CO212 - Networks and Communications

14th January 2020 — Week 1, Lecture 1

Evolution of the Internet

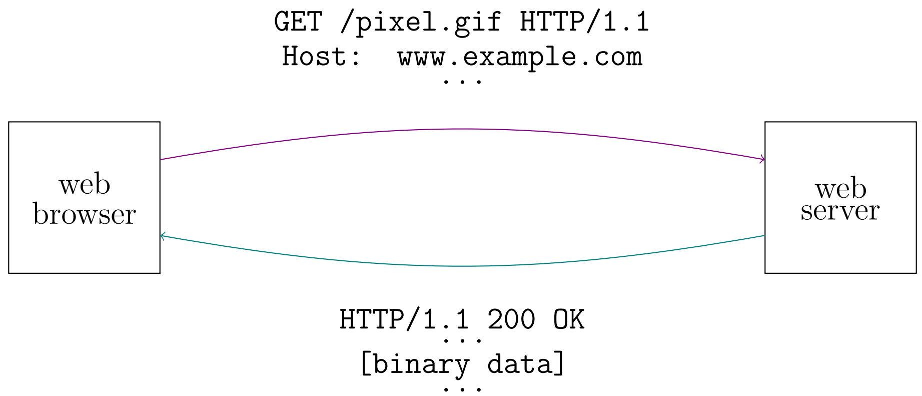
Literally only writing this part so I have something for the first lecture.

- (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.



Layers

- **application layer**

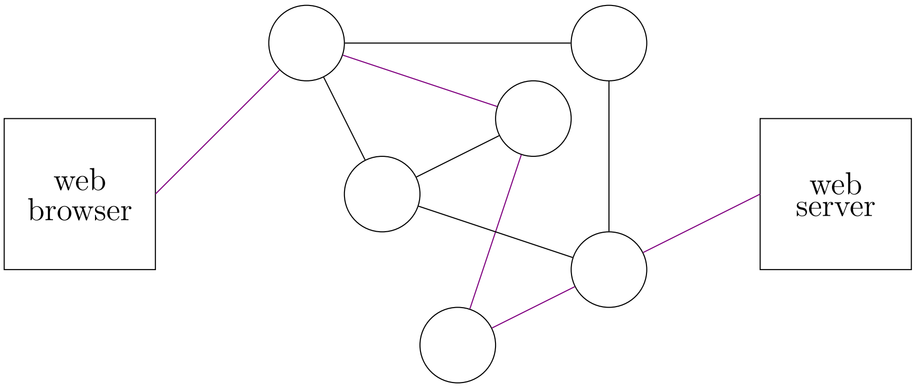
  Any software written for the internet is on the application layer.

- **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.

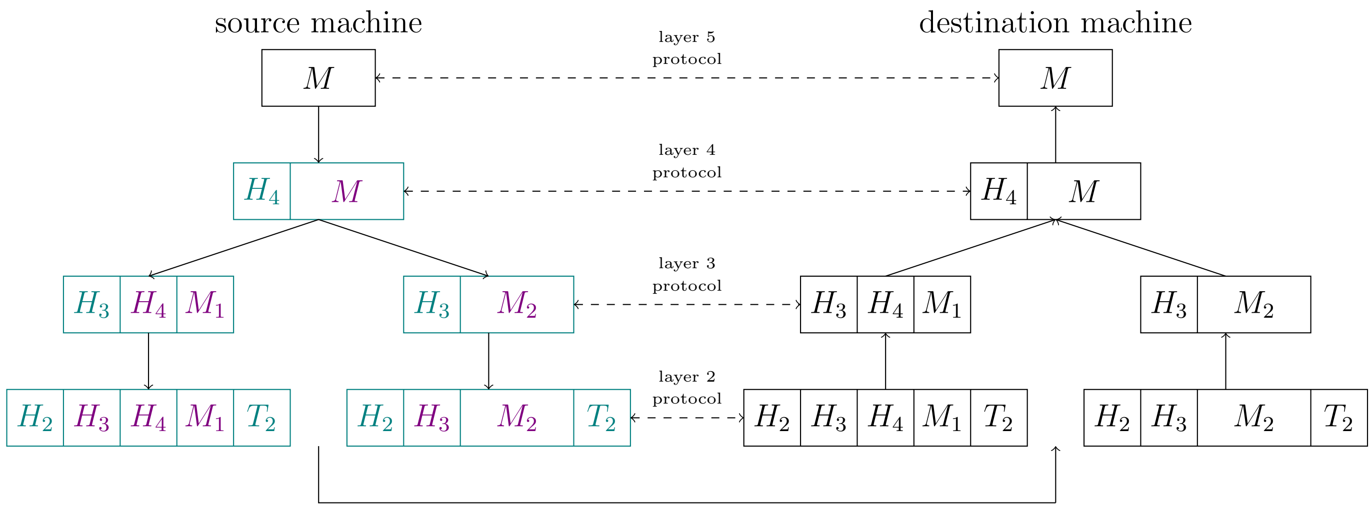
The TCP/IP (4 layer) stack consists of application, transport, internet, and network access (which combines data link and physical). On the other hand, the OSI (7 layer) model consists of the application layer, presentation, session, transport, network, data link, and physical.

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 $\frac{\text{distance}}{\text{wave propagation speed}}$)

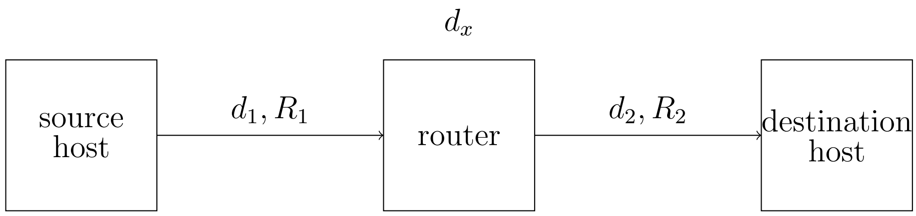
  note that this is half the RTT (round-trip time)

- 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.



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

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;

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

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

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

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