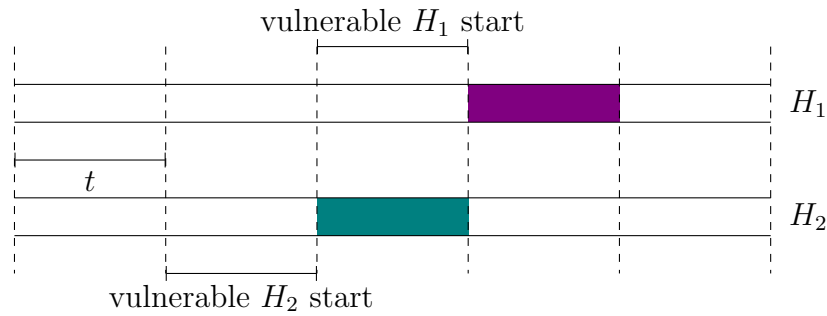
The fixed slots for transmission reduce the vulnerable period, as collision only happens with an exact overlap.

- **carrier sensing**

  This listens for other transmission before sending, and only allowing transmission when channel is idle;

  - CSMA/CD (Carrier Sense Multiple Access / Collision Detection) Ethernet (IEEE 802.3)

    The station senses the channel while transmitting to know that its frame is OK. It stops transmission as soon as collision occurs (collision detection), and adds a jamming signal to create an agreement about the collision. The host must transmit for long enough to know that the frame is OK, thus the minimum frame length is 2η, where η is the end-to-end transmission delay.

    Note that if we back-off for the same amount of time, we will collide again (this is addressed in the **Back-off** section below).

    CSMA/CD has no notion of authority to transmit, and collisions are inevitable hence this is a best-effort service. In the worst case, a particular station may be delayed indefinitely; this is acceptable for many applications such as office LANs, however this is not acceptable for real-time systems.

    This is continued in a later section, **Carrier Extension and Frame Bursting**.
  - CSMA/CA (Carrier Sense Multiple Access / Collision Avoidance) WiFi (IEEE 802.11)

  However, collisions may still occur due to transmission delay.

- **token passing** similar to token ring

Back-off

Some strategies for addressing back-off are as follows;

- **1-persistent**

  The station keeps checking if the channel is free, and transmits immediately when it is (Ethernet does this). This is fine in a switched architecture.

- **non-persistent**

  If the channel is busy, wait a **random** amount of time before checking again, and then transmit when free.

- **p-persistent**

  The station keeps checking if the channel is free, and then transmits with a probability of p. Even if the channel is free, we may end up not transmitting.

- **binary exponential**

  When the channel is idle, the station attempts to transmit. If a collision occurs, it waits either 0 or 1 slots before attempting to transmit again. If another collision occurs, it waits 0, 1, 2, or 3 slots before attempting again. After c collisions, it chooses a slot in the range $[0, 2^c - 1]$ for the next attempt. After 10 collisions, we give up (upper limit of 1023 slots). The minimum frame length is treated as the slot length.

Carrier Extension and Frame Bursting

Any Ethernet frame less than 512 bytes is extended by the host sending carrier, which keeps the line busy, as if it was originally 512 bytes long. This allows the network length to be 200m end-to-end, but is very inefficient.

To use this idle time, we can send multiple frames, via frame bursting. After an Ethernet frame is sent, if it isn't long enough, it pauses for a bit and sends another, once again, if it still isn't long enough it sends another, and so on. Note that we can extend beyond 512 bytes, and if no more frames are available to send, then we still have the padding.

This allows us to increase the network speed with minor algorithmic changes (software).

Switched Topology

In a fully switched topology, it is possible to ensure collisions never occur with switches, by buffering frames and transmitting when a channel is available. Each channel has at most two stations.

3rd March 2020   Week 9, Lecture 1

Wires

Examples of communication media include wired (such as UTP), and wireless (such as radio). A low cost medium is produced by twisting a pair of wires together (unshielded twisted pair). Twisting wires reduces interference and cross-talk.

Patch Panels

- **socket panel**

    where cables end up (rarely changed)

- **network switch (layer 2)**

    used to create LANs (sometimes changed)

- **private branch exchange (PBX)**

    used to interlink phone systems
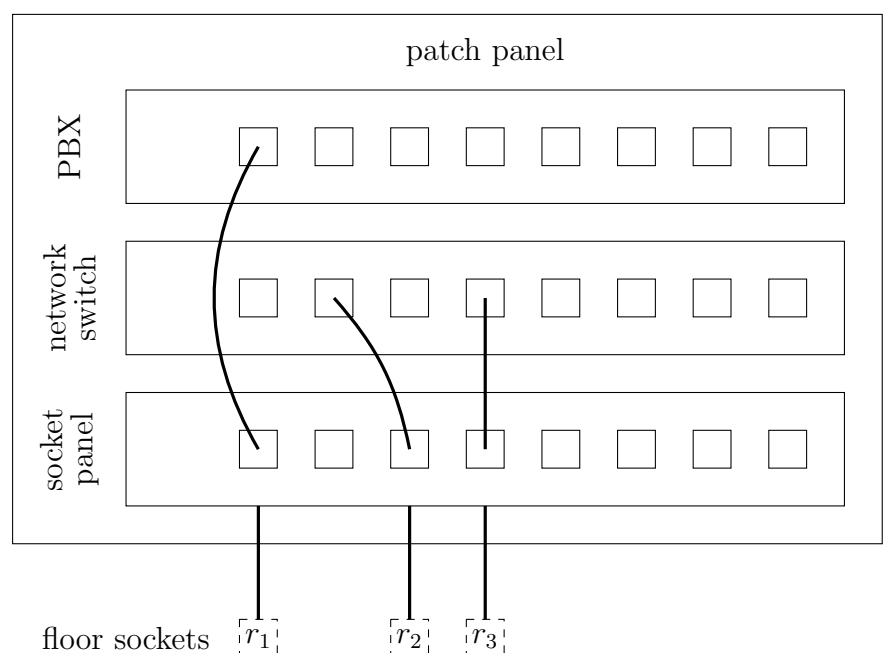
Coaxial Cables and Optical Fiber

The conductors are placed concentrically in a coaxial cable, thus avoiding the problems associated with twisted pairs. This supports a wider range of frequencies, and therefore has a higher bandwidth - but it has a higher cost compared to UTP.

Optical fibers do not suffer from, nor generate, electromagnetic radiation. It transmits data by exploiting the refraction of light, as light rays are refracted at the boundary when passing from one medium to another. Signal loss (attenuation) is low, and it has high bandwidth.

	frequency range	typical attenuation	typical delay	repeater spacing
twisted pair	0 - 1 MHz	0.7 dB/km at 1 kHz	$5\mu s$/km	2 km
coaxial cable	0 - 500 MHz	7 dB/km at 10 MHz	$4\mu s$/km	1 - 9 km
optical fiber	186 - 370 THz	0.2 - 0.5 dB/km	$5\mu s$/km	40 km

Wireless Transmission

The signal is carried in electromagnetic spectrum, and communication is bidirectional by default. This is mostly a broadcast medium. Radio is an example of a wireless medium, and different frequencies have different propagation properties.

Signal Representation

Signals can be represented in one of following ways;

- **analogue** (continuous signal)

    Changes in the information are represented in a natural manner analogous to a physical property of the channel.
- **digital** (discrete signal)

    Information is represented in a discrete set of states, using some coding scheme. A binary digital channel is a digital channel using only two symbols or states (0 and 1).

We can quantify the symbol rate per second with Baud rate (Bd) - similar to the bit rate, which is for binary digital signals, a symbol can express more than 1 bit.

A digital channel can be implemented by an analogue channel via a modem (modulator-demodulator).
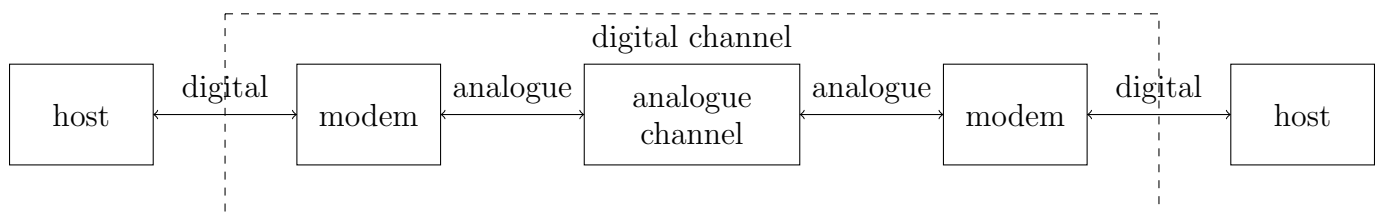
Similarly, in inverse, an analogue channel can be implemented by a digital channel with a codec (coder-decoder). A DAC is Digital to Analogue Converter, whereas an ADC is Analogue to Digital Converter.

Signals have the following properties;

- **waveform**   shape of the signal
- **amplitude**   range of values signal varies over

    maximum value or string of signal (voltage, brightness, etc.)
- **wavelength** (λ)   distance signal travels before repetition

    the length of single cycle of signal (typically measured as distance)

- **frequency**   number of repetitions per unit time

      number of cycles per second; typically measured in Hertz (Hz)

A formula we can use is the following (where $c = 3 \times 10^8 ms^{-1}$ - the speed of light);

$$c = f\lambda$$

Modulation

We need a modulation scheme to encode a (digital) information signal to be transmitted effectively using the range of frequencies (bandwidth) supported by a channel.

- **baseband modulation**   transmit the information signal unmodified (used for dedicated lines)
- **broadband modulation**

  Modifies physical carrier signal to encode information signal. The amplitude is changed, where a high amplitude indicates 1, and a low amplitude indicates 0. Different frequencies can also be used to represented different bits, and the phase of the waveform can also be changed.

  – **amplitude modulation** (AM)

  The standard amplitude represents a 0, and a higher amplitude represents a 1. This is also referred to as ASK (amplitude shift keying). The following example is for the byte `00011010`;
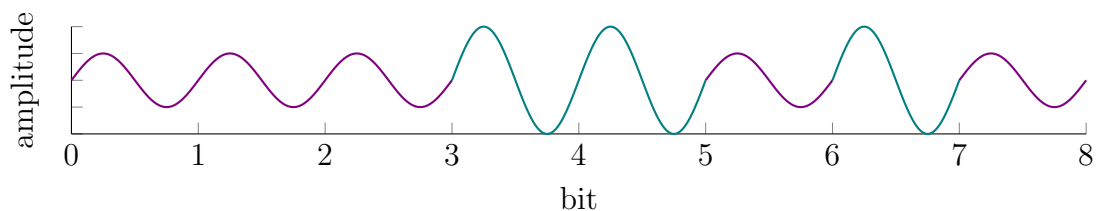

  – **frequency modulation** (FM)

  The standard frequency represents a 0, and a higher frequency represents a 1 (such as two cycles in a time unit). This is also referred to as FSK (frequency shift keying). The following example is for the byte `00011010`;
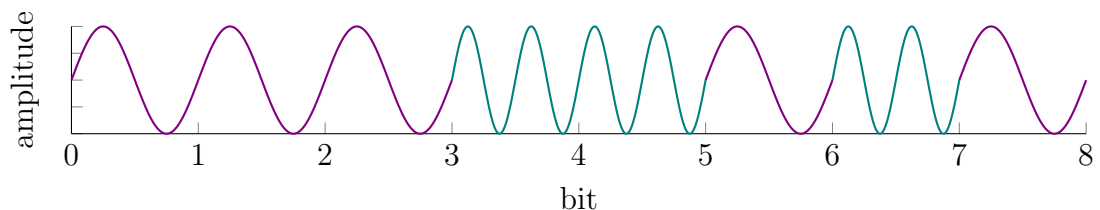
  – **phase modulation** (PM)

  A shift in phase indicates a bit (not an accurate example). This is also referred to as PSK (phase shift keying). The following example is for the byte `00011010` (not accurate, but just shows the idea);
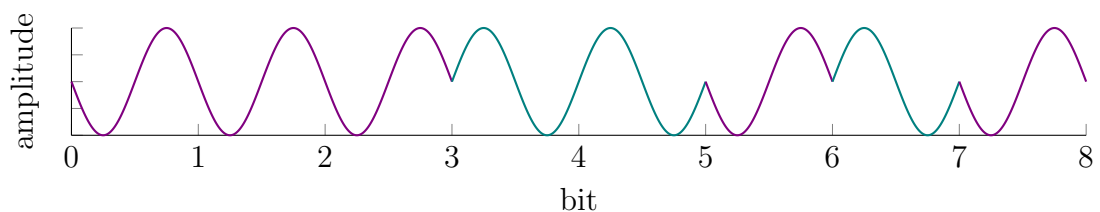
  We can transmit at a higher data rate by transmitting multiple bits per symbol in our modulation scheme - by using a combination of modulation schemes. Examples of this include Quadrature Phase Shift Keying (QPSK), N-Quadrature Amplitude Modulation (N-QAM), and many others.

Digital Subscriber Line (DSL)

On conventional phone lines, we managed to reach 56000 bps (56 Kbps) downstream, and 33000 bps (33 Kbps) upstream with the V.90 modem standard. Phone lines are artificially limited to 3000 Hz bandwidth, as normal voice goes from 400 Hz to 3400 Hz. This bandwidth filter is removed with DSL.

In ADSL (Asymmetric DSL), the connection is divided into 256 channels of 4000 Hz each. Channels 1 - 5, which are 4 - 25 kHz are not used to avoid interference, and more channels are allocated for download than upload, as downloading is more common for Internet users.