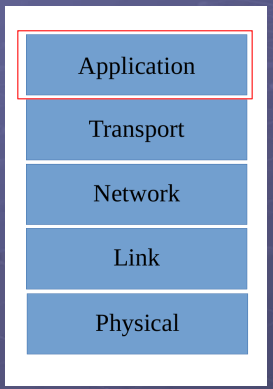
Systems and Devices 2 (Network)
Lec 2b: Application Layer

Before we get started ...

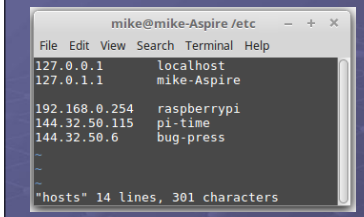
- We considered the top level view of two file transfer protocols: HTTP, FTP and looked at SMTP.
  - How did your research into SMTP go?
    - Not expecting you to be an expert, more of an exercise to see if you start to understand the ideas, the structure of a protocol.
- We now need to consider how the Internet is organised e.g. how we identify different machines
  - Domain Name System (DNS): mapping domain names to IP
  - DNS servers: Berkeley Internet Name Domain (BIND)
  - DNS tools: Domain Information Groper (DIG)
- Then consider how IP addresses are allocated
  - Static or dynamic IP.
  - Dynamic Host Configuration Protocol (DHCP)

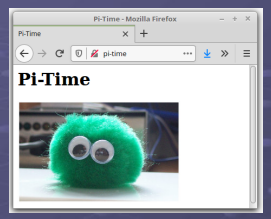
Internet protocol stack



- Application layer
  - Need to consider house keeping protocols e.g. to identify and allocate network addresses, or simply operate the computer.
    - Domain Name System (DNS)
    - Dynamic Host Configuration Protocol (DHCP)
    - Network Time Protocol (NTP)
- Transport
- Network
- Link
- Physical

Host and domain names





pi1.desk85.lan

.com .org .net .int .edu .gov .mil

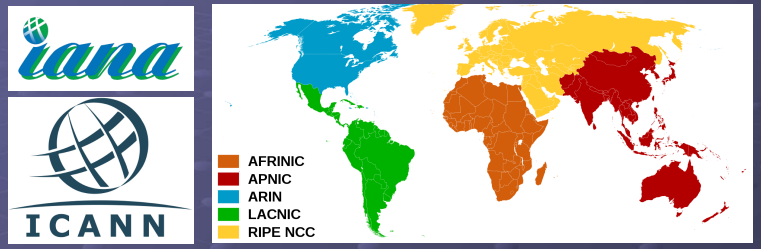
.uk .fr .be .de .ca .us .aq ...

.bank .airforce .eurovision ...

https://www.iana.org/domains/root/db

- The internet's phone box
  - How do we organise it?
    - Broken down into domains based on the location (organisational / geographical) of the computer.
    - People identify hosts by their hostnames and domains, networks use IP addresses. Need a way to map between the two.
  - One solution is to maintain a single, centralized host table e.g. /etc/hosts, mapping hostnames to IP an address. This works, but by the early 1980s this approach was becoming unmanageable.

Host and domain names

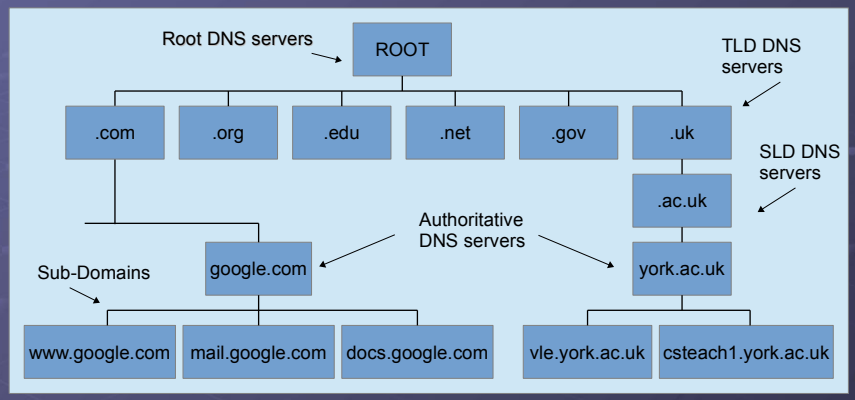


- The Internet Assigned Numbers Authority (IANA) oversees global allocation of IP and AS numbers to Regional Internet Registries (RiR)
- Also maintains the top-level domain name servers in association with Internet Corporation for Assigned Names and Numbers (ICANN)

DNS

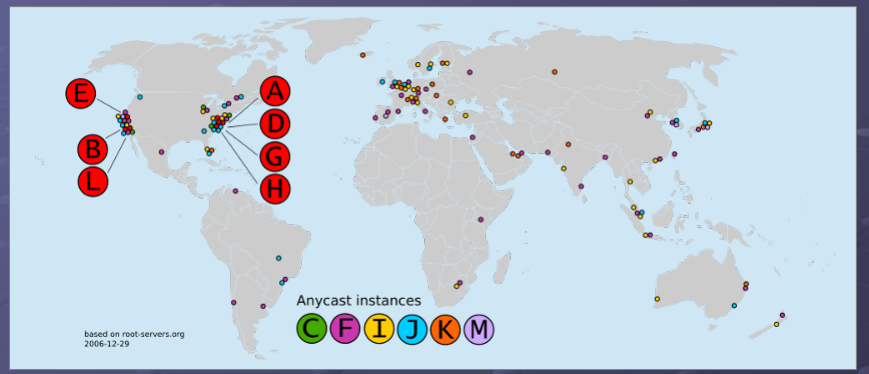
- Domain Name System Protocol
  - RFC 1035: https://tools.ietf.org/html/rfc1035
  - Domain Name System (DNS) protocol created in 1983
    - Distributed database, replicated servers
      - Improves reliability (SPF), reduces network load (quires), reduces delays (location) and processing load (updates)
  - Client-Server model, default port 53.
  - Communicates across UDP links
  - Used extensively by other application layer protocols
    - Uniform Resource Locator (URL), or web address
  - Organised as:
    - Root DNS servers
    - Top-level domain (TLD) DNS servers
    - Authoritative DNS servers
    - Local DNS server

DNS



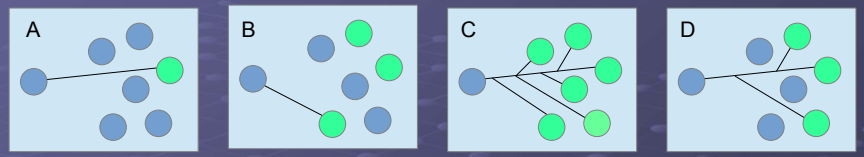
- TLD and Authoritative servers: vle.york.ac.uk
  - https://www.iana.org/domains/root/db
    - SOA: Start of authority records for specific zones

DNS



- 13 logical root name servers A – M
  - a.root-servers.net - m.root-servers.net (https://root-servers.org)
  - Have to start somewhere, top level uses “well know” ip addresses.
    - If changes need to be made you still have 12 more until updated

Quick Quizzz

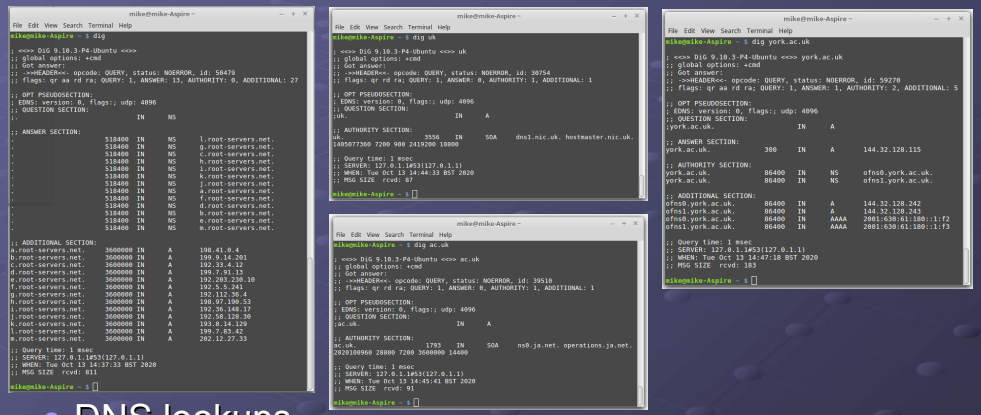


- Q : match the above pictures to connection types below.
  - Unicast : one-to-one association between hosts
  - Broadcast : one-to-all association, packet routed to all possible hosts on network
  - Multicast : one-to-many-of-many, packet routed to select hosts.
  - Anycast : one-to-one-of-many association, packet routed to any single host of a group of potential hosts.
- Q : how are connections to the root DNS servers managed? As of Aug 2024: 1865 actual root DNS servers?

DNS

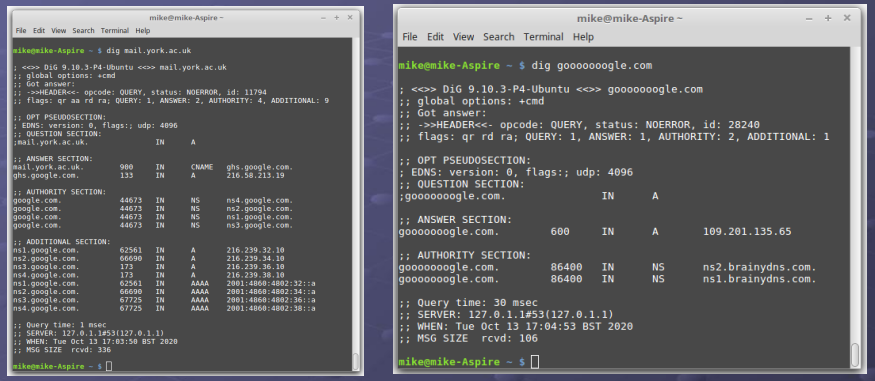
- A DNS query can contain up to 253 characters
  - A domain names consists of one or more labels
  - A label may contain 0 to 63 characters
    - Root has 0 characters
  - In theory can contain up to 127 sub-domains i.e. single letter labels, but typical will be a lot less e.g. 3 to 4.
    - [char].[char].[char].[char].[char]……..
    - 63[char].63[char].63[char].61[char]
  - Fully qualified domain names: defines all levels from top to bottom e.g. “csteach0.york.ac.uk.”
    - Trailing “.” is for the root DNS server
- Can create an alias for a domain name i.e. map to a canonical name (CNAME)
  - Multiple domains sharing the same IP address.

Demo



- DNS lookups
  - dig uk, dig ac.uk, dig york.ac.uk, dig www.york.ac.uk
  - Dept DNS servers: 144.32.128.242 & 144.32.128.243

Demo



- Q: how many OOO’s are there in Google?
  - Do not open these suspicious sites in a browser, especially if its a campus / department machine :)

Demo

reverseDNS.py

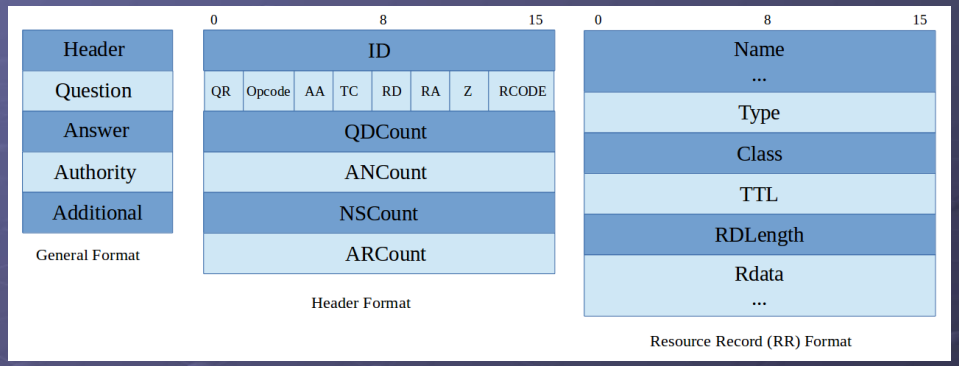
```
import os
import subprocess

BASE_IP = "144.32.50."

for i in range(1, 254):
    print( (BASE_IP + str(i) + " = " + \
    subprocess.check_output(['dig', '-x', BASE_IP+str(i), '+short']).decode("utf-8")).replace('\n','') )
```

Python 2 Spaces: 4 Ln 1, Col 1 INS

```
mike@mike-tp: ~/Documents/SYS2_2024/Slides/LEC_02/Demo
File Edit View Search Terminal Help
mike@mike-tp:~/Documents/SYS2_2024/Slides/LEC_02/Demo$ python3 reverseDNS.py
144.32.50.1 = csusb3dongle001.cs.york.ac.uk.
144.32.50.2 = lapmjf5.cs.york.ac.uk.
144.32.50.3 = prj006.cs.york.ac.uk.
144.32.50.4 = wv8586.cs.york.ac.uk.
144.32.50.5 = to0168.cs.york.ac.uk.
144.32.50.6 = pd022.cs.york.ac.uk.
144.32.50.7 = to0169.cs.york.ac.uk.
144.32.50.8 = to0171.york.ac.uk.
144.32.50.9 = wv8950.cs.york.ac.uk.
```

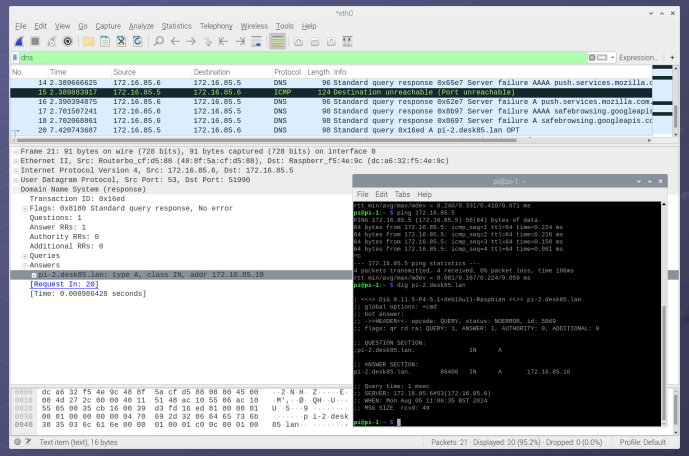
- Reverse DIG, fill in those holes :)

DNS



- DNS Query / reply packet format
  - Question : Name, Type and Class
  - Answer, Authority and Additions : use RR format

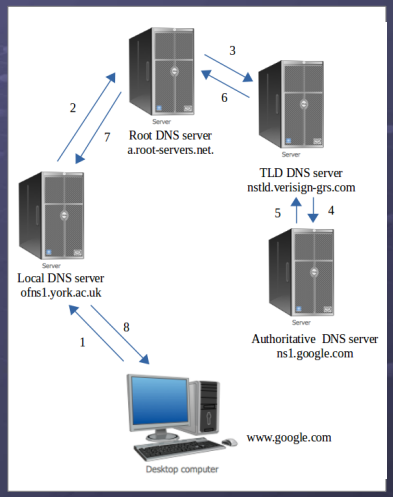
Demo



- DNS on Pi system
  - No Internet connection, but we do have a Pi-hole

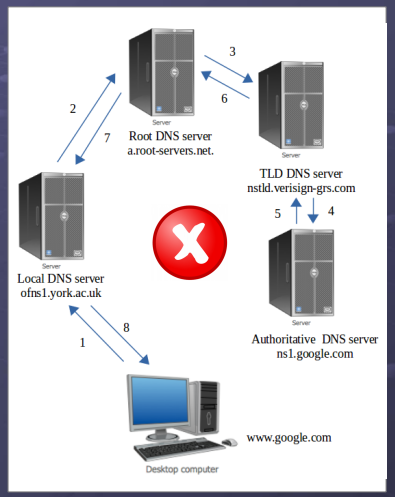
DNS

- Iterative or recursive query
  - Flag in header
- Round-robin DNS
  - Responses contain a list of potential servers (IP) that host identical services
    - Load balancing
- To minimise communication overheads previous DNS lookups are cached in the local DNS server
  - Time To Live (TTL)



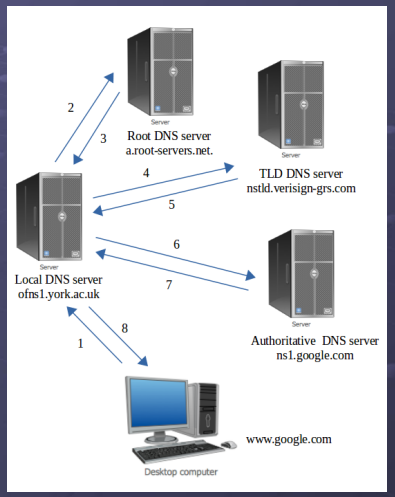
DNS

- Iterative or recursive query
  - Flag in header
- Round-robin DNS
  - Responses contain a list of potential servers (IP) that host identical services
    - Load balancing
- To minimise communication overheads previous DNS lookups are cached in the local DNS server
  - Time To Live (TTL)



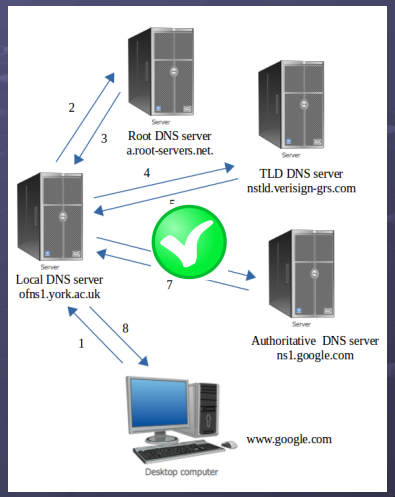
DNS

- Iterative or recursive query
  - Flag in header
- Round-robin DNS
  - Responses contain a list of potential servers (IP) that host identical services
    - Load balancing
- To minimise communication overheads previous DNS lookups are cached in the local DNS server
  - Time To Live (TTL)



DNS

- Iterative or recursive query
  - Flag in header
- Round-robin DNS
  - Responses contain a list of potential servers (IP) that host identical services
    - Load balancing
- To minimise communication overheads previous DNS lookups are cached in the local DNS server
  - Time To Live (TTL)



Pause to consider ...

- We are starting to get a picture of how our network is working
  - Application layer protocols for transferring data
    - HTTP, FTP, SMTP …
  - To help organise and identify machines on our network we can use domain names. My office test machine
    - csestudentdev01.cs.york.ac.uk
  - We can translate domain names to an IP address using a DNS query (or reverse dig -x), but to do any of this our machine must first be connected to the network.
- Q : how do we get an IP address from the network when we don't have an IP address.
  - A chicken and egg situation :)

How to get a network address

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
    address 192.168.0.254
```

- One solution is to define this manually, on the Pi this is in: /etc/network/interfaces (file can vary). For each network interface controller (NIC) present we can assign it a static IP address.
  - Advantage : simple for small networks that do not change.
  - Disadvantage : becomes difficult to manage, especially if machines leave and then return to a network.

DHCP

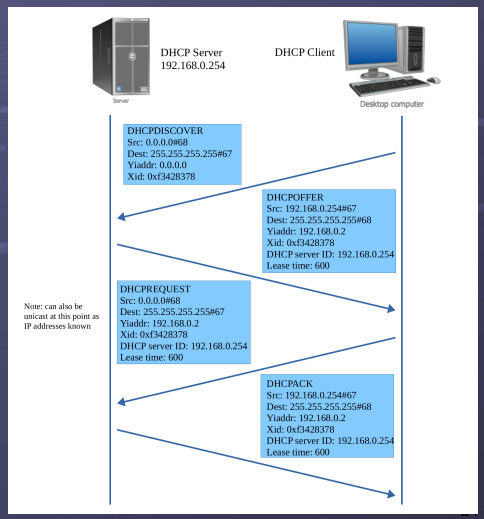
- Dynamic Host Configuration Protocol
  - RFC 1541: https://tools.ietf.org/html/rfc1541
  - DHCP created in 1993 (BOOTP 1985, RFC 951)
  - Client-Server model, default port 67 (server) and 68 (client)
  - Connectionless communicates across UDP links
    - Broadcast IP address 255.255.255.255 port 67
    - Host IP uses “unknown” address 0.0.0.0 port 68
      - Special local network and broadcast address, non-routable
  - IP address leased from server for specific time.
    - 600 sec (10 min), 7200 sec (2 hours), 86400 sec (one day)
    - Client tries to renew when 50% of lease elapses
  - If DHCP fails client assigned an IP address 169.254.0.0/16
    - RFC 5735: https://tools.ietf.org/html/rfc5735

DHCP

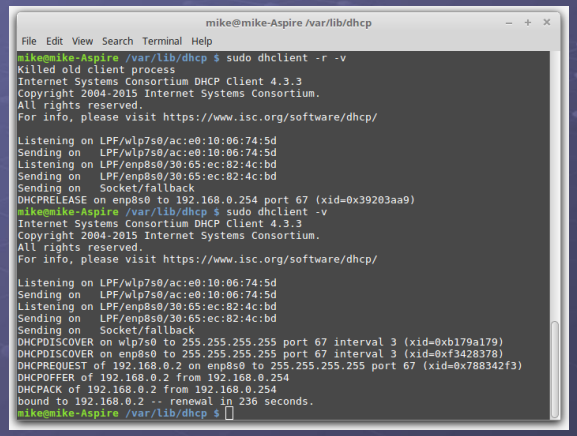
- DHCP packet format
  - Based on the Bootstrap protocol (BOOTP)
- Fields
  - Op: 1=request, 2=reply
  - Xid: transaction id (random) used to link request / reply
  - Ciaddr: client address (requested)
  - Yiaddr: current address
  - Options: DHCP phase, DNS server address and other network information (varies with phase)

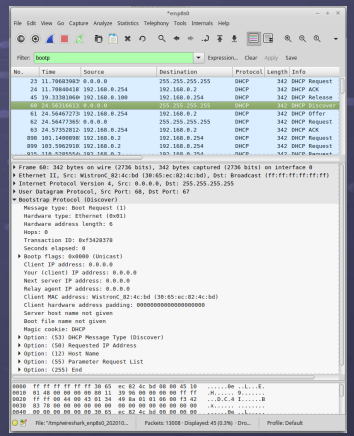
DHCP

- DHCP phases
  - May vary a little from shown if client requests a specific IP address etc.
  - Assigned IP addr can be dynamic, automatic or manually specified in the server config.



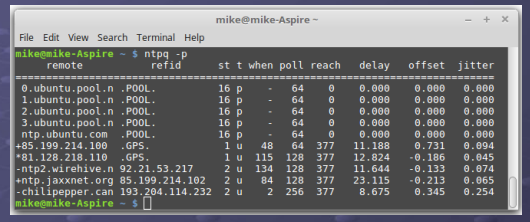
Demo





- DHCP phases : discovery, request, offer and acknowledge

NTP



- Network Time Protocol
  - RFC 958 (5905): https://tools.ietf.org/html/rfc958
  - NTP created in 1985
  - Client-Server model, default port 123
  - Connectionless communicates across UDP links
  - Hierarchy of reference clocks and servers i.e. stratum levels
    - Typically millisecond accuracy achievable

Go and research

- Have a look at how the NTP functions
  - Can't just send time, need to compensate for network delays.
  - How is time represented i.e. what format, data types are used in the protocol?
  - How many stratum levels are there?
    - What are stratum level 0s?
  - Using the ntp query command: ntpq -p what are the fields: st, poll, reach, delay offset, jitter etc?

Summary

- In addition to the headline protocols we need some glue to "bind" things together :)
- Note, we have only had a quick look at these protocols, there is a lot more detail behind the scenes e.g. how the distributed DNS database is maintained / updated etc.
- However, some unanswered questions
  - What are the TCP and UDP?
  - Why do we need two different protocols?
  - What is happening in the lower layers of the protocol stack?