University of Glasgow | School of Computing Science

Networks & Operating Systems Essentials

Dr Angelos Marnerides

<angelos.marnerides@glasgow.ac.uk>

School of Computing Science, Room: S122

Based on slides © 2017 Colin Perkins

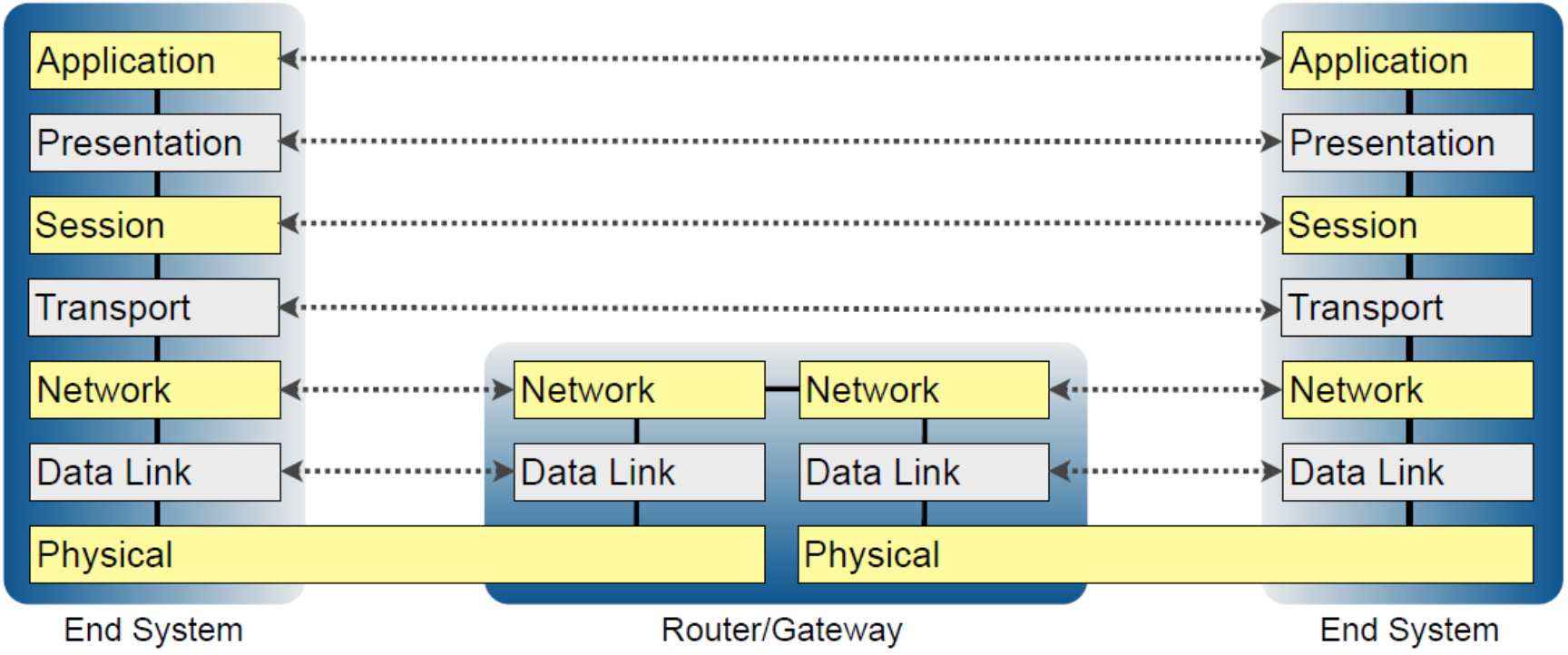
NETWORK ADDRESS TRANSLATION

Network Address Translation

Source: https://xkcd.com/742/

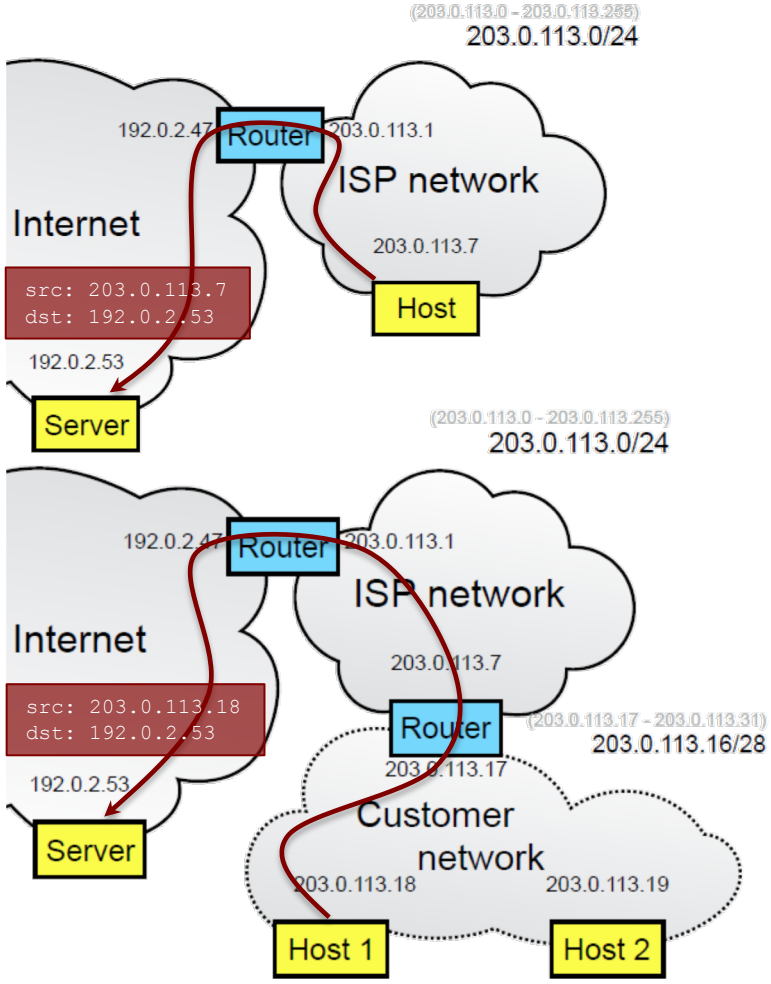
OSI Reference Model

- A standard way of thinking about layered protocol design
- A design tool; real implementations are more complex



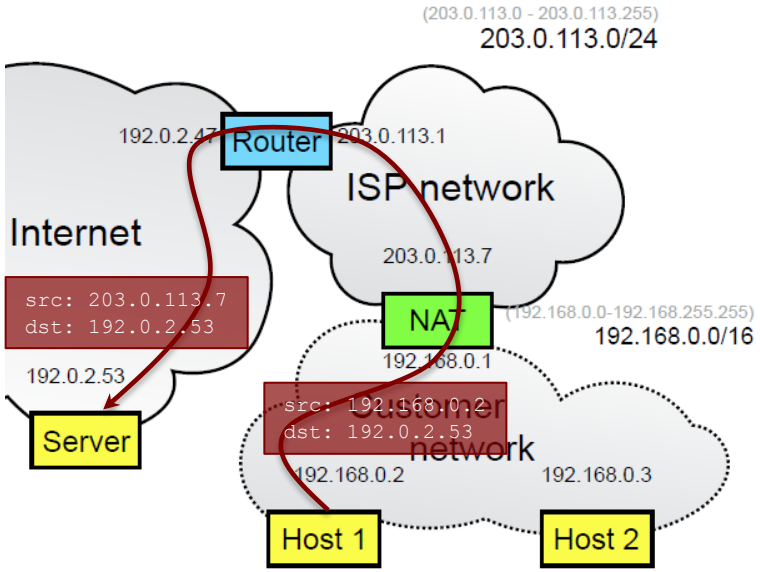
Network Address Translation

- Suppose an Internet service provider owns an IP address prefix
  - They assign a customer a single address for a single host
    - No address translation necessary
  - The customer buys another host
    - How does it connect?
  - What's supposed to occur:
    - Customer acquires a router, which gets the customer's previous IP address
    - ISP assigns new range of IP addresses to customer (from the ISP's prefix)
    - Customer gives each host an address from that new range
    - No address translation necessary



Network Address Translation

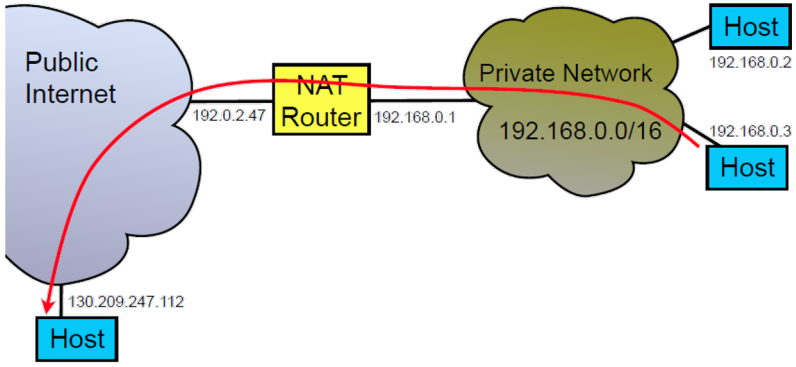
- What actually happens…
  - Customer acquires a NAT box, which gets the customer's previous IP address
  - Customer gives each host a private address
  - NAT performs address translation
    - Rewrites packet headers to match its external IP address
    - Likely also rewrites the TCP/UDP port number



- The NAT hides a private network behind a single public IP address
  - Private IP network addresses:
    - 10.0.0.0/8 (Class A - Host addresses in the range: 10.0.0.1 - 10.255.255.254)
    - 172.16.0.0/12 (Class B - Host addresses in the range: 172.16.0.1 - 172.31.255.254)
    - 192.168.0.0/16 (Class C - Host addresses in the range: 192.168.0.1 - 192.168.255.254)
- Gives the illusion of more address space

NAT vs Internet Transport Protocols

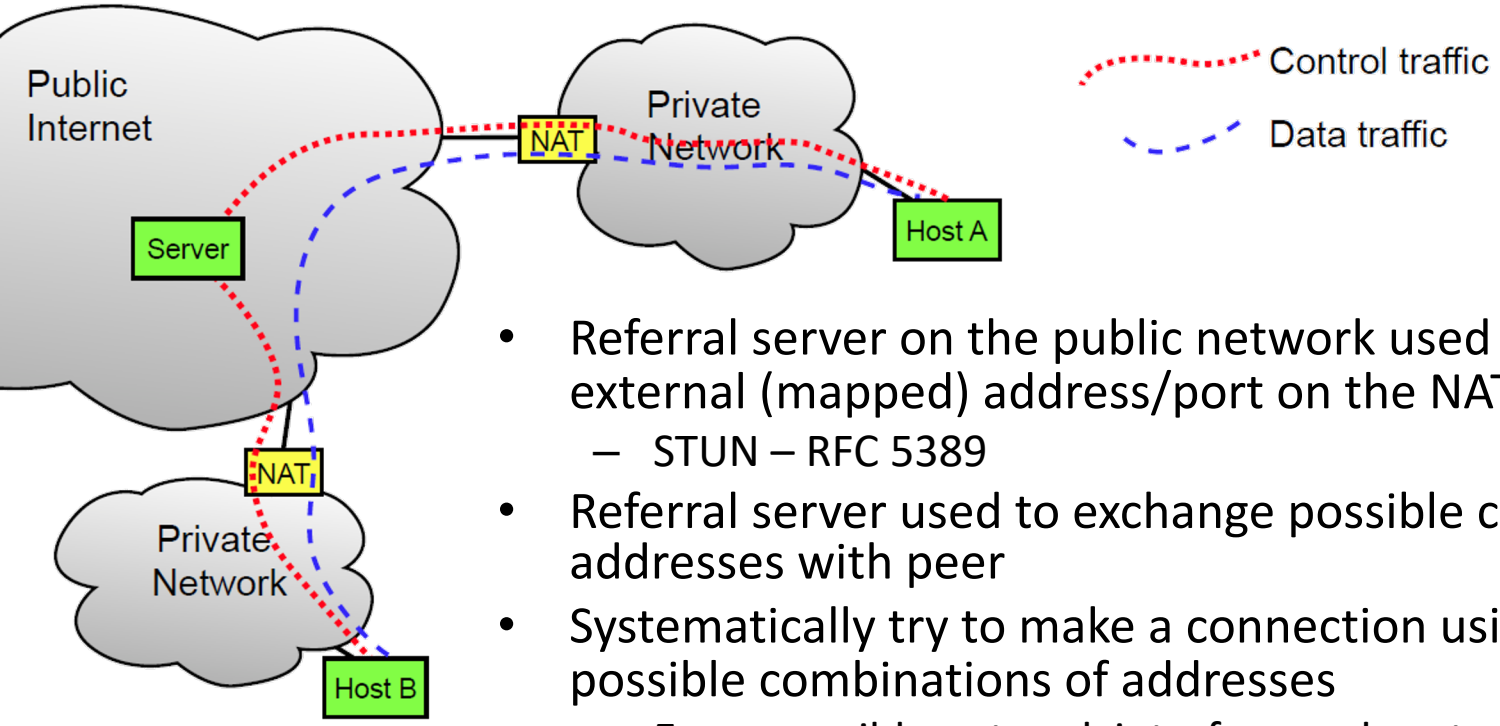
- NAT vs TCP:
  - Outgoing connection creates state in NAT
    - Need to send data periodically
      - Else NAT state times out
    - Recommended time out interval is 2 hours
      - Many NATs use shorter [RFC5382]
  - No state for incoming connections
    - NAT can't know where to forward incoming connections, without manual configuration (aka port forwarding)
    - Affects servers behind a NAT, or peer-to-peer applications



NAT vs Internet Transport Protocols

- NAT vs UDP:
  - NATs tend to have short time outs for UDP
    - Not connection-oriented, so they can't detect the end of flows
    - Recommended time out interval is not less than two minutes
      - Many NATs use shorter intervals
      - The VoIP NAT traversal standards suggest sending a keep alive message every 15 seconds [RFC4787]
  - Peer-to-peer connections easier than TCP
    - UDP NATs often more permissive about allowing incoming packets than TCP NATs
      - Many allow replies from anywhere to an open port

Extra: NAT Traversal Concepts



- Referral server on the public network used to discover external (mapped) address/port on the NAT
  - STUN – RFC 5389
- Referral server used to exchange possible connection addresses with peer
- Systematically try to make a connection using all possible combinations of addresses
  - Every possible network interface and protocol, mapped and local
  - Complex and generates significant traffic overhead
  - The ICE algorithm – RFC 5245

Network Address Translation

- Many applications fail with NAT:
  - Client-server applications with client behind NAT work without changes –web and email
  - Client-server applications with server behind NAT fail – need explicit port forwarding
  - Peer-to-peer applications fail – complex algorithm (ICE -- RFC 5245) needed to connect
- NAT provides no security benefit:
  - Most NATs also include a firewall to provide security, but NAT function gives no security or privacy benefits on its own

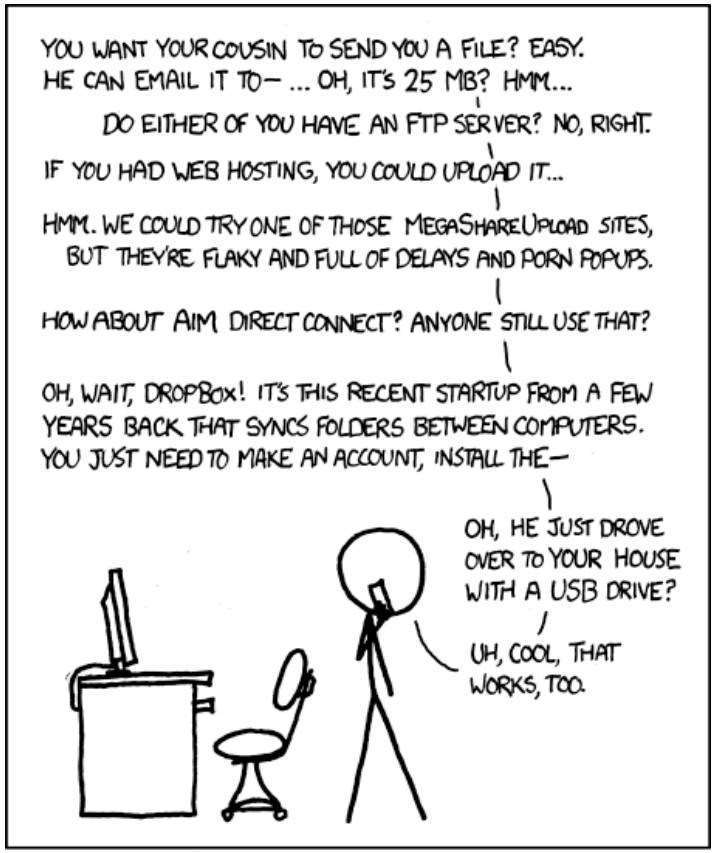
Network Address Translation

- NAT breaks many applications – so why use it?
  - Many ISPs have insufficient addresses to give customers their own prefix
  - Many customers don't want to pay their ISP for more addresses
  - ➢ Both problems due to limited IPv4 address space
    - IPv6 is designed to make addresses cheap and plentiful, to avoid these problems
  - To avoid re-numbering a network when changing to a new ISP
    - Hard-coding IP addresses, rather than DNS names, in configuration files and application is a bad idea
    - Many people do it anyway – makes changing IP addresses difficult
  - ➢ IPv6 tries to make renumbering networks easier, by providing better auto-configuration
    - Insufficient experience to know how well this works in practice
    - Some vendors also offer IPv6-to-IPv6 NAT [RFC 6296]

Based on slides © 2017 Colin Perkins

HIGHER LAYER PROTOCOLS

Higher Layer Protocols

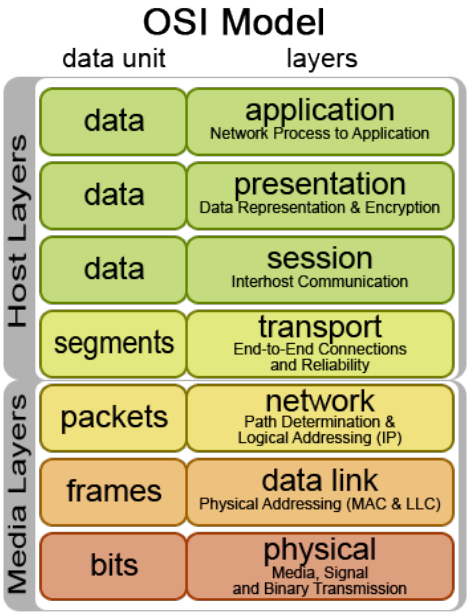


I LIKE HOW WE'VE HAD THE INTERNET FOR DECADES, YET "SENDING FILES" IS SOMETHING EARLY ADOPTERS ARE STILL FIGURING OUT HOW TO DO.

Source: https://xkcd.com/949/

Higher Layer Protocols

- The OSI reference model defines three layers above the transport:
  - Session layer, Presentation layer, Application layer
- Goal – support application needs; e.g.:
  - Setup/manage transport layer connections
  - Name and locate application-level resources
  - Negotiate data formats, and perform format conversion if needed
  - Present data in appropriate manner
  - Implement application-level semantics
- Relatively ill-defined boundaries between layers
- Typically implemented within an application or library, rather than within the kernel
  - Layering a design aid rather than a required implementation strategy

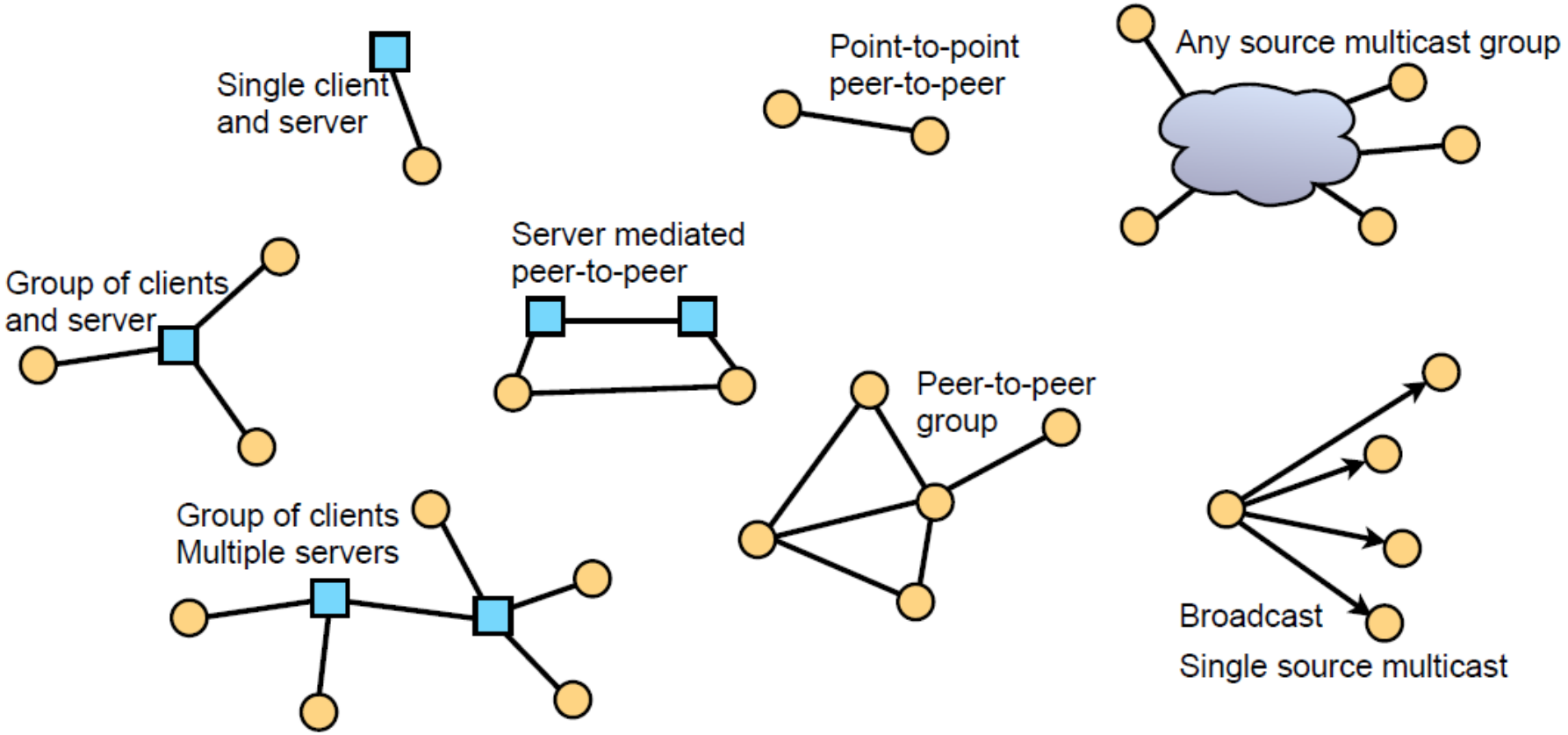


Based on slides © 2017 Colin Perkins

SESSION LAYER (L5)

The Session Layer

- What connections does the application need?



The Session Layer

- How to find participants?
  - Look-up name in a directory (e.g. DNS, web search engine)
  - Server mediated connection (e.g. instant messenger, VoIP call)
- How to setup connections?
  - Direct connection to named host (→ NAT issues)
  - Mediated service discovery, followed by peer-to-peer connection
- How does session membership change?
  - Does the group size vary greatly?
  - How rapidly do participants join and leave?
  - Are all participants aware of other group members?

The Session Layer

- IP addresses encode location → mobility breaks transport layer connections
  - Session layer must find new location, establish new connections
    - Might be redirected by the old location – e.g., HTTP redirect
    - Mobile devices may update a directory with new location
    - Complexity is pushed up from the network to the higher layers (end-to-end principle)
- A single session may span multiple transport connections
  - Session layer responsible for co-ordinating the connections
    - E.g., retrieving a web page containing images – one connection for the page, then one per image
    - E.g., a peer-to-peer file sharing application, building a distributed hash table

The Session Layer

- Some protocols rely on middleboxes or caches
  - Web cache – optimise performance, moving popular content closer to hosts
  - Email server – supports disconnected operation by holding mail until user connects
  - SIP proxy servers and instant messaging servers – locate users, respond for offline users
- The end-to-end argument applies, once again
  - Only add middleboxes when absolutely necessary

Reading Material

- Kurose and Ross, Chapter 4, section 4.4 (NAT), pp. 53-54 (Session Layer)
- Session layer reading, at all sections where the general OSI model is described; no needs to go in much further depth.

Based on slides © 2017 Colin Perkins

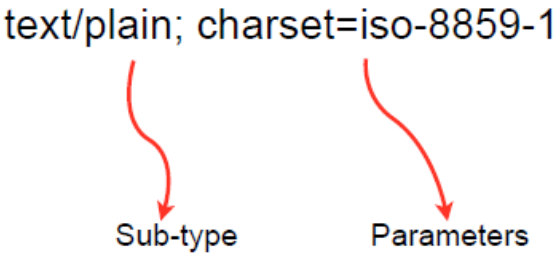
PRESENTATION LAYER (L6)

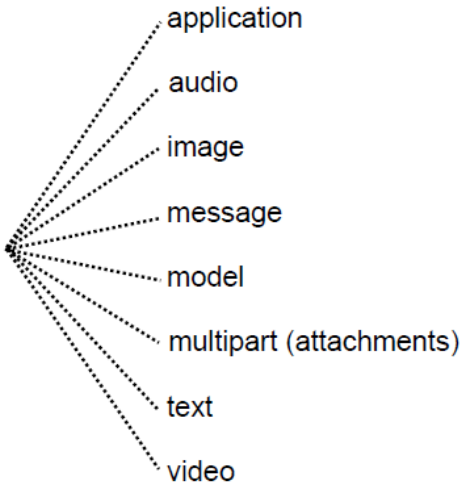
The Presentation Layer

- Managing the presentation, representation, and conversion of data:
  - Media types and content negotiation
  - Channel encoding and format conversion
  - Internationalisation, languages, and character sets
- Common services used by many applications

Media Types

- Data formats often not self-describing
  - Media types identify the format of the data
    - http://www.iana.org/assignments/media-types/
    - Categorise formats into eight top-level types
    - Each has many sub-types
    - Each sub-type may have parameters





- Media types included in protocol headers to describe format of included data

Content Negotiation

- Many protocols negotiate the media formats used
  - Ensure sender and receiver have common format both understand
- Typically some version of an offer-answer exchange
  - The *offer* lists supported formats in order of preference
  - Receiver picks highest preference format it understands, includes this in its *answer*
  - Negotiates common format in one round-trip time

Channel Encoding

- Does the protocol exchange text or binary data?
  - Text – flexible and extensible
    - High-level application layer protocols (e.g., email, web, instant messaging, …)
  - Binary – highly optimised and efficient
    - Audio and video data (e.g., JPEG, MPEG, Vorbis, …)
    - Low-level or multimedia transport protocols (e.g., TCP/IP, RTP, …)
- Recommendation: prefer extensibility, rather than performance, unless profiling shows performance is a concern
- Text-based protocols can't directly send binary data
  - Now what?

Channel Encoding

- Data must be encoded to fit the character set in use and the encoding must be signalled
  - May require negotiation of an appropriate transfer encoding, if data passing through several systems
- Issues when designing a binary coding scheme:
  - Must be backwards compatible with text-only systems
    - Some systems only support 7-bit ASCII, enforce a maximum line length, etc.
  - Must survive translation between character sets
    - Legacy systems using ASCII, national extended ASCII variants, EBCDIC, etc.
  - Must not use non-printing characters
  - Must avoid escape characters that might be interpreted by the channel (e.g., $ \ # ; & “)
    - E.g., quoted-printable encoding uses = as escape character, so that the string straße is quoted as stra=dfe (and = is represented as =3d)

Channel Encoding: Considerations

- Many protocols send binary directly, not encoded in textual format
  - E.g. TCP/IP headers, RTP, audio-visual data
- Two issues to consider:
  - Byte ordering
    - The Internet is big endian
    - Must convert from little-endian PC format
  - Word size
    - How big is an integer (e.g., 16, 32, or 64 bit)?
    - How is a floating point value represented?

Based on slides © 2017 Colin Perkins

APPLICATION LAYER (L7)

The Application Layer

- Protocol functions specific to the application logic
  - Deliver email, retrieve a web page, stream video, …
- Issues to consider:
  - What types of message are needed?
    - Highly application dependent – difficult to give general guidelines
  - How do interactions occur?
    - Does the server announce its presence? Or does it wait for the client to start?
    - Is there an explicit request for every response? Can the server send unsolicited data?
    - Is there a lot of chatter, or does the communication complete within a single round?
  - How are errors reported?
    - Many applications settled on a three digit numeric code
      - First digit indicates response type, last two digits give specific error/response
      - Allows signalling new error types, but lets older clients give meaningful response

Error Code	Meaning
1xx	In progress
2xx	Ok
3xx	Redirect
4xx	Client error
5xx	Server error

Based on slides © 2017 Colin Perkins

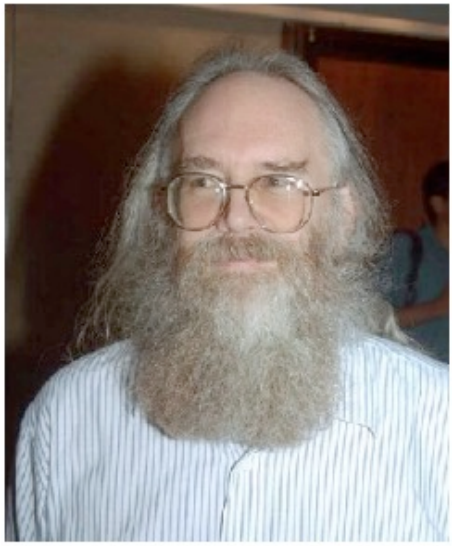
CASE STUDY: DOMAIN NAME SERVICE

Domain Name Service

- The most widely used UDP-based application is the domain name service/system (DNS)
  - The network operates entirely on IP addresses, and has no concept of names for hosts
  - The DNS is an application that translates from user-visible names to IP address
    - www.dcs.gla.ac.uk → 130.209.240.1
    - DNS is an application layer protocol, running over the network
    - Not necessary for the correct operation of the transport or network layers, or lower
  - Why run over UDP?
    - Request-response protocol, where it was thought TCP connection setup and congestion control was too much overhead
    - Unclear if this design is appropriate

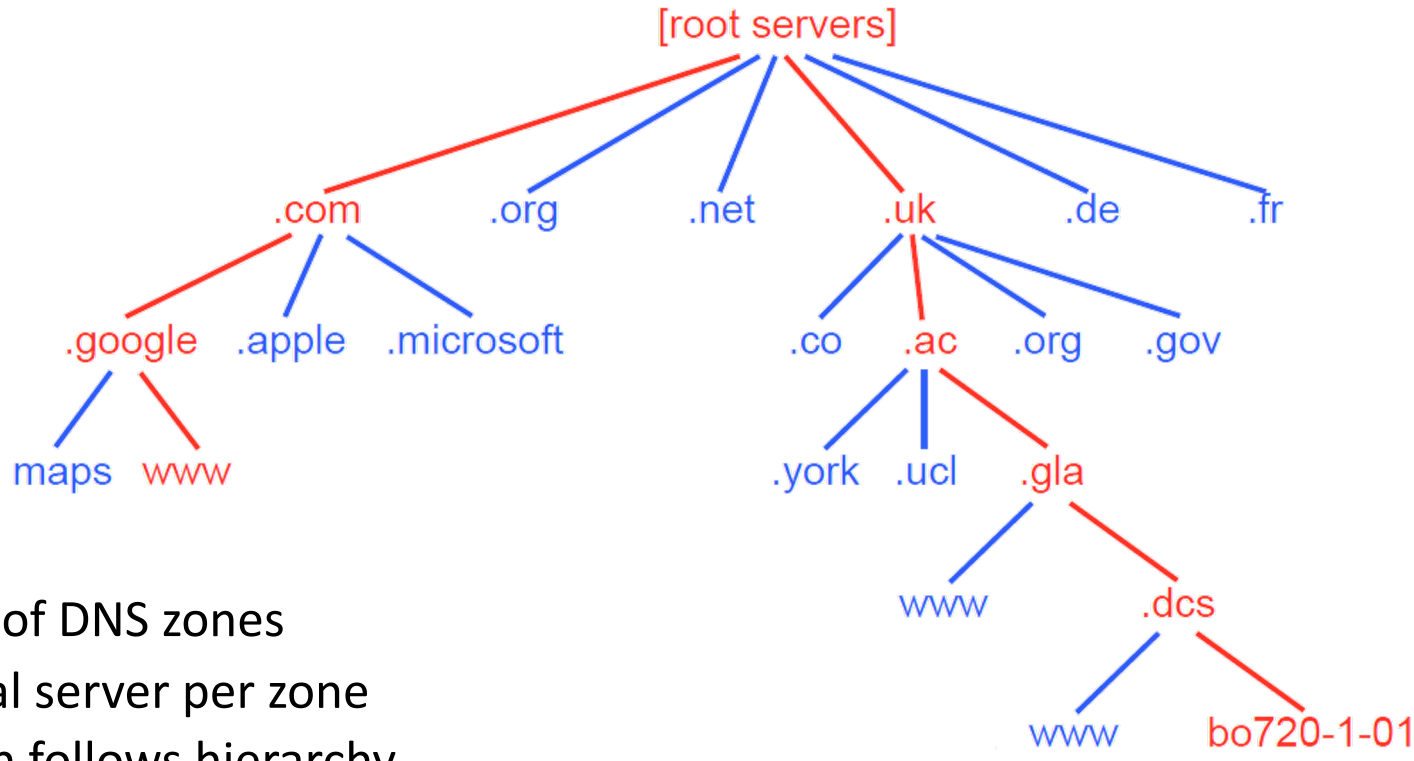
Aside: History of the DNS

- Early Internet didn't use DNS
  - Flat file hosts.txt listing all host names and addresses
  - Maintained by central domain
  - Updated by email every few days
  - Manually installed in hosts
- DNS proposed in 1983 as distributed database of host names
  - Solve scaling problems with hosts.txt
- Originally administered by IANA
  - Jon Postel *was* IANA from its creation until his death in 1998
- DNS now managed into ICANN
  - The US government ~~asserts~~ asserted ultimate control over ICANN, and hence the DNS
  - Significant attempts to move control of national domains to the UN, and hence to the countries concerned



Jon Postel

Operation of the DNS



- Hierarchy of DNS zones
- One logical server per zone
- Delegation follows hierarchy
- Hop-by-hop name look-up, follows hierarchy via root
- Results have TTL, cached at intermediate servers

Extra: Contents of a DNS Zone (SOA-Start Of Authority record)

```
$TTL 3600          ; 1 hour
example.org.       IN      SOA     ns1.example.org. admin.example.org. (
                                   2006051501      ; Serial
                                   10800           ; Refresh
                                   3600            ; Retry
                                   604800          ; Expire
                                   86400           ; Minimum TTL
                                   )
; DNS Servers
                   IN      NS      ns1.example.org.
                   IN      NS      ns2.example.org.
; MX Records
                   IN      MX 10   mx.example.org.
                   IN      MX 20   mail.example.org.
; Machine Names
ns1                IN      A       192.168.1.2
ns2                IN      A       192.168.1.3
mx                 IN      A       192.168.1.4
mail               IN      A       192.168.1.5
mail               IN      AAAA    2001:200:1000:0:25f:23ff:fe80:1234
server1            IN      A       192.168.1.10
server2            IN      A       192.168.1.11
; Aliases
www                IN      CNAME   server1
```

Source: adapted from The FreeBSD Handbook

CASE STUDY: HTTP

The HyperText Transfer Protocol

- "[…] a generic, stateless, protocol which can be used for many tasks […] through extension of its request methods, error codes and headers."

  (RFC2616)

- Basic characteristics
  - An Application Layer protocol (SSL at the Presentation Layer protocol)
  - Request-response client-server (*pull*) protocol
  - Presumes running on a reliable Transport Layer protocol (usually TCP/IP)
  - Stateless -- each request is agnostic of previous requests
  - One connection per request vs multiple requests per connection
    - Old default (HTTP/1.0): a single request/response per connection
    - New default (HTTP/1.1): multiple request/response pairs per connection
  - Typing and negotiation of data representation
    - Allows systems to be built independently of the data being transferred
  - Text-based requests
    - Binary data should be encoded, e.g., using Base64 encoding

Aside: History of the HyperText Transfer Protocol

- Originally introduced by Tim Berners-Lee in 1991
  - HTTP/0.9 -- still usable today, if no need for newer features
- Currently two versions of HTTP are in use
  - HTTP/1.0 (RFC1945 -- 1996)
    - Added MIME-like messages
  - HTTP/1.1 (RFC2616 -- 1999)
    - Added support for proxy servers, content caching, persistent connections, virtual hosts, and ranged downloads
- Maintained by W3C (World-wide Web Consortium)
  - http://www.w3.org/standards/techs/http

The HyperText Transfer Protocol

- Endpoints defined by a Uniform Resource Identifier (URI -- RFC3986)
  - Consists of 4 parts:
    1. Protocol (e.g., http, https)
    2. Hostname (IP address or DNS domain name)
    3. Port number (if omitted, the protocol's default port is used instead)
    4. Path and filename (location of requested resource)
       - Can refer to items other than files/directories, mapped through the server's configuration
       - Can also include request parameters (after a ?), named anchors (#anchor), etc.
  - Cannot contain special characters, such as blank or ~
    - Special characters are encoded in the form %XX, XX being the character's ASCII hex code (e.g., "~" => "%7E", " " (space) => "%20" or "+", etc.)
  - Examples:
    - https://www.gla.ac.uk/schools/computing/
    - mailto:info@dcs.gla.ac.uk
    - https://en.wikipedia.org/wiki/Internet#Terminology
    - https://t4.gla.ac.uk/terminalfour/SiteManager?ctfn=content&fnno=30&sid=74507

The HyperText Transfer Protocol

- Client parses URI
  - Extracts protocol, hostname, port number
  - Extracts path-and-filename
- Client constructs request message
  - Format:

    Method Request-URI HTTP-version `\r\n`
    (*Misc headers* `\r\n`)*
    `\r\n`

```
GET / HTTP/1.0\r\n\r\n
```

```
GET /index.html HTTP/1.1\r\n
Host: www.gla.ac.uk\r\n
Accept: image/gif, image/jpeg, */*\r\n
Accept-Language: en-us\r\n
Accept-Encoding: gzip, deflate\r\n
User-Agent: Mozilla/4.0\r\n
\r\n
```

- Client connects to server using a TCP/IP socket
  - Uses hostname and port number extracted earlier
    - If hostname is not an IP, must first resolve it to one
    - If no port number was defined, uses default for protocol
- Client sends request message through socket

The HyperText Transfer Protocol

- Server parses client request
  - Maps path-and-filename to local resource
- Server constructs response message
  - Format:

    HTTP-version Status-code Reason-phrase `\r\n`
    (*Misc headers* `\r\n`)*
    `\r\n`
    [*Message body*]

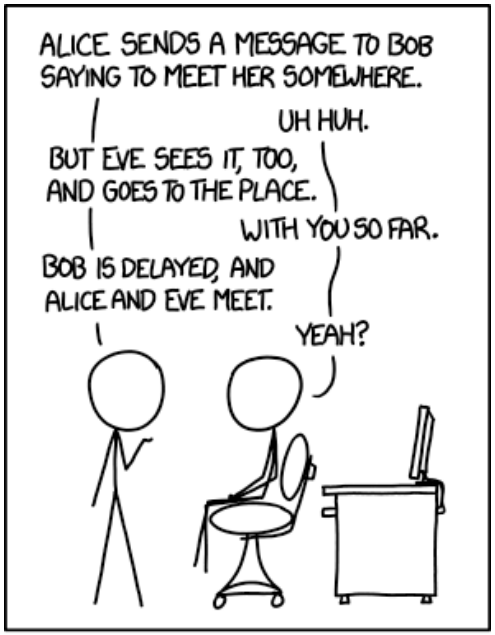
```
HTTP/1.1 200 OK\r\n
Date: Tue, 16 Oct 2018 12:00:00 GMT\r\n
Server: Apache/2.2.14\r\n
Last-Modified: Fri, 14 Sep 2018 19:46:29 GMT\r\n
Accept-Ranges: bytes\r\n
Content-Length: 44\r\n
Connection: close\r\n
Content-Type: text/html\r\n
<html><body>Hello World!</body></html>
```

- Client reads response from socket until completion (i.e., *Content-Length* bytes received after last \r\n) or until server closes connection (if keep-alive is off)
- Client parses HTML and extracts URIs
- Subsequent requests start from the beginning
  (or reuse this socket, if keep-alive is on)

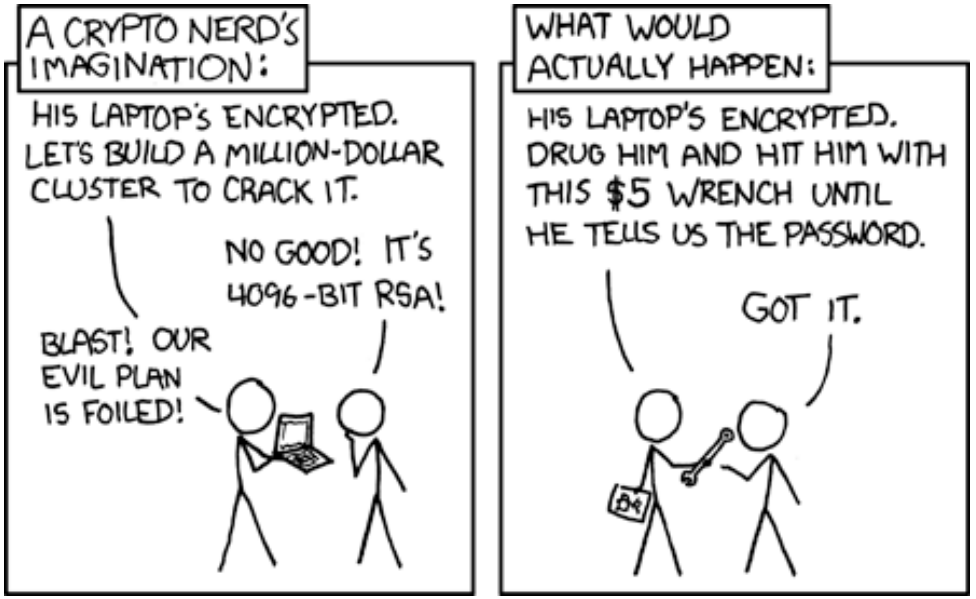
Reading Material

- Kurose & Ross: Chapter 2 , Sections 2.1, 2.2 and 2.5 . Also for Sockets you can revisit: section 2.7.
- Bonaventure : https://www.computer-networking.info/1st/html/application/dns.html (DNS)
- Bonaventure : https://www.computer-networking.info/1st/html/application/http.html (HTTP)

Coming up next…



Source: https://xkcd.com/1323/



Source: https://xkcd.com/538/