Cardiff's School of Computer Science & Informatics

CM1102 "Web Applications"

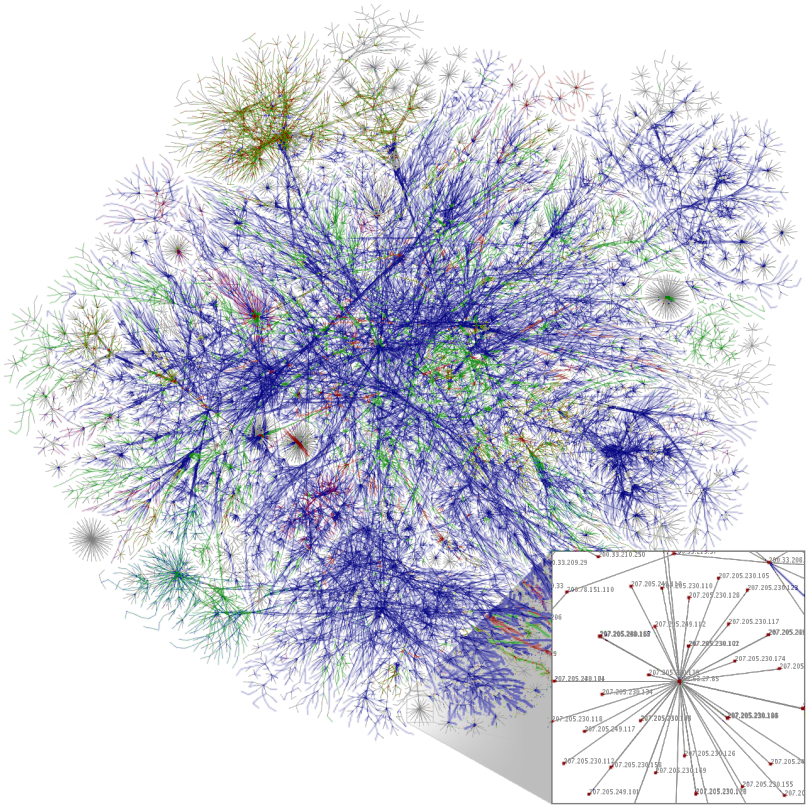
Networks

Dr Natasha Edwards

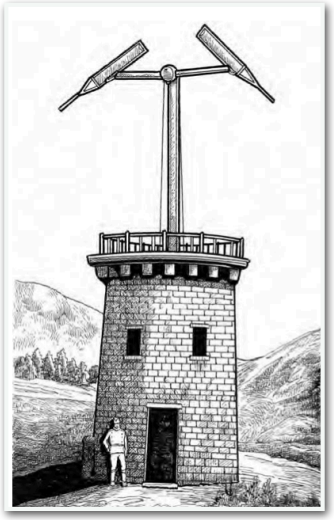
(Special thanks to Dr Kirill Sidorov)

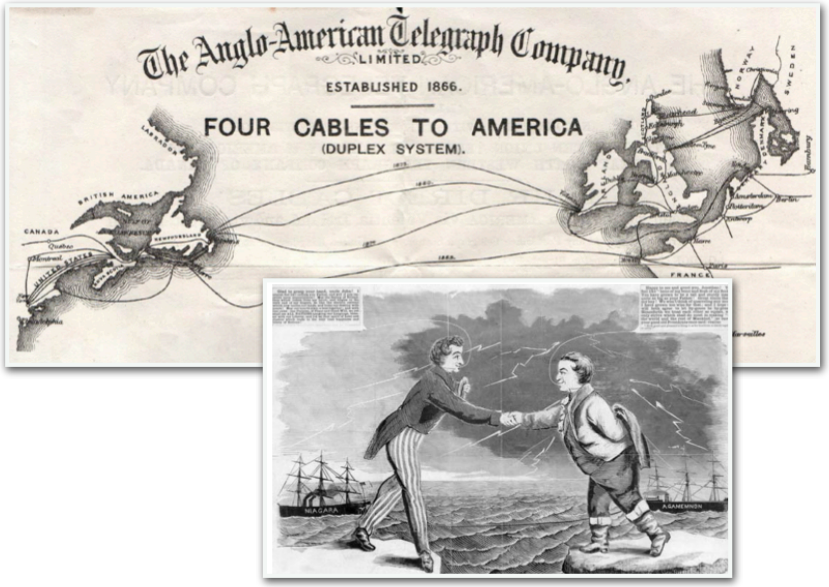
Evolution of Internet

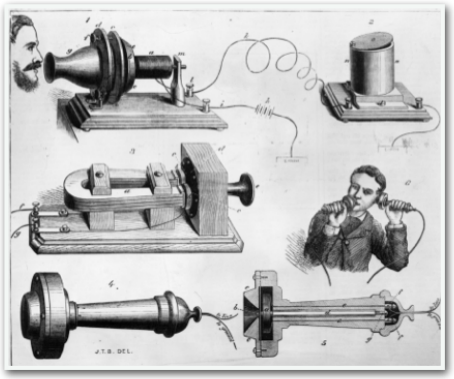
From smoke and drums signals (3000 BC) - to Modern Internet map

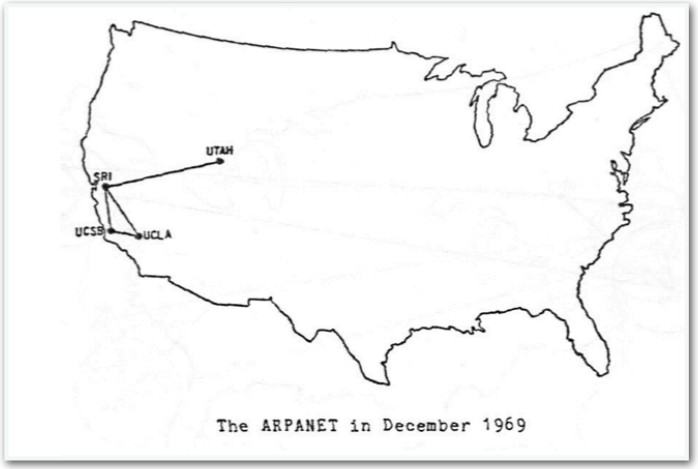


and things in-between...









Internet

What is Internet?

Broad definition: **Internet = a global, interconnected network of networks and computers (public, private, etc.)**

- Connecting millions of devices (*e.g.* PCs, laptops, phones, PDAs, TVs, sensors. . .)
- These **end systems**, or **hosts** (**client**s or **server**s, or both), send or receive web pages, telemetry, email, streaming videos. . .
- **End systems** are connected by **communication links**, made from various different physical media: copper wire, fibre optics, radio. . .

Internet vs WWW

- **World Wide Web (WWW or Web) = collection of linked documents/ files (web pages)**

Internet

Internet comprises

- physical infrastructure

  Anatomy: how is Internet put together?

- communication standards

  Function: how does Internet operate?

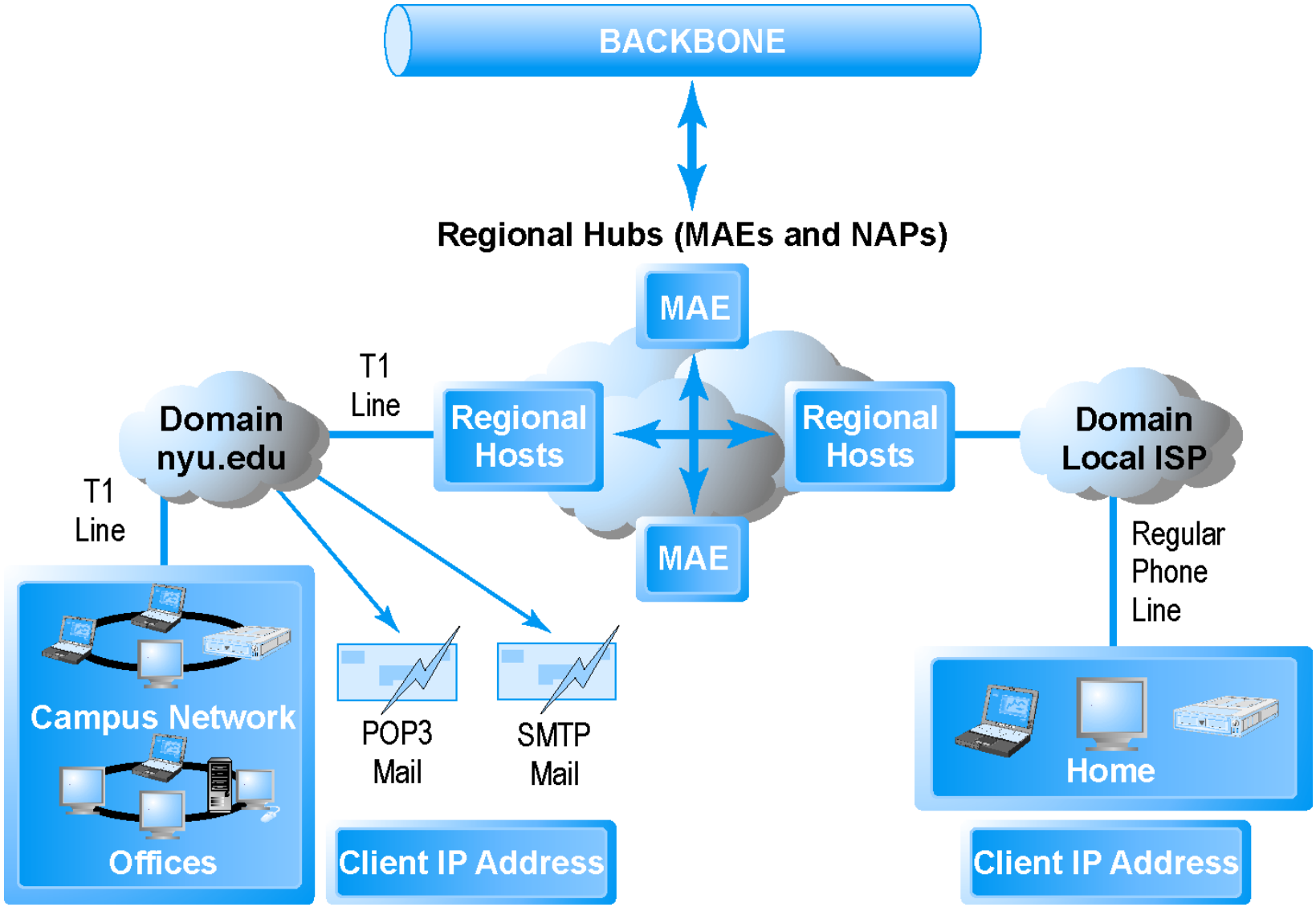
Types of Networks

- **by spacial scope:**
  - Interplanetary **(!)** Internet 
  - Internet
  - Cloud (IAN)
  - Wide (WAN)
  - Metropolitan (MAN)
  - Backbone
  - Local (LAN) (Home HAN, Storage SAN, Wireless WLAN)
  - Near-me (NAN)
  - Personal (PAN)
  - ... and others (Body BAN, Near-field NFC, Nanoscale)

Internet's physical infrastructure: EXAMPLES

- **Backbones**: provides principal routes between large interconnected networks and core Internet routers
- **Regional** networks: operate within specific geographical areas
- **Commercial** networks: provide access to backbones (e.g. ISPs)
- **Local** networks: network of a company/ organisation

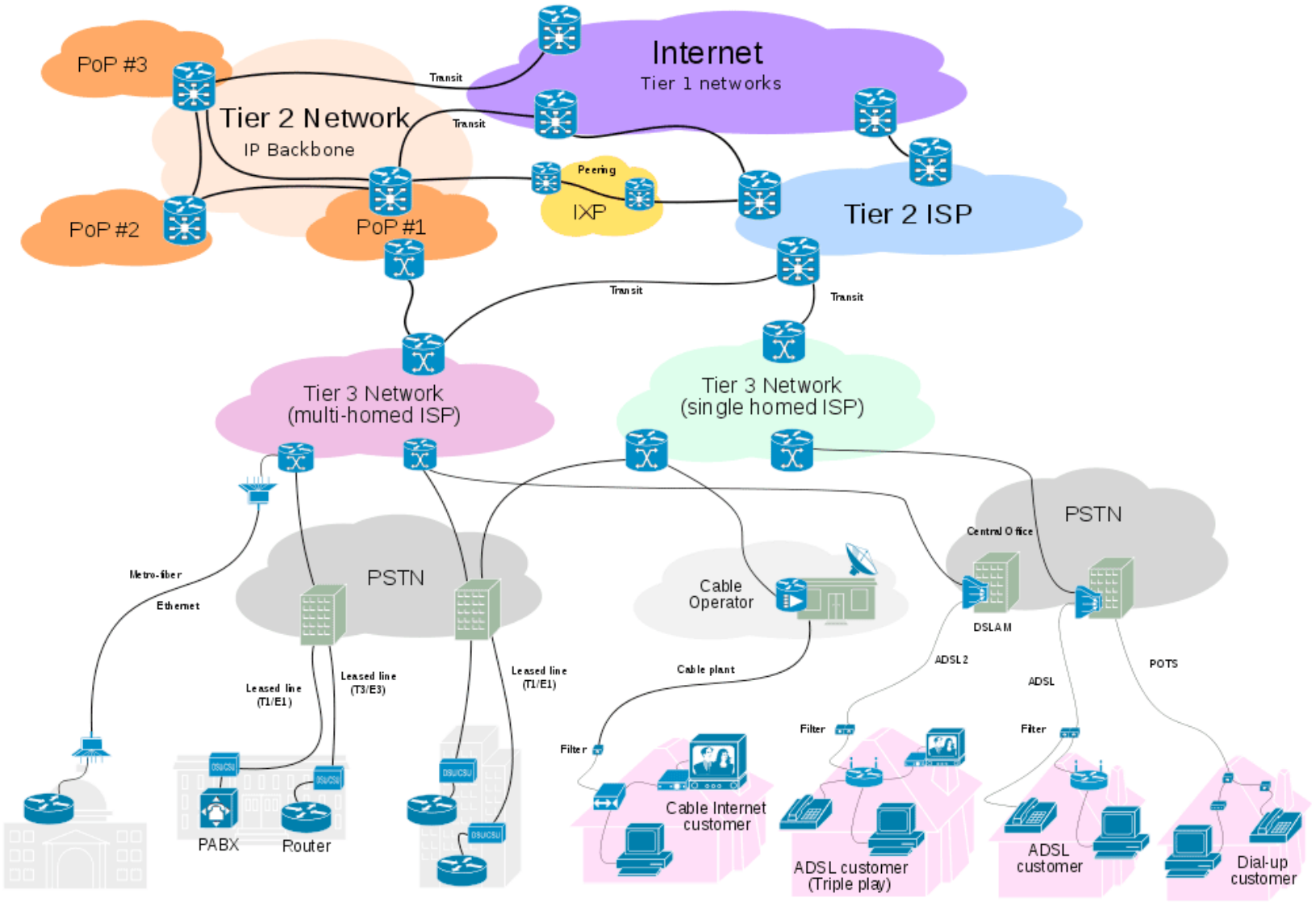
Internet's physical infrastructure: EXAMPLES



MAE = Metropolitan Area Exchange
NAP = Network Access Point
T1 line = international telephone standard for digital communication; guarantees delivery at 1.54 mbps

Source: Laudon, et. al (2010)

Multi-tier architecture



Internet's physical infrastructure: EXAMPLES

Source: [Atlas Of Cyberspaces]

Connectivity

How are end systems connected?

"The Internet is a network of networks"

— Thus in order to gain an understanding of the Internet we need to consider:

- What a **simple network** is.
- How computers communicate over a network.
- A variety of **protocols** (**rules of communication**) to govern communication.
- How networks are **connected together** to form larger networks.

What is a network?

A **network** is simply two or more computers (or other things) connected together.

Examples: mail, phone system, social connections, railroad system, roads.

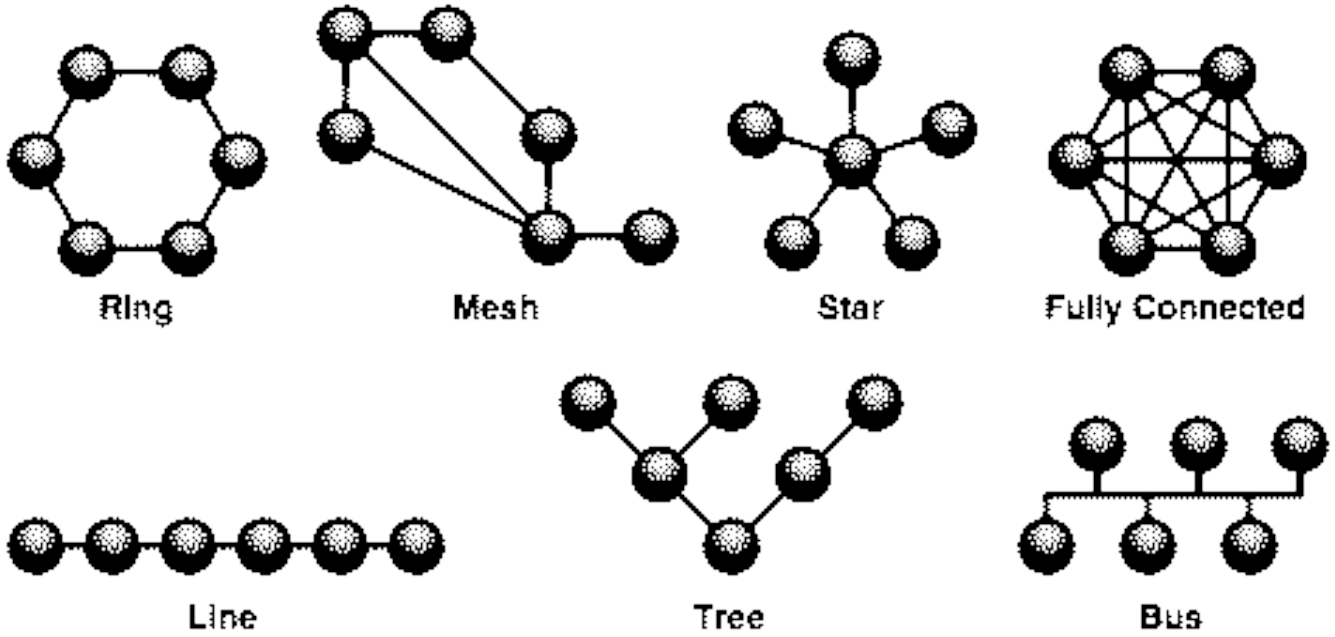
There are two basic ways in which we can classify computer networks:

- According to configuration of the network graph — **topology**.
- According to how nodes use communication links — **type of connection**:
  - **Circuit switched.**
  - **Packet switched.**

Network topologies

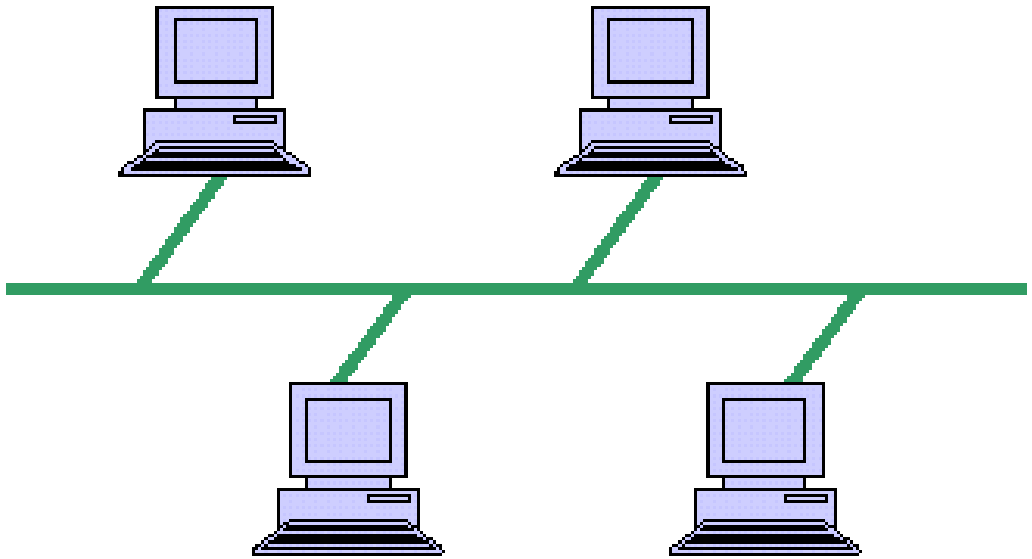
Connecting two computers together is a straightforward task — simple **Point-to-Point**. However, connecting several computers together is different.

Here are some ways:



Many of these topologies are used in the Internet.

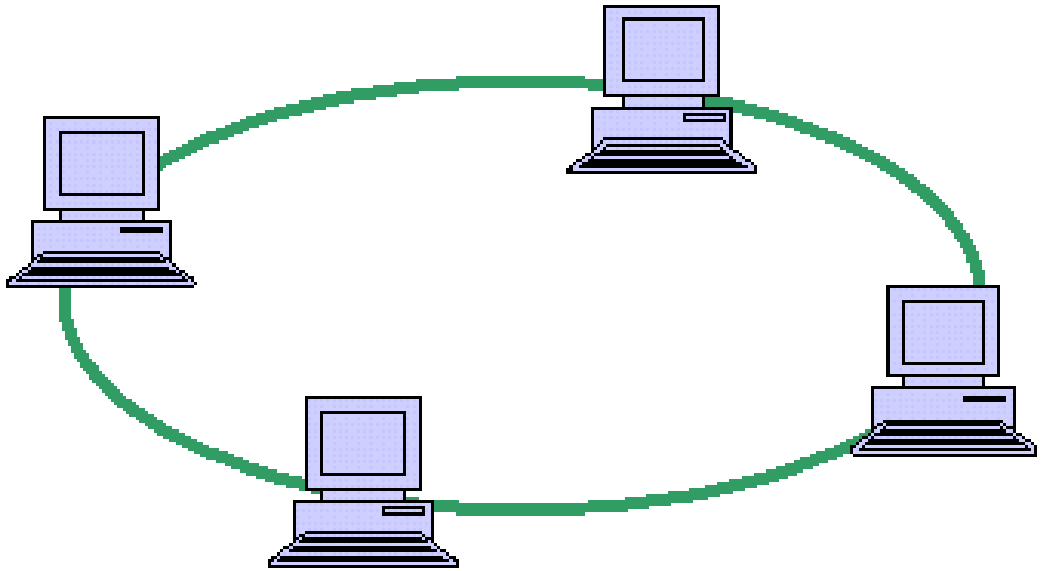
Bus topology



Advantages: cost-effective, requires little cable, easy to implement, simple hardware.

Disadvantages: Network disruption when nodes are added or removed, not fault tolerant, difficult to find faults.

Ring topology



Advantages: Cable faults are easier to find, moderately easy installation.

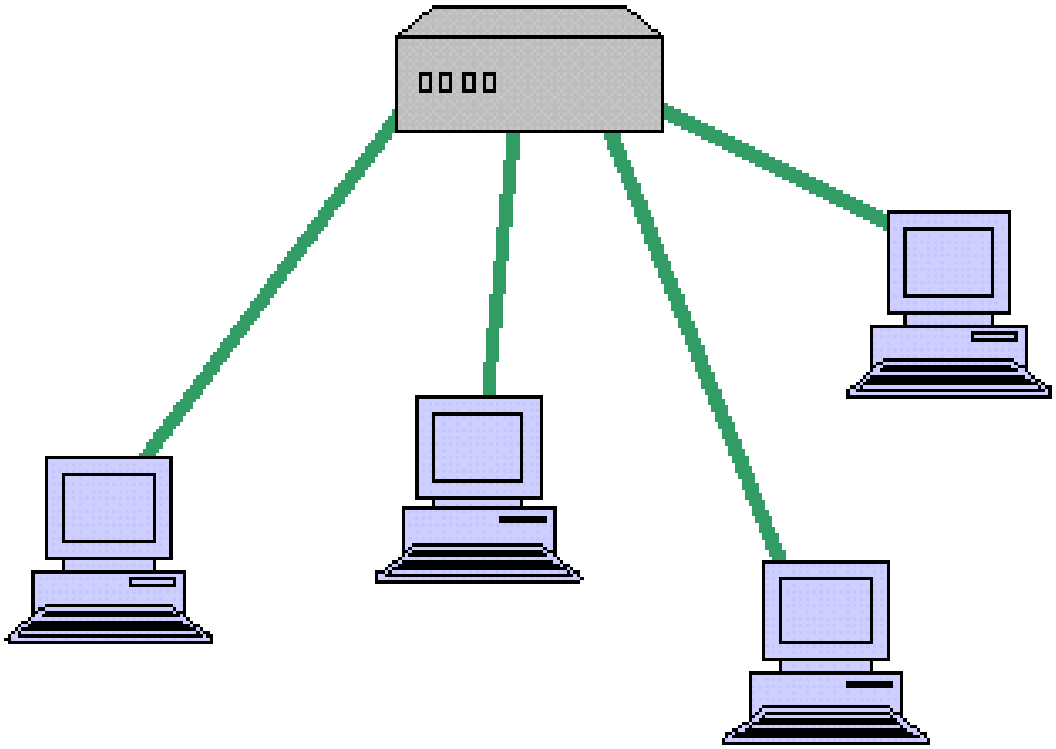
Disadvantages: Network disruption when nodes are added or removed, not fault tolerant.

Network topologies

Bus and ring configurations are called **broadcast** topologies:

- A message is placed on the bus or ring containing the name of the intended receiving computer.
- All computers listen constantly.
- If name identified by listener, message is captured.
- Only one node can broadcast at a time: needs a protocol.
    - Imagine two people talking via two tin cans connected via piece of string. If the two people talk at the same time then we get what is known (in networking terms) as **data collision**. Gets worse if more people are talking.
- Bus and ring topologies are common.

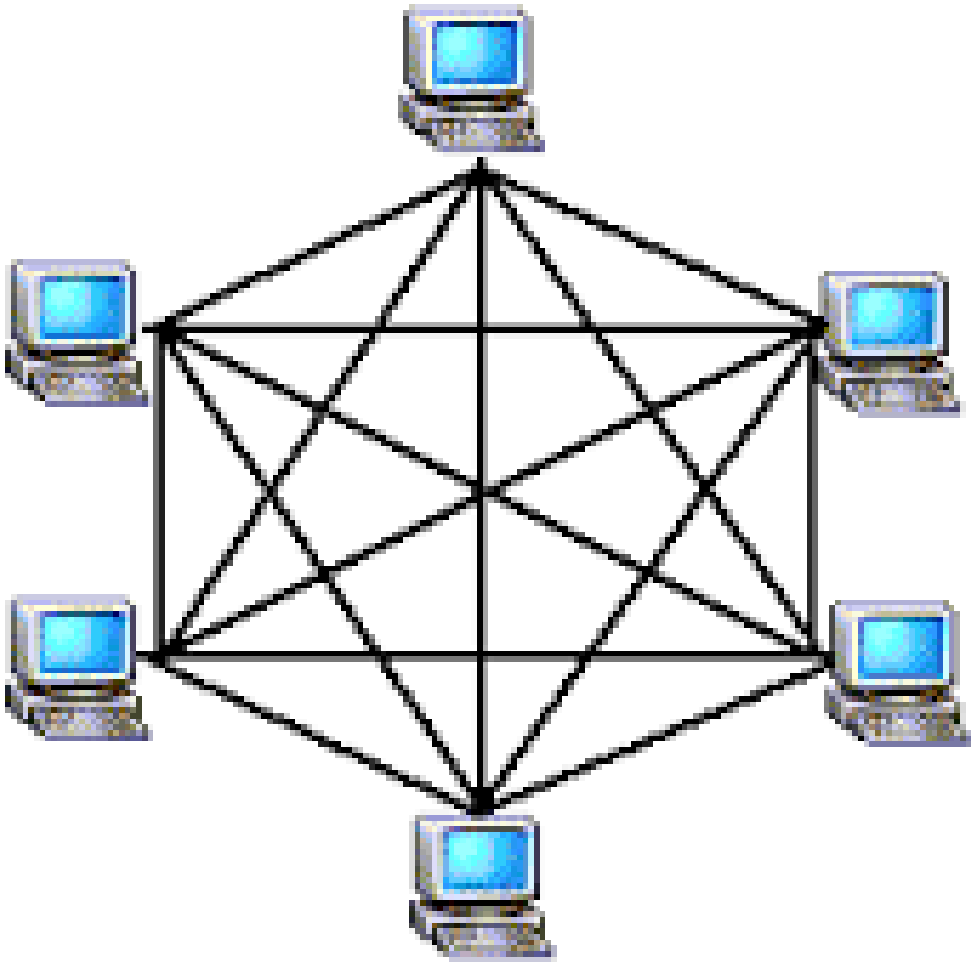
Star topology



Advantages: Easily expanded without disruption to the network, link failure affects only a single host, easy to find faults.

Disadvantages: Requires more copper, not fault tolerant, single point of failure, higher setup cost.

Mesh topology



Advantages: Excellent fault tolerance due to redundancy, expansion without disruption of service.

Disadvantages: Requires a lot of copper, higher setup cost.

Network topologies

Star, tree, mesh configurations are all **point-to-point** topologies:

- Each computer communicates directly with its neighbour.
- Relies on neighbour to rely data around network (except in dense mesh).
- Problems if one critical host fails.
- Star is an extreme example as a central host is present. If this fails. . .

The client-server model

Another view of the internet.

- We can to distinguish between computers that
  - access information (**clients**) and
  - provide information (**servers**).
- A computer can be either of both at the same time.
- The internet is made up of **client** computers which can access information and **servers** which provide/process/sort/distribute information.
- A program becomes a **client** when it sends a request to a **server** and awaits a response.
- A server is a program that offers a service that can be obtained over the network.

Circuit vs packet switching

Circuit switching:

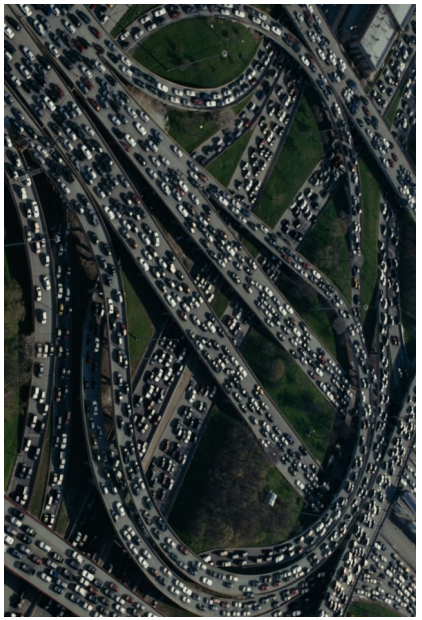
- Set up a dedicated route (**circuit**) between nodes.
- Use the entire circuit for "conversation".

Examples: old-fashioned telephone network, railroads and trains (remote analogy).

Packet switching:

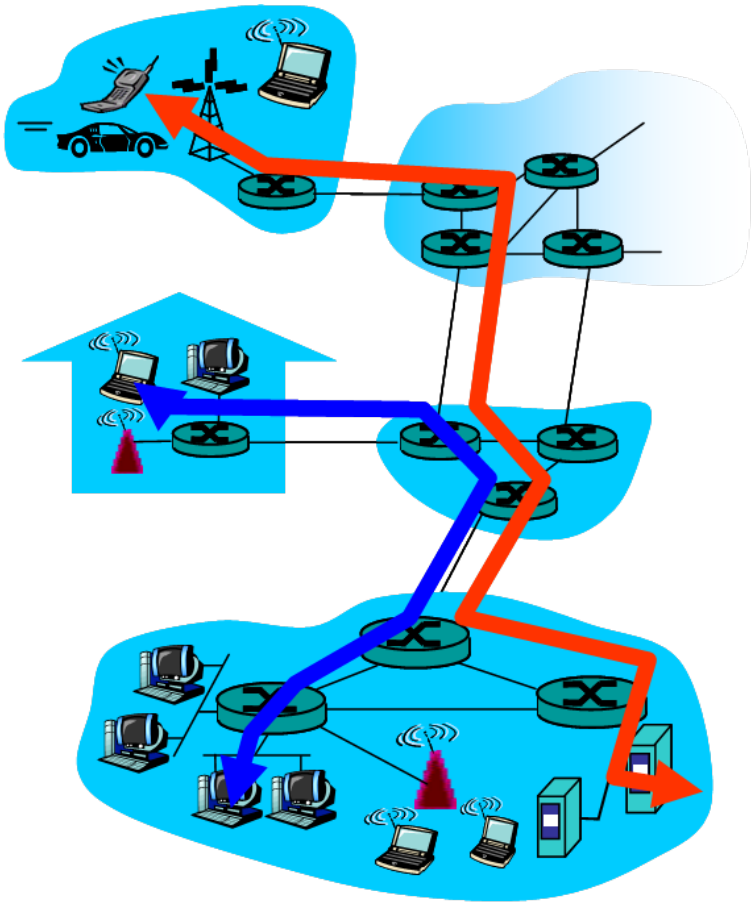
- Divide data into small chunks (**packets**).
- Each packet is delivered independently.

Examples: Internet, mail, motorways and cars (remote analogy).



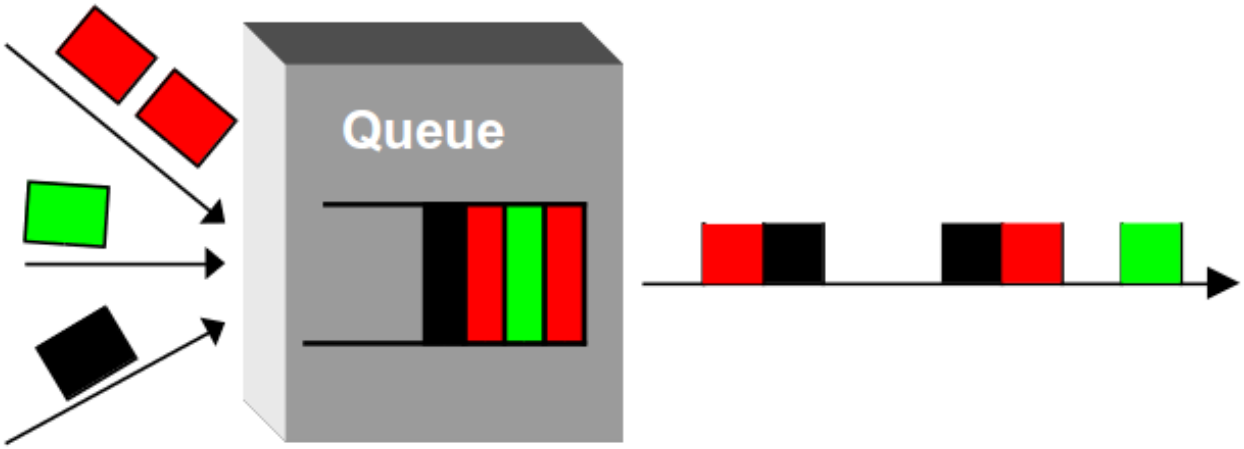
Circuit switched networks

- **Dedicated connection** is established between two stations: end-to-end resources are reserved for a "call".
- **Advantage: guaranteed capacity** (performance) — once a circuit is established, no other network activity will decrease the capacity of the circuit. Resources allocated to a call cannot be used by any other call.
- **Disadvantage: cost** — circuit costs are fixed, independent of traffic.



Packet switched networks

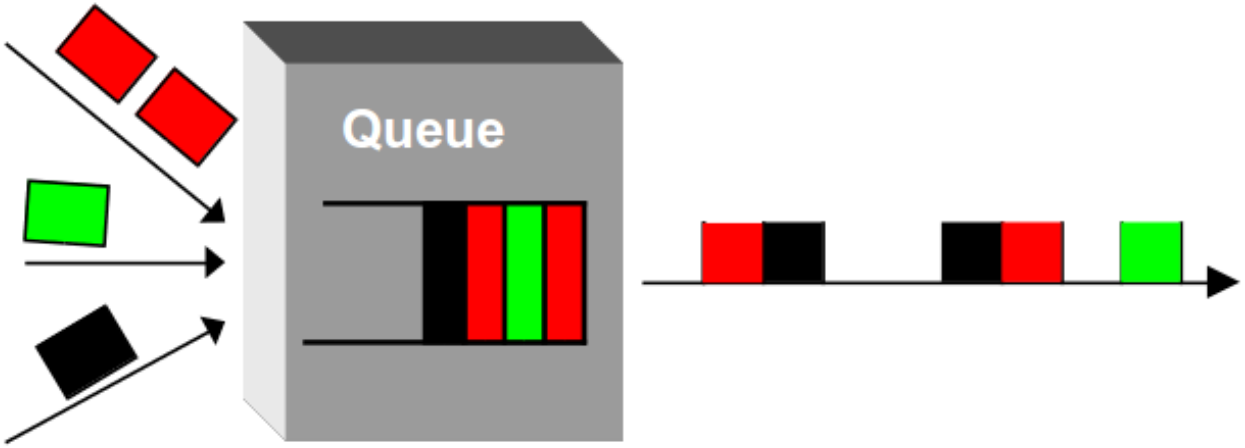
- Network traffic is divided into small **packets**, usually a few hundred bytes in size.
- **Advantage:** packets from different hosts **share** network resources, so fewer interconnections are required. Resources are used **as needed**, not reserved in advance.
  - (Each packet uses full link bandwidth.)
- **Disadvantage:** as activity increases, a given pair of communicating computers receives a smaller share of network capacity.



Packet switched networks

Internet is comprised predominantly of packet switched networks.

- **Statistical multiplexing:** Time division, but **on demand**.
- Reschedule link on a per-packet basis.
- Packets from different hosts are interleaved.
- **Buffer** packets that are competing for the link. **Congestion** may occur.



Resource contention

Problem! Resource demand from multiple messages can exceed the amount available!

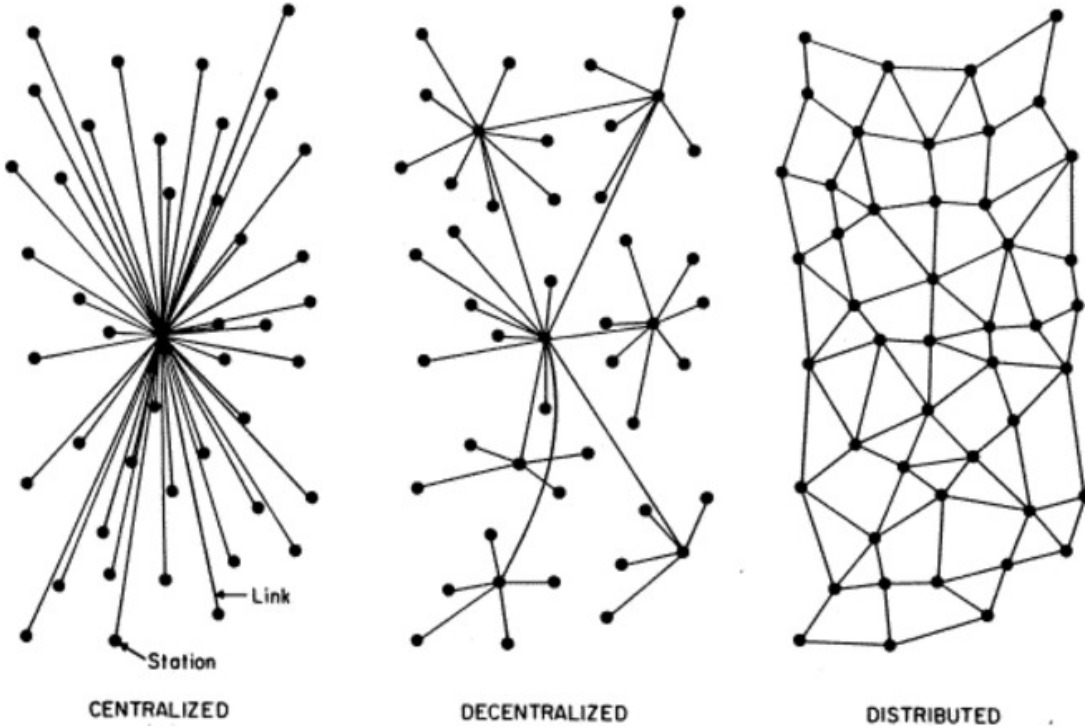
- **Congestion**: packets can queue, have to wait for link to become available.
- **Store and forward**: packets move one hop at a time.
- Entire packet must arrive at router (node) before it can be transmitted on next link.
- Router decides which link the packet should be sent out on.

How to send packets?

- Each packet is marked with:
  - **Source and destination addresses.**
  - **Packet position in whole data.**
  - Packet length.
  - Time to live.
- Receiving hosts must assemble packets to re-create the original message.
- Within the network route packets are passed on by **routers** or **link-layer switches** (junctions in the network with incoming and outgoing links). These look at the destination address and decide the best route to send them.

Routes

- A **route** is the **path** a packet takes through the network graph from the sending end system to the receiving end system.
- With **packet switching**, the path or parts of paths can be shared by multiple communicating end systems.



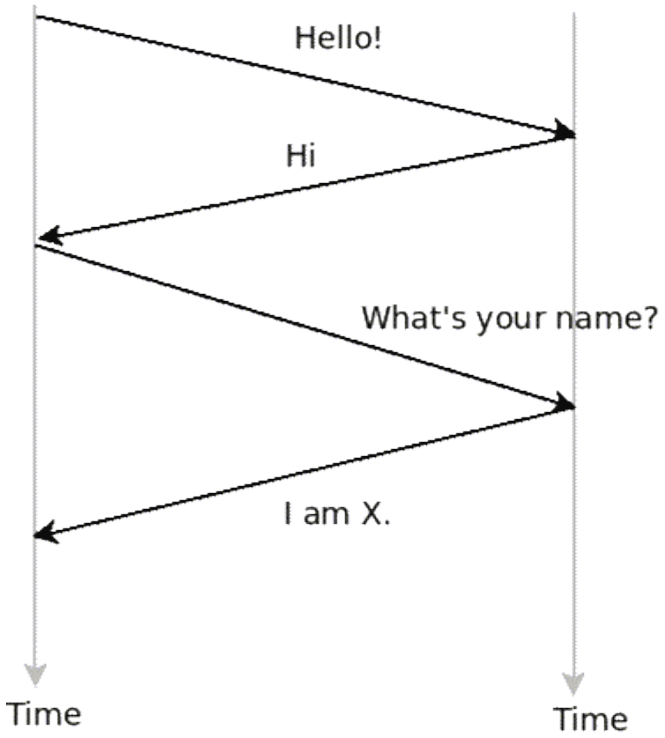
More on this later.

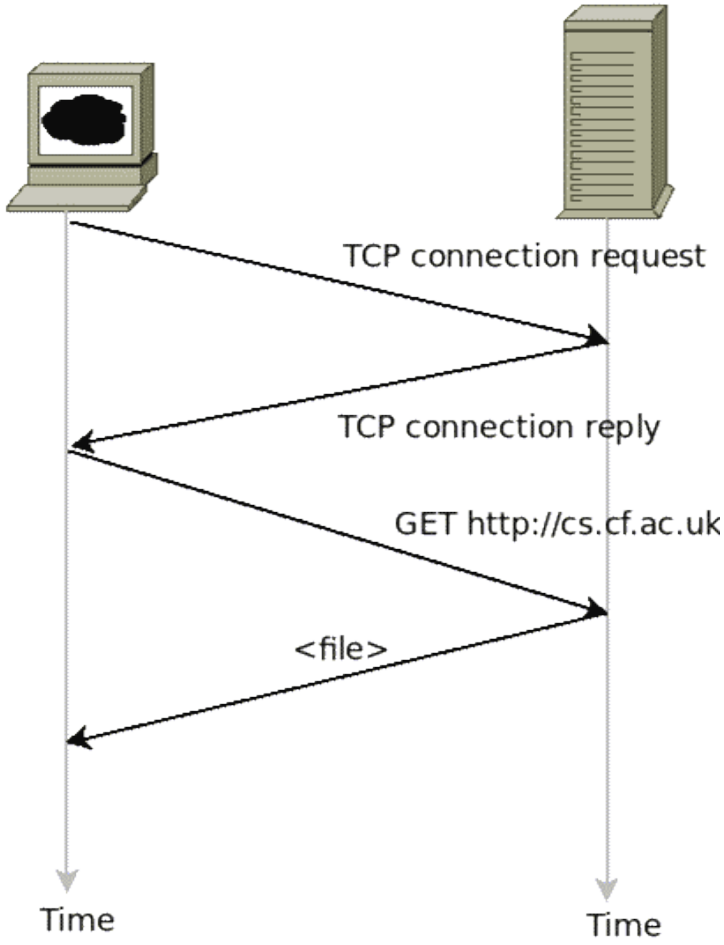
What is a protocol?

- A **protocol** is simply **rules for communication**.
- Humans use protocols all the time:
  - Introducing each other.
  - Asking the time.
  - Ordering a meal in a restaurant.
- Rules of human protocols:
  speak in turns; say "thank you"; in a group, make it clear to whom you are speaking *etc.*
- Two communicating entities have to run the same protocol in order to achieve the task.
- A protocol involves agreement on language, vocabulary, structure and sequence of actions, events *etc.*
- Computers are no different.

Human vs computer protocol

Computer network protocols are similar to human protocols, but the entities are hardware or software components.





Network protocols

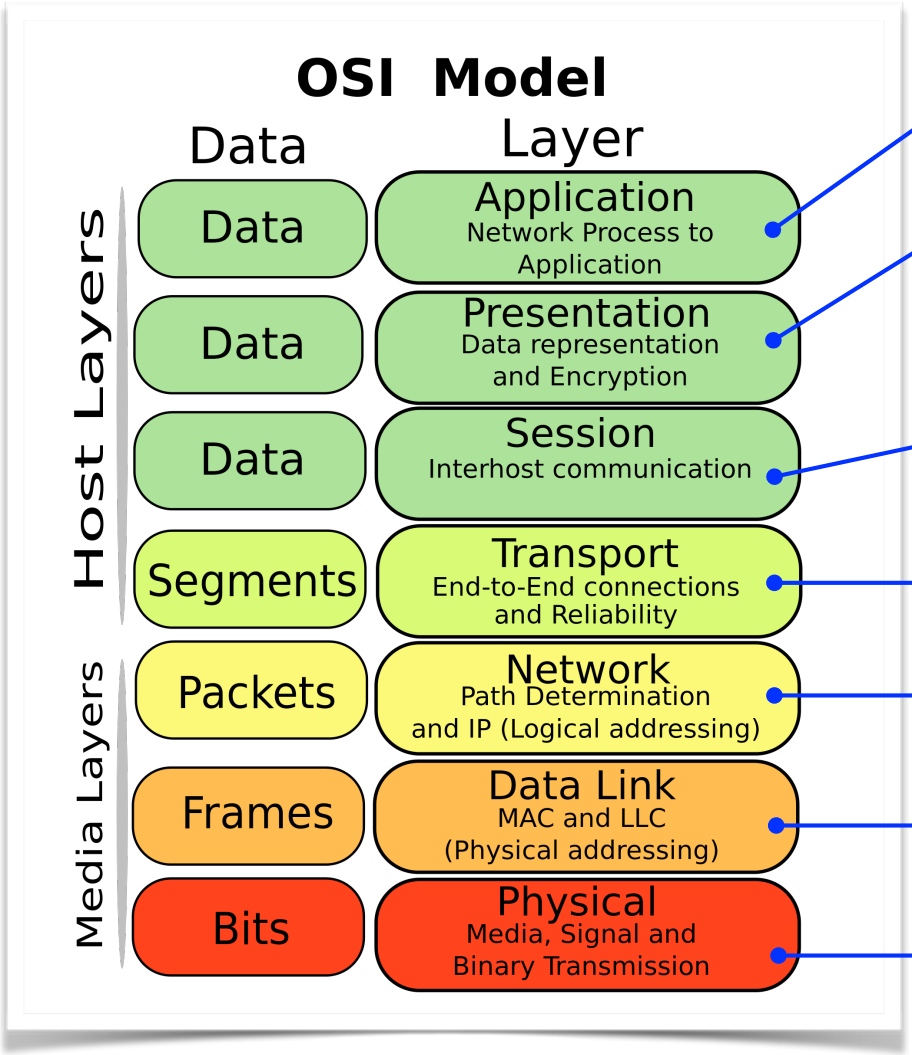
Communicating over a network requires a very complicated set of rules (protocol) to be adhered to. We shall reduce the complexity by employing an abstraction hierarchy.

- Each abstraction layer can be regarded as a black box.
- Well defined inputs and outputs exist, **but**
- Inner workings of layers can be regarded as being independent.
- Thus, new versions/updates/methods can be implemented without affecting whole system.
- Networking is **future proofed** to a great extent.

Communication need only take place at the layer appropriate for the task.

7-Layer OSI Model

- OSI = Open Systems Interconnect (an ISO standard from 1970s)
- **Reference** model of **data communication**
- Basis for understanding how a network protocol should operate



network applications (resource sharing, remote file access, e.g telnet, ftp, http, pop3, smtp, etc.)

formatting of data (e.g. conversion, compression, …) and encryption/decryption

establishment and maintenance of sessions (synchronisation of operations between hosts, checkpoint/ recovery of data exchange, ...)

reliable or unreliable end-to-end delivery (segmentation, acknowledgement, multiplexing)

packet delivery, routing

framing of units of information and error checking

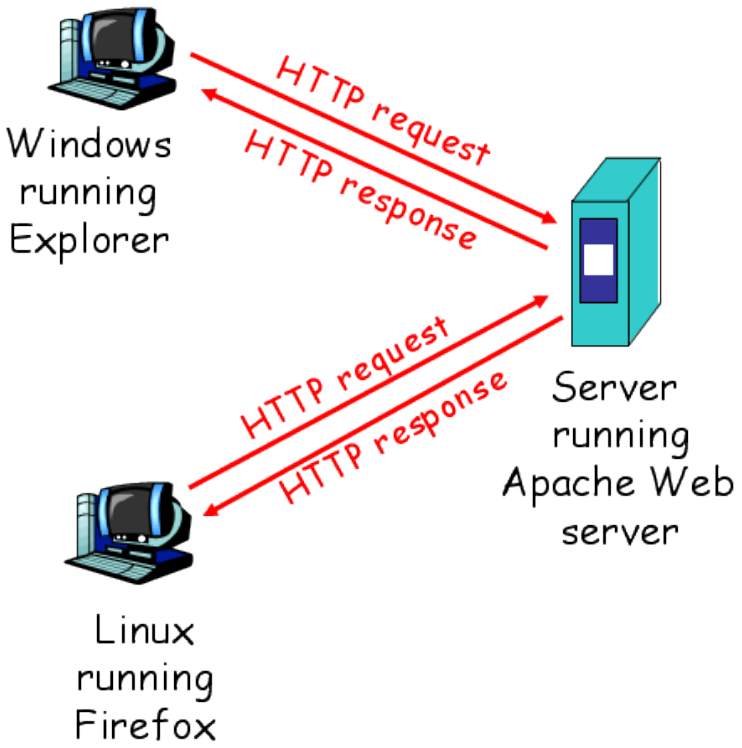
transmission of bits on the physical hardware

Full list of protocols: https://en.wikipedia.org/wiki/List_of_network_protocols_(OSI_model)

HyperText Transport Protocol (HTTP)

Application layer

- HTTP is a protocol to enable a browser to retrieve a web page from a web server.
- Uses Client-Server model:
- Client = browser (requests objects and, when received, displays it).
- Server = web server (receives requests and sends objects to client).



HyperText Transport Protocol (HTTP)

- HTTP uses TCP.
  - Client initiates TCP connection to server (on port 80 usually).
  - Server accepts connection.
  - HTTP messages are exchanged between the client and server.
  - TCP connection closed.

Non-persistent HTTP One object is sent over the TCP connection.

Persistent HTTP Multiple objects sent over single TCP connection. *E.g.* basic page text plus multiple images (each with own URL) to be displayed.

HTTP messages: **request**, **response**.

HTTP request

- First, the browser locates the web server at specified URL.
- It then sents a request, in human-readable ASCII format:

The request line: what we want to see and how we want to communicate:

```
GET /index.php HTTP/1.1
```

- `GET` is the HTTP request method (other options are `POST`, `PUT`, `DELETE`, `HEAD`)
- Second part is the path to the resource.
- The last part indicates that we are using the HTTP 1.1 standard to make the request.

HTTP request

The lines after the request line are the **header**. The header tells the web server about the browser's configuration and the document formats it will accept. Example:

```
Host:   www.cs.cf.ac.uk
User-Agent: Mozilla/4.05
Accept: text/html
Accept-Language: en-us,en
Accept-Encoding: gzip, deflate
Accept-Charset: ISO-8859-1,utf-8
Connection: keep-alive
```

This is followed by an extra blank line to indicate the end of header.

HTTP request

- The **body** it the rest of the message following the header.
- With the `GET` request, the body is empty.
- If form data is submitted using `GET` request, the form data is sent as part of URL:

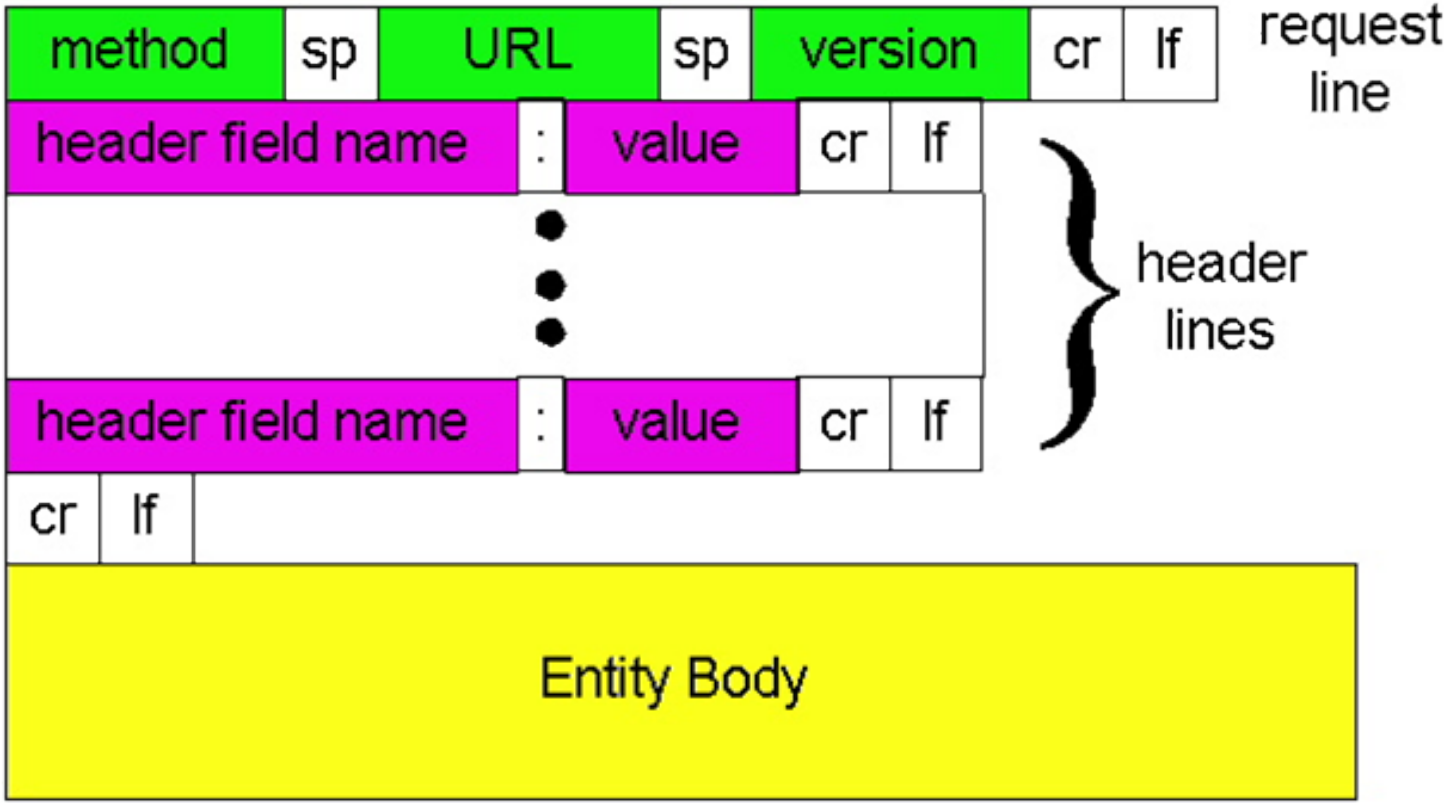
  `?attribute1=value1&attribute2=value2`

  e.g.

  `GET /myscript.php?username=chekov&item=phaser`

- If it is a `POST` request (submitting form data), the form data is sent in the body. `POST` has semantics of changing the state.
- If no form data needs to be sent use `GET`. It has semantics of retrieval.

HTTP request



sp = space
cr = carriage return
lf = linefeed

Other request methods

HEAD	request to just send the **headers** but not the requested object (otherwise identical to `GET`). Useful for retrieving meta-information.
PUT	request to upload a file in the body to location on server specified in the URL.
DELETE	request to delete the file specified in URL.
OPTIONS	request the HTTP methods that the server supports.
TRACE	Echoes back the received request (useful to see what modifications have been made to in on the way).
CONNECT	to establish end-to-end tunnel through a proxy.
PATCH	to apply partial modifications to a resource.

HTTP response

The response has similar format.

- The response line:

  ```
  HTTP/1.1 200 OK
  ```

  First part is the protocol used to communicate. Then the status of the request (200 = success) followed by textual description of the status.

- Then follows the header with the information about the server and the requested document:

  ```
  Date: Mon, 11 Mar 2019 08:20:33 GMT
  Server: nginx/1.8.0
  Last-modified: Tue, 12 Feb 2019 12:15:22 GMT
  Content-type: text/html
  Content-length: 1234
  ```

HTTP response

- The header is followed by an extra blank line to indicate the end of header.
- If the request was successful (200), the body contains the requested data. The data may be a copy of a static HTML document or a result of running a PHP program.

Some HTTP response codes

- `1XX` Informational responses.
- `200 OK` Request succeeded. (All `2XX` codes indicate success.)
- `301 Moved Permanently` Requested object moved to location (specified in response). (All `3XX` have to do with redirection.)
- All `4XX` indicate error on the client part:
- `400 Bad Request` The request could not be understood by the server. The client should not repeat the request without modifications.
- `404 Not Found` Object not found on server.
- `401 Unauthorized` The request requires user authentication.
- All `5XX` indicate error on the server part:
- `500 Internal Server Error`.

HTTP is stateless

- HTTP maintains no record about previous actions by the client.
- TCP (used by HTTP) **is** stateful but only for the time needed to complete successful transmission of the message.
- Recall out discussion of sessions.

Transport Layer

- Transports messages between application layer end systems.
- Two protocols:
  - **TCP** (Transmission Control Protocol)
    - Used by services such HTTP, SMTP, FTP and Telnet.
    - Connection-oriented — guarantees delivery.
    - Monitors segments to ensure reaching destination.
  - **UDP** (User Datagram Protocol)
    - "Connectionless" — no guarantee of delivery (but guarantees integrity).
    - No flow control: if messages are sent too quickly data may be lost.
    - Used for real-time audio-video, interactive games...

TCP (Transmission Control Protocol)

TCP provides: reliable transport, flow control, and congestion control.

Reliable An application can rely on the connection to deliver all of its data without error and in proper order. (Send and forget.)

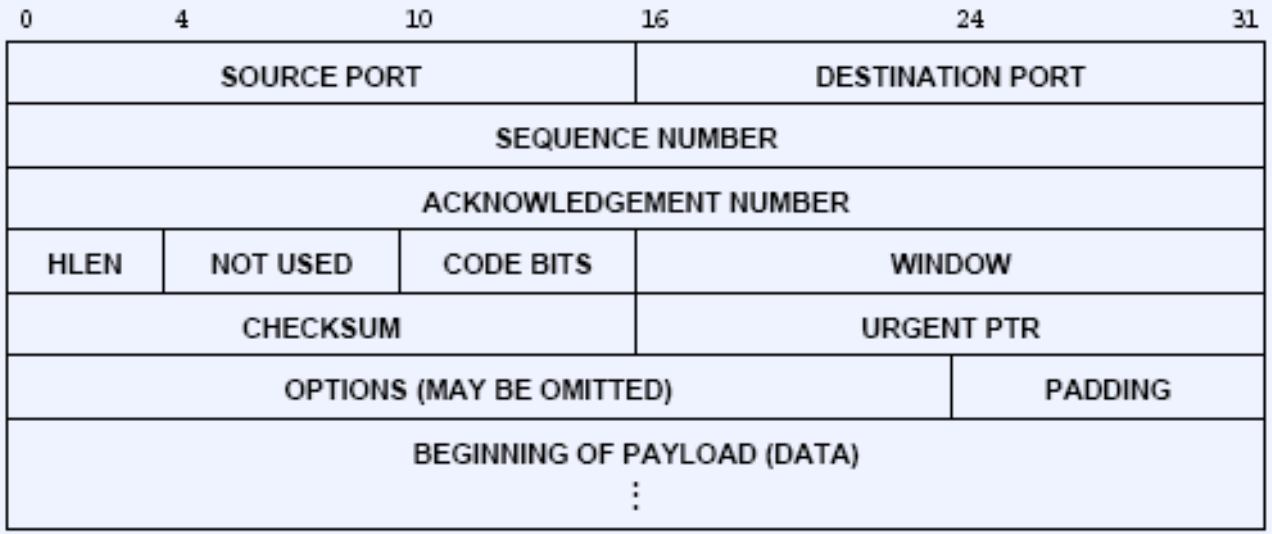
Flow Control Neither side of the connection overwhelms the other side by sending too many packets too fast — buffered at end-system.

Congestion Control If routers get congested (queue overflows), end systems decrease (**throttle**) their send out rates for packets.

TCP (Transmission Control Protocol)

- Closely tied to the Internet Protocol (TCP/IP): data are sent in IP datagrams.
- Guaranteed transfer protocol.
- Uses **sequence numbers** to identify packets.
- Receiver sends an acknowledgement for each packet.
- Lost packets (no acknowledgment from the receiver) can be identified and re-sent.
- Each packet is routed individually.
- Client must re-order packets before re-assembling the data.

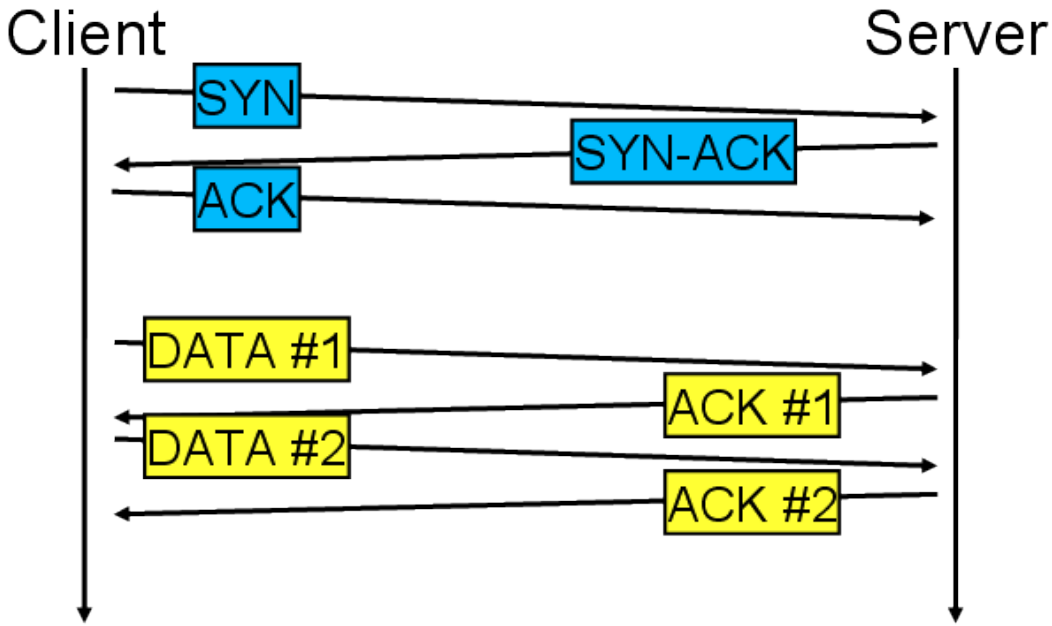
TCP Packet Structure



- **Source Port** and **Destination Port** identify sending and receiving ports.
- **Sequence Number** specifies the number assigned to the first byte of data in the current message.
- **Code bits** carry a variety of control information (flags).
- **Checksum** can be used to verify if data was damaged in transit.

Establishing TCP connection

To establish a connection, TCP uses a three-way handshake.

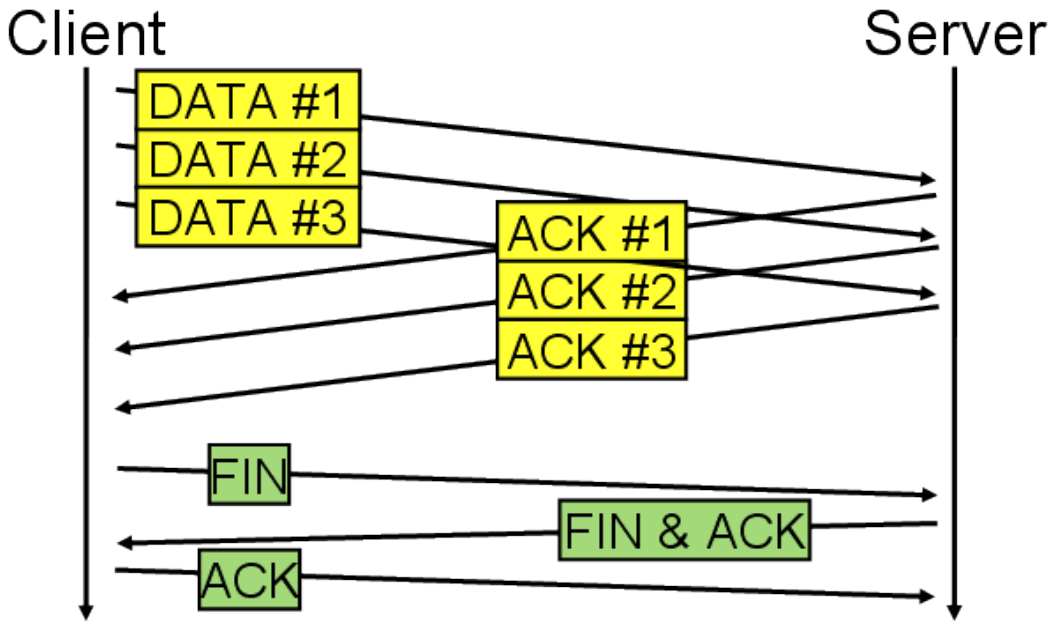


- `SYN` is sent to the server.
- In response, the server replies with `SYN-ACK`.
- The client sends an `ACK` back to the server.

Now both the client and server have received an acknowledgment of the connection.

Closing TCP connection

Similar three-way procedure is used to close a connection.



Ports and sockets

- An end system (host) is identified by an IP address.
- The host could be running multiple processes (servers).
- A process (server) is identified with a **port number** on which it is **listening**.
- Ports are 16 bits numbers. Ports 0–1023 are so-called **well-known ports**, *e.g.* 80=HTTP (web sever); 21 = FTP; 22 = SSH.
- The endpoints of a bidirectional interprocess (e.g. web server – browser) connection are called **sockets**.
- The socket is used as part of the addressing information when constructing packets.

Ports and sockets

- TCP socket is described by a 4-tuple: local/remote IP address, local/remote port.
- TCP/IP headers includes information about
  - the socket to which the segment should be delivered.
  - the socket that it came from (source).
- When a server host first receives a TCP packet, it creates a socket (either in a new process or part of an existing process) with that identity.
- Subsequent communication from the client then uses this server socket.

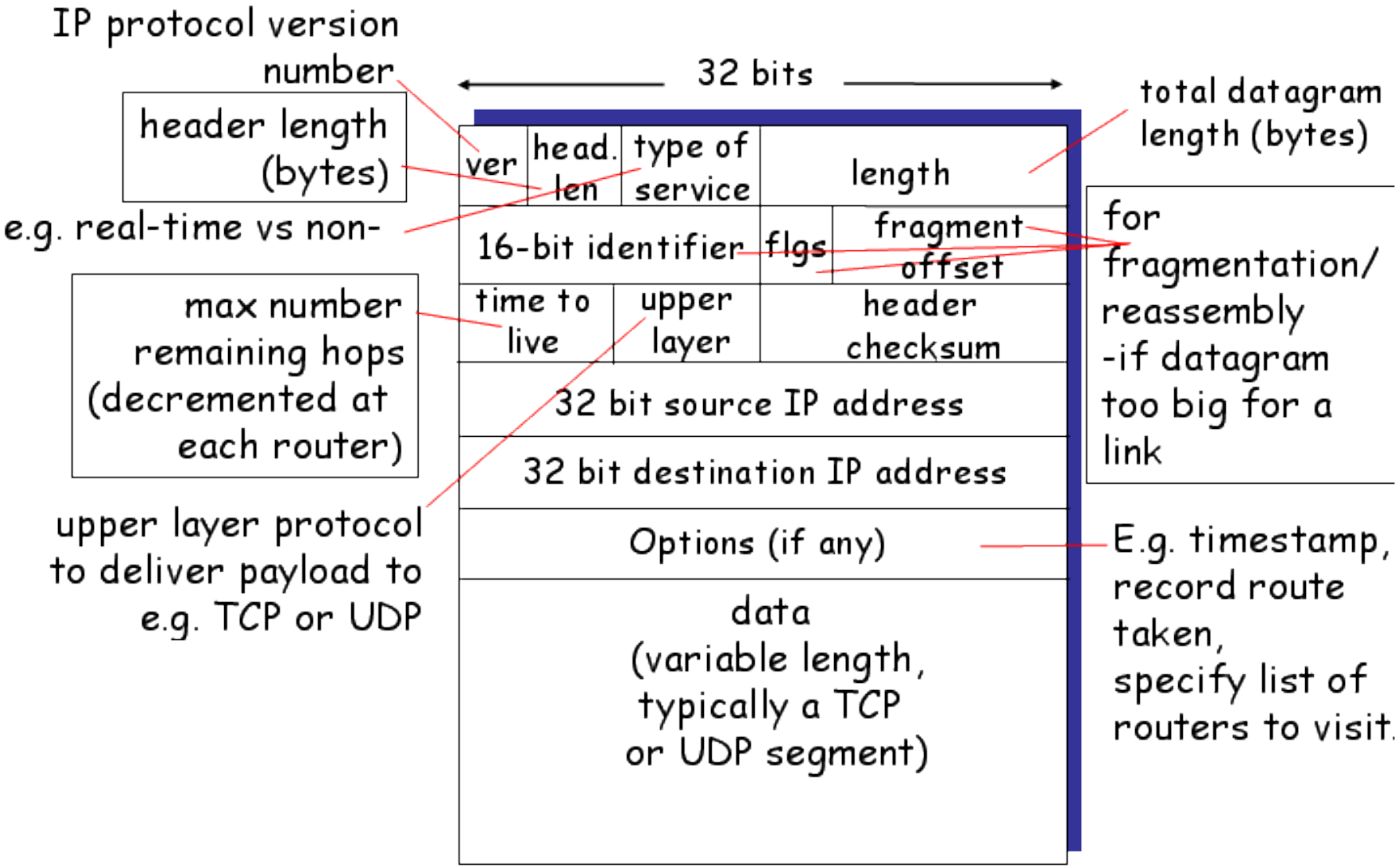
Ports and sockets

- So, an HTTP web server creates a new process (or sub-process) for each client that connects to it.
- That process lasts long enough to conduct the HTTP request/response — which could require multiple TCP packets to be constructed and sent back to the client.
- When the session is closed the web server has no record of the transaction (it is **stateless**) unless additional action was taken to record the client.
  - Cookies.

Network Layer

- Transport Layer (TCP) splits user message into packets (segments) and manages the sending of packets and reconstruction to original message.
- Network Layer manages the journey of the packets (referred to at this level as **datagrams**) across network.
- IP protocol defines format/content of the datagrams.
- Network Layer has router protocols to control the routes that datagrams take in network.

IP datagram


IP protocol version number
header length (bytes)
e.g. real-time vs non-
max number remaining hops (decremented at each router)
upper layer protocol to deliver payload to e.g. TCP or UDP
32 bits
ver
head. len
type of service
length
16-bit identifier
flgs
fragment offset
time to live
upper layer
header checksum
32 bit source IP address
32 bit destination IP address
Options (if any)
data (variable length, typically a TCP or UDP segment)
total datagram length (bytes)
for fragmentation/ reassembly -if datagram too big for a link
E.g. timestamp, record route taken, specify list of routers to visit.

Routing

- When one network is connected to another, a device called a **router** connects both networks and passes data between them.
- In a simple ring network a packet may be routed around the ring until it gets "caught" or gets back to the sender.
- The Internet is not that simple — multiple topologies.
- A router can be connected to more than one network.
- The route selected depends on traffic loads, what backbones are working *etc.*
- Not all packets may be routed over the same route.

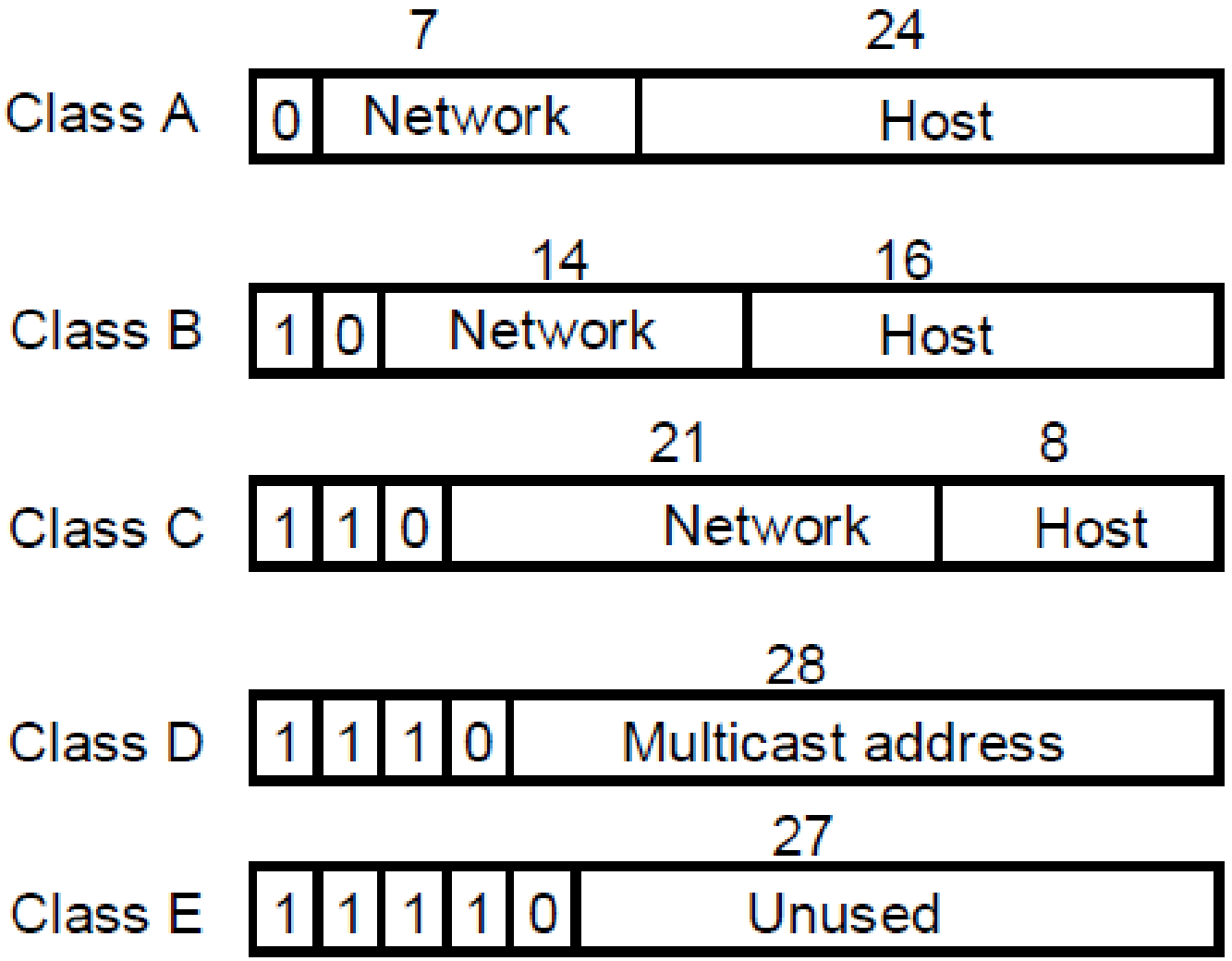
Some routing methods

- **Routing tables** — specific maps (a list of routes) on how to get somewhere. Try one route from list until you succeed.
- **Centralised point** — send all traffic through a centralised node in a network.
- **Nearest neighbour** (centralised adaptive routing) — a central node within each network knows only about its direct connections to the outside world. Send to nearest connection.

IP addresses

- Every host connected to a network must have a unique **IP address** to be identified (a bit like a telephone number).
- An IP address is 32 bits (4 bytes) wide (in IPv4).
- It is composed of two parts:
  - the network number,
  - the host number.
- By convention IP addresses are expressed as four decimal numbers separated by periods, such as `131.251.42.1` (One decimal value of each of the four bytes.)

IP addresses



Domain Names

- As internet grew larger (early 1980s) so did number of hosts: difficult to remember all IP addresses!
- For convenience of humans, internet hosts can have names as well as numbers: these are called **domain names**.
- Which is easier to remember?
  `173.194.67.106` or `google.com`
  Both of these refer to the same web server.
- This is a bit like phone book: we remember peoples names, but don't have to remember many numbers.

Domain Name Service (DNS) provides us with the translation of easier to remember domain names into the IP addresses needed to route over the internet, and vice versa. (This is at the **Application Layer**.)

Domain Name Service (DNS)

We need to map between names and numbers.

- **Domain Name Service (DNS)**.
- A distributed database of domain names and corresponding IP addresses.
- Hosts run DNS or know how to find one.
- The DNS system is, in fact, a whole network (of **Domain Name Servers**).
- If one DNS doesn't know how to translate a particular domain name, it asks another one, and so on, until the correct IP address is returned.

Domain Names

- Domain names have at least two parts, separated by a dot.
- Domain names are read right to left, from general to more specific locations.

  `www.cs.cf.ac.uk`

- The rightmost part after the dot is called the Top Level Domain (TLD). The Top Level Domain serves to broadly categorize the name as to its type or purpose.

Top level domains

Common TLDs include:

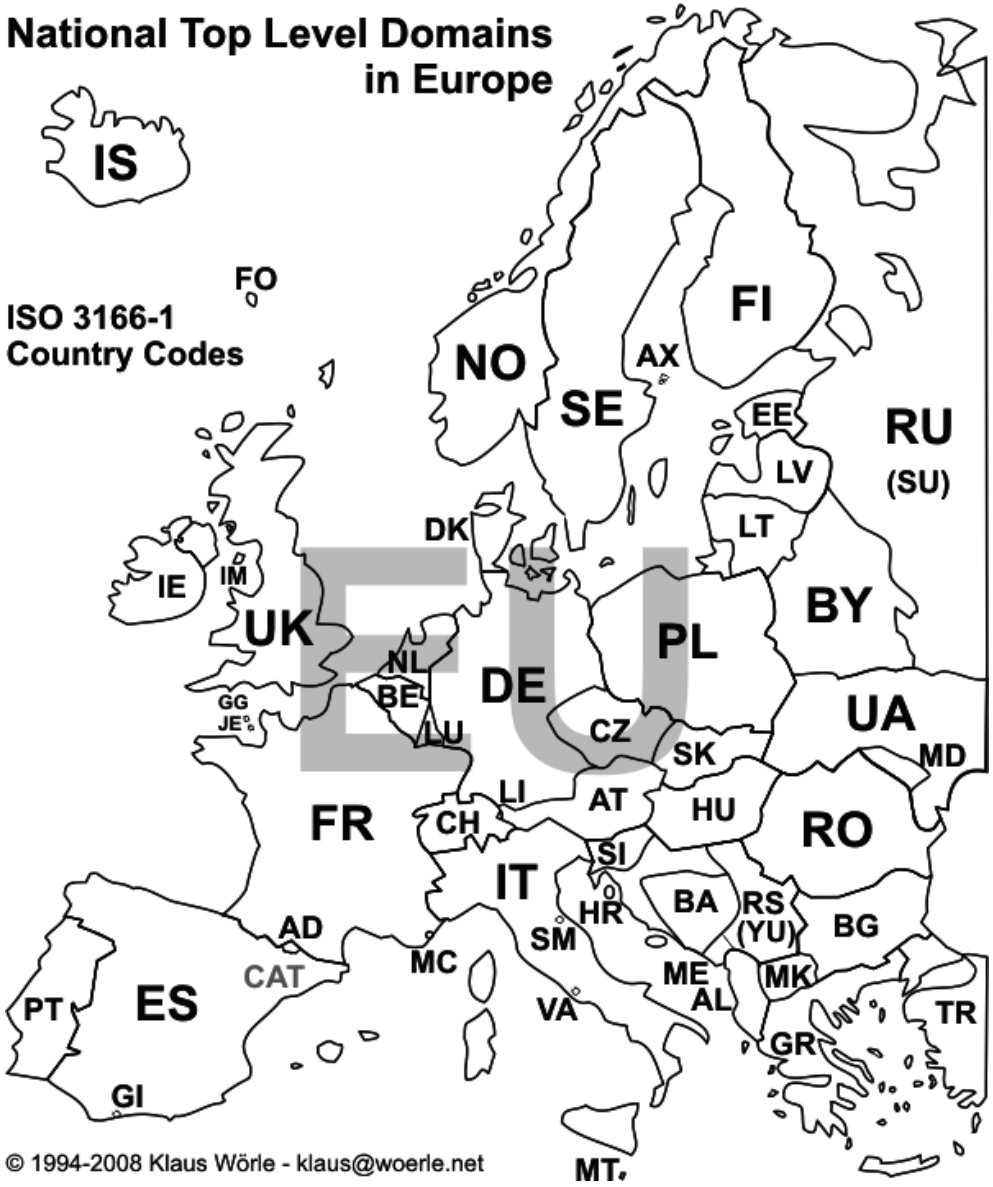
- `.com` — Commercial organisation worldwide (mainly US).
- `.org` — Organizations worldwide (mainly US).
- `.edu` — US educational institutions.
- `.net` — US networks, public service, state-run offices.
- `.gov` — U.S. government.
- `.mil` — U.S. military.
- `.int` — International organisations.

For example: `apple.com`, `mit.edu`, `nasa.gov`.

Domain Names

- There are also country TLDs:
  - `.uk` — UK.
  - `.su` — Soviet Union.
  - `.ru` — Russia.
  - `.tv` — Tuvalu.
  - `.cm` — Cameroon.
- The part of the domain name before the dot is the Second Level Domain (SLD).
- In the UK you can categorise institutions via the last but one rightmost field (`.uk`):
  - `.ac.uk` — academic institutions.
  - `.co.uk` — commercial organisations.

Top Level Domains



(Image by Klaus Wörle)

URI, URL, URN

- **URI** - Uniform Resource Identifier; is a string of characters used to identify a particular resource - by name, location or both.
  - URI generic syntax:

    ```
    URI = scheme:[//authority]path[?query][#fragment]
    ```

- **URL** and **URN** are specific types of **URI** (i.e. subset):
  - **URL** (Universal Resource Locator), a.k.a. web address, - is a URI used to identify the network location of a resource. The syntax conforms to the URI's one.
  - **URN** (Uniform Resource Name) - is a URI that identifies a resource by name in a particular namespace. It uses the `urn:` scheme, e.g. urn:isbn:0-486-27557-4

Link Layer

- Transfers a datagram across a single link from one node/router to the next.
- There are several protocols, *e.g.* Ethernet, WiFi, PPP (Point to Point).
- A datagram may be moved by different protocols on different links of its journey.
- At this level datagram sometimes referred to as a **frame**.
- Hosts and routers have link addresses = **MAC addresses** (can work with non-IP network protocols). Unique identifier assigned to network interfaces.
- Translation between MAC address and IP address is done with Address Resolution Protocol (ARP).

Protocol encapsulation

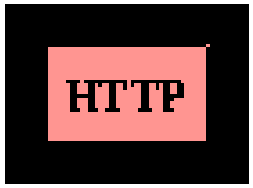
Data from higher layers is **encapsulated** in lower layers.
Layered protocol models rely on encapsulation:

- One protocol can be used for relaying another's messages.
- Encapsulation refers to the practice of enclosing data using one protocol within messages of another protocol.
- The encapsulating protocol must be open-ended:
  - Allowing for arbitrary data to be placed in its messages.
  - Another protocol can then be used to define the format of that data.

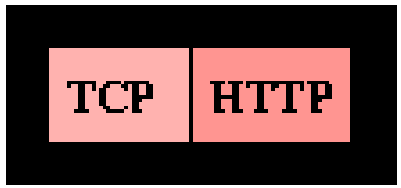
Encapsulation example

Consider an Internet host that requests a hypertext page over a dialup serial connection. The following scenario is likely:

- HyperText Transfer Protocol (HTTP) is used to construct a message requesting the page. The message, (exact format not relevant here), is represented as follows:

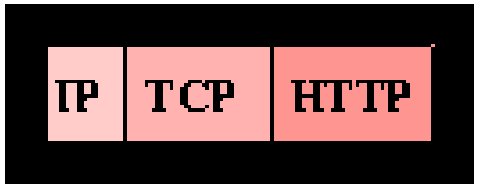


- Transmission Control Protocol (TCP) is used to provide the connection management and reliable delivery that HTTP requires, but does not provide itself. TCP defines a message header format, which can be followed by arbitrary data. So, a TCP message is constructed by attaching a TCP header to the HTTP message, as follows:

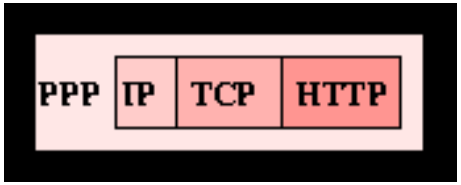


Encapsulation example

- TCP does not provide any facilities for actually relaying a message from one machine to another in order to reach its destination. This feature is provided by the Internet Protocol (IP), which defines its own message header format. An IP message is constructed by attaching an IP header to the combined TCP/HTTP message:

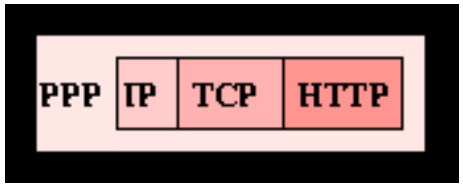


- Although IP can direct messages between machines, it can not actually transmit the message from one machine to the next. This function is dependent on the actual communications hardware. In this example, we're using a dialup modem connection, so it's likely that the first step in transmitting the message will involve the Point-to-Point Protocol (PPP):



Encapsulation example

Note: the PPP encapsulation drawn a little differently, by enclosing the entire message, not just attaching a header.



- This is because PPP may modify the message if it includes bytes that can't be transmitted across the link.
- The receiving PPP reverses these changes, and the message emerges intact.
- The point to remember is that the **encapsulating protocol can do anything it wants to the message**:
  - encrypt it,
  - compress it,
  - expand it,

  as long as the original message is recovered at the other end.

Physical Layer

- Link Layer takes responsibility for entire frames (datagrams).
- Physical Layer takes responsibility for individual bits.
- Protocols vary according to the physical medium (twisted-pair, copper wire, fibre optics).
- What considerations/implications of transmission medium?
  - Interference — electro-magnetic noise, cross-talking.
  - Physical limitation — speed of transmission through wires, optical cables, radio/satellite. Governs amount/size of packets that can be send in given time — affects bandwidth.
  - Signal attenuation in medium — limits range.

Physical Layer

Using wire to transmit signals has been around since the telegram.
However, there are problems with this medium.

- A pair of wires can carry (send and receive) telephone conversation for some distance but suffer from
  - Electromagnetic interference (EMI).
  - Crosstalk.
- Twisted pairs (two wires tightly twisted together)
  - Invented by Alexander Bell.
  - Reduces EMI effects: the difference is voltages is used to carry information.
  - Used in telephone cables, Ethernet *etc.* Can carry several simultaneous telephone conversations (100 voice channels).

Physical Layer

- Coaxial cables (one wire inside another conductor separated by an insulator)
  - Superior rejection of EMI andcCrosstalk.
  - Can support 10,000 analog voice channels.
- Fibre-optic cables
  - Very little outside interference.
  - Can carry very large amounts of data (several gigahertz).
  - For example, laser light modulation carries data at 140 million bits per second. Best telephone rates 56,000 bits per second.
- Radio
  - Limited range (need powerful receivers/transmitters or infrastructure, *e.g.* cellular networks.
  - Security issues.

Other 'popular' tools and technologies

(outside of scope for this lecture, but might be useful to you in the future)

- Web services, using XML, SOAP, WSDL and UDDI
- AJAX and ReST

A variety of tutorials, e.g.

- https://www.w3schools.com/xml/xml_services.asp
- https://www.w3schools.com/xml/xml_soap.asp

Other issues to consider

- Overcoming current limitations (e.g. bandwidth, QoS, language, …)
- Mobile and wireless (e.g. 5G)
- Who governs Internet?
  - Prof. D Marshall: "Internet is like a church..."
    - See: http://users.cs.cf.ac.uk/Dave.Marshall/Internet/node25.html
- Net Neutrality

https://lifehacker.com/how-to-explain-why-net-neutrality-matters-to-your-frien-1820768000