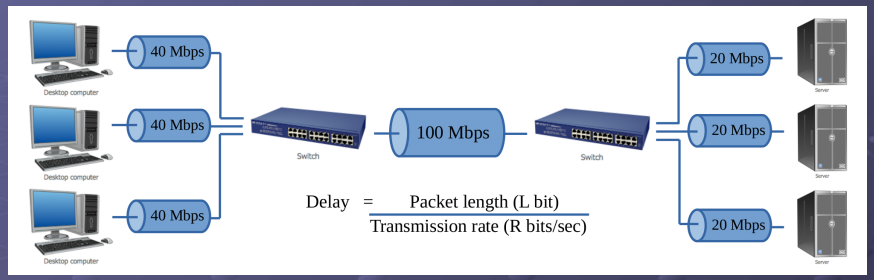
Systems and Devices 2 (Network)
Lec 2a: Application Layer
CiC : 895416

Before we get started ...

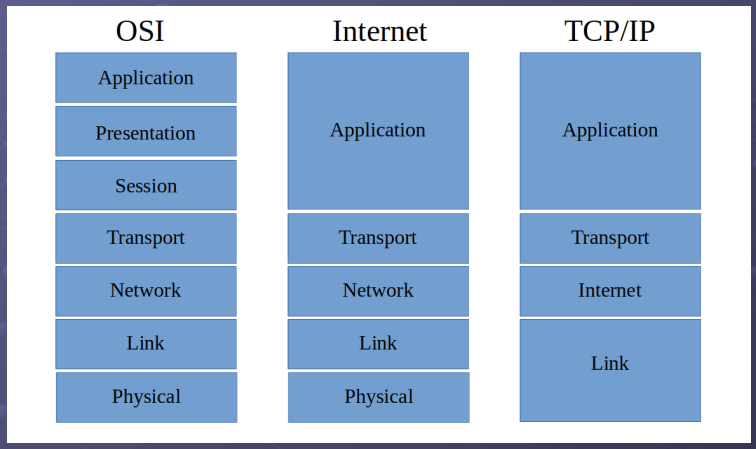
- We have started to consider how communication networks could be structured / organised.
  - Circuit switch vs packet switching
- The main focus of this module will be the Internet protocol stack, the protocols it uses and how we can use them to build systems.
- So starting at the top, “ignoring” other layer for the moment: Application Layer.
  - HTTP, SMTP and FTP
- Also at this level some house keeping stuff:
  - DNS, DHCP and NTP

Quick Quizzz



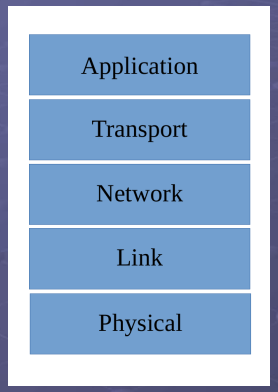
- Messages sent from desktop computers to a server are broken down into 1KB packets.
  - Estimate how long it will take to transfer one packet.
  - What factors can affect this is transfer speed i.e. can you identify the bottlenecks in this network?
  - Hints, bits and bytes, store and forwards.

Network protocol stacks



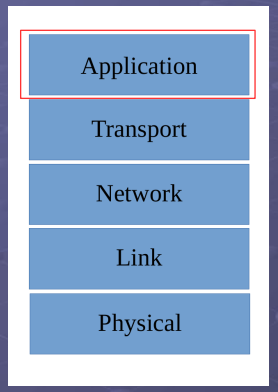
- Hmmmm, what to use: 4, 5, or 7 layer model?

Internet protocol stack



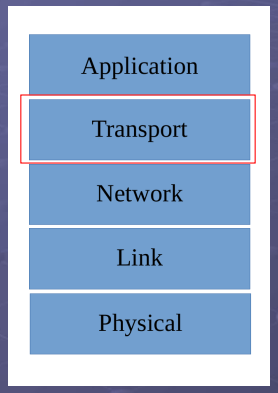
- Application
- Transport
- Network
- Link
- Physical

Internet protocol stack



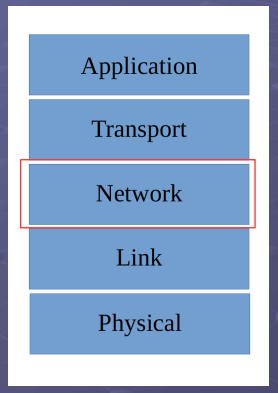
- Application (layer 5)
  - API to network e.g. used by web browser, email etc. A range of protocols to support different services, transferring **messages**.
    - HyperText Transfer Protocol (HTTP)
    - File Transfer Protocol (FTP)
  - Also, contains house keeping protocols to identify and allocate network addresses.
    - Domain Name System (DNS)
    - Dynamic Host Configuration Protocol (DHCP)
- Transport
- Network
- Link
- Physical

Internet protocol stack



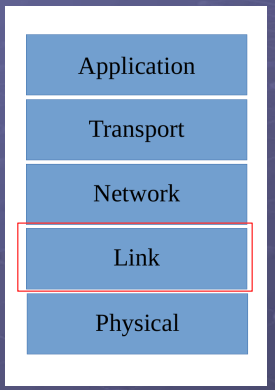
- Application
- Transport (layer 4)
  - Breaks messages down into **segments** that will be transferred. Also deals with error detection, flow / congestion control and retransmission in the event of lost packets.
    - Transmission Control Protocol (TCP)
    - User Datagram Protocol (UDP)
- Network
- Link
- Physical

Internet protocol stack



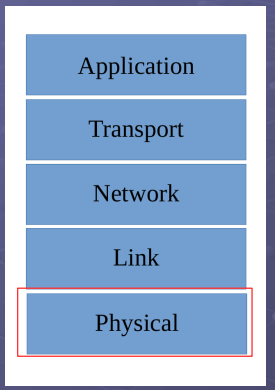
- Application
- Transport
- Network (layer 3)
  - Routing techniques to direct packets (**datagrams**) from one host to another across the network of networks that form the Internet. All based around:
    - Internet protocol (IP)
    - Note, routing, here be dragons :)
- Link
- Physical

Internet protocol stack



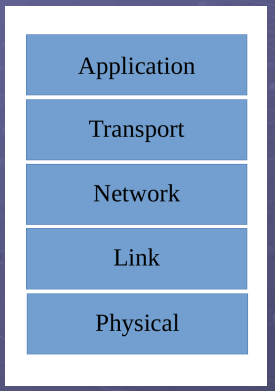
- Application
- Transport
- Network
- Link (layer 2)
  - Moves packets (**frames**) from one node (PC, router, switch etc) to another, error detection / correction. Different protocols depending on transmission medium used, we will focus on:
    - Ethernet protocol
  - Also, contains house keeping protocols to identify network addresses used.
    - Address Resolution Protocol (ARP)
- Physical

Internet protocol stack

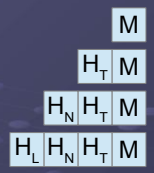


- Application
- Transport
- Network
- Link
- Physical (layer 1)
  - Defines how bits within a frame are transferred, link dependant e.g. twisted-pair copper wire, or fibre optic cable. Specifies link hardware requirements i.e. distances, voltages, connectors …
    - Category 5 cable (Cat 5) : twisted pair copper cable with RJ45 connectors

Internet protocol stack



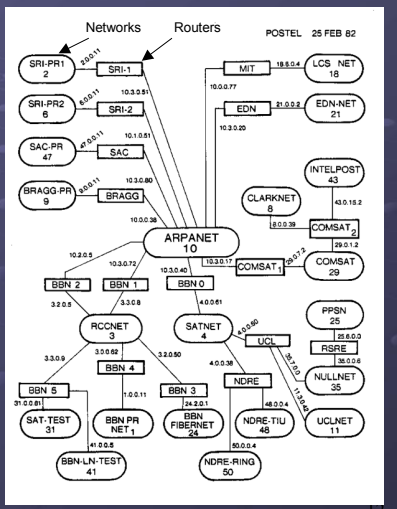
- Application layer : message
- Transport : segment
- Network : datagram
- Link : frame
- Physical



- As data goes down the protocol stack additional headers (fields) are added to the packet to implement each protocol.
  - Encapsulation
- As data goes up the protocol stack headers are removed as packets are received by hosts, routers etc.
  - Decapsulation

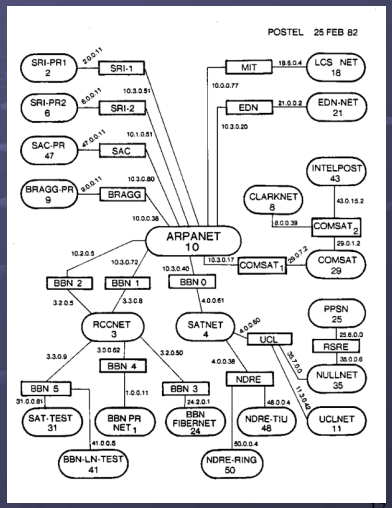
The Internet

- A network of networks
  - Started in 1969, Advanced Research Projects Agency (US DoD) funding the ARPANET project.
    - A packet switched network.
    - Aims: to share computing resources across a fault tolerant network.
    - Figure shows class-A network structure in 1982.
  - Video : https://bit.ly/363xsPg



The Internet

- A key requirement in connecting networks together is that they all speak the same “language”.
  - In 1978 we had the birth of the Transmission Control Protocol (TCP) and the IP protocol (IP), where each machine is given an unique number:
    - IPV4 : 32bit address
      - Typical broken down into 8bit chunks 0.0.0.0 – 255.255.255.255
    - IPV6 : 128bit address

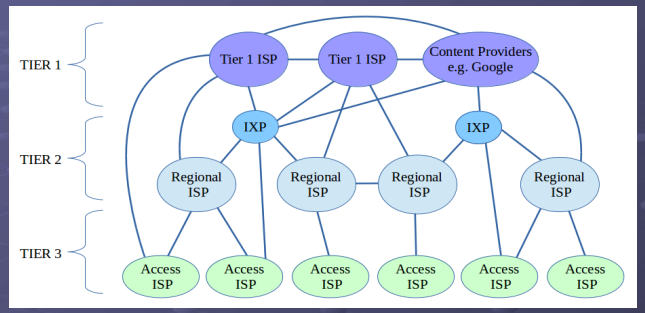


The Internet

- In the 1980’s we see a shift away from the DoD to other government departments, industrial & academic partners:
  - National Aeronautics and Space Administration (NASA)
  - National Science Foundation (NSF)
  - Computer Science Network (CSNET).
  - Joint Academic Network (JANET)
- This move resulted in a different funding model, one based on Tiers i.e. bandwidth used.
- URL : https://en.wikipedia.org/wiki/JANET

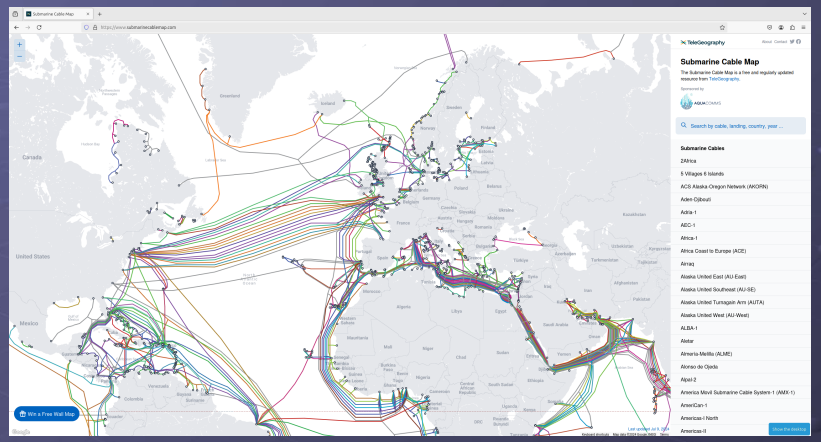
janet	
Motto	the UK's research and education network
Predecessor	SERCnet[1]
Formation	1 April 1984
Type	National research and education network
Purpose	To manage the operation and development of the UK's national education and research network
Headquarters	Harwell, Oxfordshire, United Kingdom
Region served	UK
Director (Jisc Technologies)	Tim Kidd
Website	www.ja.net/janet
Formerly called	Janet(UK); JANET

The Internet



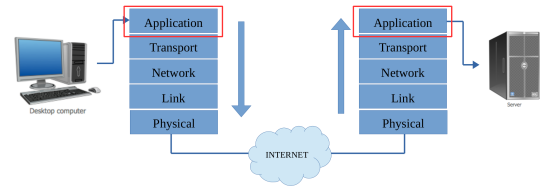
- Tier 1 : backbone, settlement-free connections, large telecommunications companies or content providers
  - Lower tiers pay for connection to / traffic used.
- Tier 2 : regional ISP, Internet exchange point (IXP)
- Tier 3 : access ISP
- Videos : https://bit.ly/3Wt6SrT, https://bit.ly/3LQJdwM

Submarine Cable Map



- Amazon, Google, Microsoft, Meta, AT&T, Colt ...
  - URL: https://www.submarinecablemap.com/

Network basics



- Client server model
  - Running on each machine is a process. The process that initiates the communication is the Client. The process that is waiting to be contacted to is the Server.
  - Each machine is identified by its 32bit IP address.
  - Messages are sent / received through an API called a socket, each socket is identified by its Port number.
    - Clients and server may have multiple open sockets (processes) running at any time.

Demo

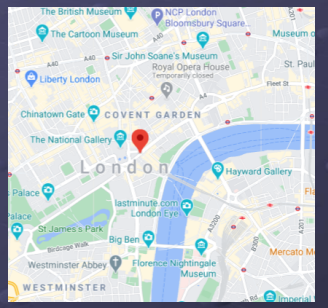
```
mike@mike-Aspire ~ $ ping www.google.co.uk
PING www.google.co.uk (216.58.213.3) 56(84) bytes of data.
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=1 ttl=110 time=11.0 ms
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=2 ttl=110 time=11.4 ms
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=3 ttl=110 time=11.0 ms
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=4 ttl=110 time=11.0 ms
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=5 ttl=110 time=11.0 ms
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=6 ttl=110 time=10.9 ms
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=7 ttl=110 time=11.0 ms
64 bytes from ber01s14-in-f3.1e100.net (216.58.213.3): icmp_seq=8 ttl=110 time=10.9 ms
^C
--- www.google.co.uk ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7008ms
rtt min/avg/max/mdev = 10.948/11.075/11.422/0.180 ms
```

Location

City	London
Region	England
Postal Code	EC1A
Coordinates	51.5085,-0.1257
Timezone	Europe/London
Local Time	October 05, 2020 | 02:43 PM
Country	United Kingdom

Connection

Hostname	lhr25s25-in-f3.1e100.net
Address type	IPv4
ASN	AS15169 Google LLC
Organization	Google LLC (google.com)
Route	216.58.213.0/24
Abuse Contact	network-abuse@google.com
Privacy	VPN ✗ Proxy ✗ Tor ✗ Hosting ✗



- Ping
  - Fully qualified domain name (FQDN)
  - IPv4 address 32 bit number represented as four octet values (0 – 255).

Demo

```
mike@mike-Aspire ~ $ ifconfig
enp8s0    Link encap:Ethernet  HWaddr 30:65:ec:82:4c:bd
          inet addr:192.168.0.100  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::7ebf:c4e4:e1bf:3674/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:487164 errors:0 dropped:0 overruns:0 frame:0
          TX packets:709226 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:248834455 (248.8 MB)  TX bytes:411936382 (411.9 MB)
          Interrupt:19
```

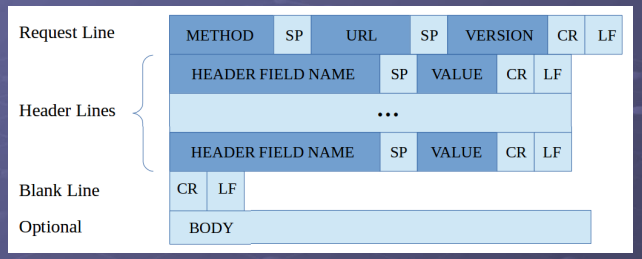
```
mike@mike-Aspire ~ $ traceroute www.google.co.uk
traceroute to www.google.co.uk (216.58.213.3), 30 hops max, 60 byte packets
 1  192.168.0.254 (192.168.0.254)  0.595 ms  0.640 ms  0.756 ms
 2  csestr0-50.york.ac.uk (144.32.51.254)  3.055 ms  3.515 ms  4.110 ms
 3  tftastr0-2419.ospf.york.ac.uk (10.16.35.177)  2.994 ms bsdcstr0-2418.ospf.york.ac.uk (10.16.35.173)  3.185 ms tftastr0-2419.ospf.york.ac.uk (1
0.16.35.177)  3.210 ms
 4  tftafab3-2402.ospf.york.ac.uk (10.16.23.133)  4.629 ms  6.431 ms bsdcfab2-2401.ospf.york.ac.uk (10.16.23.129)  3.535 ms
 5  sentry-659.york.ac.uk (144.32.65.69)  3.612 ms  3.752 ms  3.832 ms
 6  janetrouter2.york.ac.uk (144.32.255.227)  4.117 ms  2.595 ms  2.512 ms
 7  ae31.yorkbs-rbr1.ja.net (146.97.150.85)  2.573 ms  2.653 ms  2.729 ms
 8  ae6.leedlu-rbr1.ja.net (146.97.71.45)  3.049 ms  3.153 ms  3.454 ms
 9  ae26.manckh-sbr2.ja.net (146.97.36.221)  7.155 ms  7.267 ms  7.335 ms
10  ae29.erdiss-sbr2.ja.net (146.97.33.41)  8.365 ms  7.785 ms  7.832 ms
11  ae31.londpg-sbr2.ja.net (146.97.33.21)  35.762 ms  29.948 ms  29.825 ms
12  ae29.londhx-sbr1.ja.net (146.97.33.1)  11.426 ms  11.545 ms  11.775 ms
13  72.14.217.18 (72.14.217.18)  12.309 ms  12.353 ms  12.430 ms
14  * * *
15  172.253.68.212 (172.253.68.212)  10.261 ms 209.85.252.180 (209.85.252.180)  13.283 ms 74.125.242.97 (74.125.242.97)  13.171 ms
16  74.125.242.115 (74.125.242.115)  11.211 ms 74.125.242.114 (74.125.242.114)  10.110 ms 74.125.242.115 (74.125.242.115)  10.811 ms
17  209.85.250.185 (209.85.250.185)  13.513 ms 216.239.59.77 (216.239.59.77)  13.485 ms  13.512 ms
18  lhr25s25-in-f3.1e100.net (216.58.213.3)  11.107 ms  10.967 ms 216.239.58.132 (216.239.58.132)  10.149 ms
mike@mike-Aspire ~ $
```

- Traceroute
  - To see all the connections between my office and London.

HTTP

- HyperText Transfer Protocol
  - RFC 2616: https://tools.ietf.org/html/rfc2616 (HTTP1.1)
  - Developed by a team at CERN lead by Tim Berners-Lee (HTTP, HTML) in the 1990s
    - A web page (document) consists of objects (HTML files, JPEG, GIF image, Java applet etc), addressed by a URL.
  - Client-Server model, default port 80.
  - Communicates across TCP links, these can be:
    - Non-Persistent : closed after each object is transferred.
    - Persistent Connections : multiple objects sent over the same link
      - Reduce request latency
  - Stateless protocol : no information recorded about previous connections
    - Cookies can be used to maintain session information across different pages.

HTTP



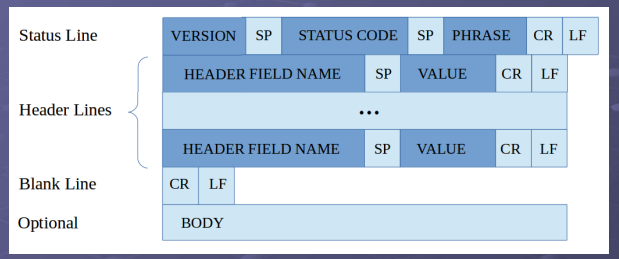
GET, POST, HEAD
OPTIONS, PUT, DELETE,
TRACE, CONNECT

HTTP/1.0 HTTP/1.1

Host: <hostname>:<port>
User-Agent: <client>
Accept: <MIME>

- Request message (client => server)
  - Method: plain text, human readable, operation to perform.
  - URL: identifies object to be accessed (relative or absolute).
  - Version: HTTP standard used.
  - Headers: zero to lots, additional information e.g. content-type
    - Multi-purpose Internet Mail Extensions (MIME) e.g. text/html.
  - Body: data to be posted (transferred) etc.

HTTP



HTTP/1.0 HTTP/1.1

1XX 2XX 3XX 4XX 5XX

OK, Bad request, Not found, Move permanently

Server: <web server>
Content-Length: <bytes>

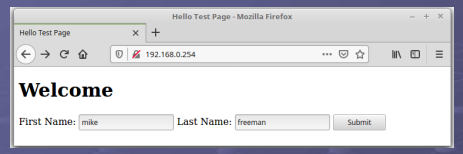
- Response message (server => client)
  - Version: HTTP standard used
  - Status code / Phrase: plain text, outcome of request
  - Headers: zero to lots, additional information e.g. server software (Apache), content length (body size) ...
  - Body: requested object e.g. HTML text, image, video …

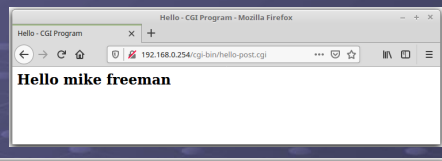
HTTP

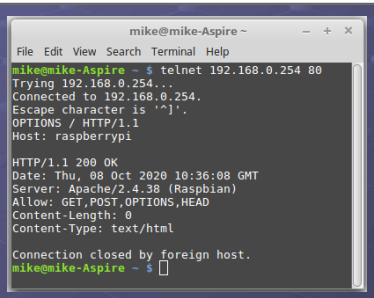
Type	Status Code	Description
Informational (1XX)	100	Continue
	101	Switching Protocols
	102	Processing
Success (2XX)	200	OK
	201	Created
	202	Accepted
Redirection (3XX)	300	Multiple Choices
	301	Moved Permanently
	302	Found
Client Error (4XX)	400	Bad Request
	401	Unauthorized
	402	Payment Required
Server Error (5XX)	500	Internal Server Error
	501	Not Implemented
	502	Bad Gateway

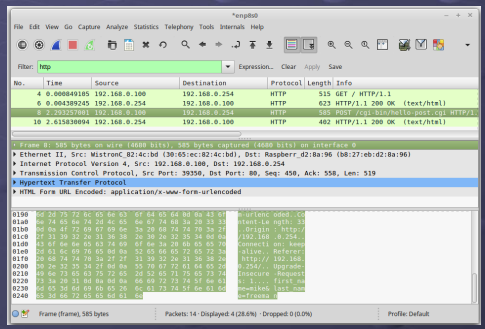
- Status codes (snapshot, there are lots …)
  - https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Demo









- Telnet and Wireshark
  - Telnet: unencrypted text communication, normally port 23.

Pause to consider ...

- What does this show us?
  - The joy of a protocol stack, minimises what we need to know i.e. we could “ignore” the lower layers in our stack.
    - Aims to avoid information leakage.
    - Can result in redundancy, repeated operations, but aim to minimise complexity at any one layer.
  - We need to identify the tasks to be performed and the information needed. Then decide how these will be implemented, handshakes / acknowledgements / transfers are performed i.e. a protocol, requests and responses.
  - We need standards
    - This is a key element of this module i.e. understanding how a network operates and the protocols it uses to achieve this.

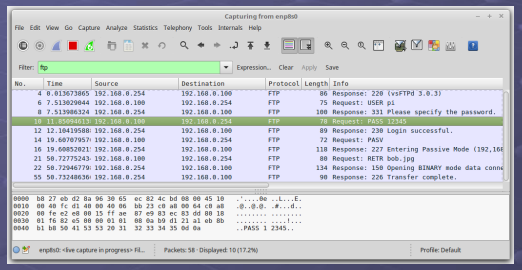
FTP

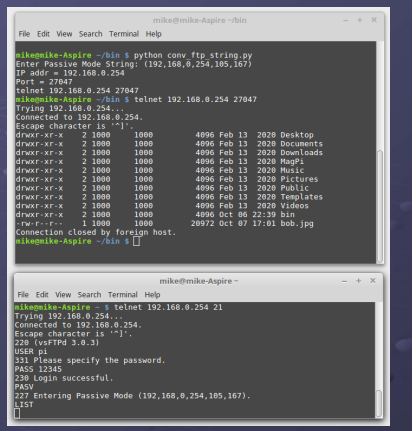
- File Transfer Protocol
  - RFC 114: https://tools.ietf.org/html/rfc959
  - Pre-internet communication protocol, dating back to 1971
    - In its original form less common now days owing to lack of encryption / security issues. Secure version available FTPS, SFTP :)
  - Client-Server model, uses separate ports for control (21) and data (20), ignoring PASV and ACTIVE modes :)
    - Both FTP and HTTP used to transfer files.
    - For FTP control is termed out-of-band, where as HTTP it is in-band.
  - Communicates across TCP links, these are:
    - Control: uses a persistent connection, maintained during sessions
    - Data: uses a non-persistent connections, a new link created for each transfer.
  - Stateful protocol : maintains user information e.g. current working directory and other flags etc = memory on server

Demo

USER, PASS, PASV, CWD, LIST, SIZE, DELE, STOR, RETR, QUIT

1XX, 2XX, 3XX, 4XX, 5XX, 6XX





- FTP commands (protocol, not client)
  - https://en.wikipedia.org/wiki/List_of_FTP_commands
- FTP status codes
  - https://en.wikipedia.org/wiki/List_of_FTP_server_return_codes

SMTP

- Simple Mail Transport Protocol
  - RFC 5321: https://tools.ietf.org/html/rfc5321
  - Pre-internet communication protocol, dating back to 1982
- SMTP server uses an end-to-end model, default port 25 (587).
- Communicates across TCP links, these are:
  - Persistent connection, maintained during sessions
- Body must be 7bit ASCII, which adds complexities
  - Multi-purpose Internet Mail Extensions (MIME) used to encode binary files other character sets e.g. like PPM images in SYS1.
- SMTP is a delivery protocol only, mail is "pushed" to a destination mail server i.e. user mail-box
  - User clients then use the Post Office Protocol (POP) or the Internet Message Access Protocol (IMAP) to retrieve emails.

Go and research

- Have a look at how the SMTP functions
  - RFC: https://datatracker.ietf.org/doc/html/rfc5321
  - How does an email get from your email client to someone's mailbox?
  - How are emails managed on the server?
  - How is text and images sent in an email?
  - We will be using this in the lab ...

Summary

- The main purpose of a network is to transfer data between machines
  - An obvious statement, but now we need to consider how this can be implemented.
    - We have looked at two file transfer protocols: HTTP and FTP.
  - Communications through the socket API, using ports:
    - 0 to 1023: system ports, assigned to core protocols.
    - 1024 to 49151: registered ports, specific applications (IANA)
    - 49152 to 65535: ephemeral ports, short-lived, OS dependent.
  - However, some unanswered questions
    - How is a machine assigned an IP address?
    - How do we convert a URL (www.google.com) into an IP address (216.58.213.3)?
    - What is happening in the lower layers of the protocol stack?