2. Introduction to Internet Applications

2.1. Representation and Transfer

- application-layer protocols specify two aspects of interaction
  - representation
  - transfer
- *representation*:
  - syntax of data items exchanged
  - specific form during transfer
  - translation of integers, characters and files between computers
- *transfer*:
  - interaction between client and server
  - message syntax and semantics
  - valid and invalid exchange
  - error handling
  - termination of interaction

2.2. Web Protocols

- World Wide Web (WWW) is one of the most widely used services on the Internet
- major WWW standards are
  - *HyperText Markup Language* (HTML): representation standard specifying contents and layout of a web page
  - *Uniform Resource Identifier* (URI): representation standard specifying format and meaning of web page identifiers
  - *HyperText Transfer Protocol* (HTTP): transfer protocol specifying how a browser interacts with a web server
- standardised by the *World Wide Web Consortium* (W3C)

2.3. Some Other Application Layer Protocols

- (usually standardised in a *Request for Comments* (RFC) by the *Internet Engineering Task Force* (IETF))
- *telnet* (for remote login)
  - defined in RFC 318 (1972)
- *ftp* (file transfer protocol)
  - defined in RFC 454 (1973)
- email protocols
  - *SMTP* (Simple Mail Transfer Protocol)
  - *POP3* (Post Office Protocol version 3)
  - *IMAP4* (Internet Mail Access Protocol)
- *DNS* (Domain Name System)
  - defined in RFC 1034 and RFC 1035 (1987)
- *RTP* (Real-time Transfer Protocol) for audio and video
  - defined in RFC 3550 (2003)

2.4. Uniform Resource Identifiers (URIs)

- a *Uniform Resource Identifier (URI)* is a unique identifier for identifying a resource on the Internet
- basic syntax is:
  `scheme ":" scheme-specific-part`
  where
  - `scheme` identifies a naming scheme, e.g., `http`
  - `scheme-specific-part` identifies resource in some way specific to the scheme
  - most commonly used URIs are *Uniform Resource Locators* (URLs)
  - URIs also include *Uniform Resource Names* (URNs) which we won't discuss further
- URIs have been generalised to *Internationalized Resource Identifiers* (IRIs) in RFC3987

2.5. Uniform Resource Locators (URLs)

- `scheme` examples include
  - `ftp`, `http`, `https`, `mailto`, `telnet`
- in the following syntax `[ ... ]` denotes *optional*
- everything else not in quotes denotes a string to be supplied
- `scheme specific part` has syntax
  `"//" [ user [ ":" password ] "@" ] host [ ":" port ] [ "/" url-path ] [ "?" query-string ] [ "#" anchor ]`
  where
  - `user` and `password` are not often used
  - `host` is a fully qualified *domain name* or IP address
  - `port` is optional (usually a default)
  - `url-path` is the path to the resource, specific to scheme
  - `query-string` includes parameters associated with the request (usually form fields)
  - `anchor` is a reference to a part of a resource (a fragment identifier)

2.6. URL example

In `https://www.bbc.co.uk/search?q=brexit`

- `https` is the scheme
- `www.bbc.co.uk` is the host
- `search` is the url-path
- `q=brexit` is the query string

2.7. URL schemes

- `http`
  - `user` name and `password` usually not applicable
  - default `port` number is 80
- `https`
  - HTTP encrypted by *Transport Layer Security (TLS)* (or previously *Secure Sockets Layer (SSL)*)
  - default `port` number is 443
- `ftp`
  - `user` name and `password` can be given
  - if not, anonymous ftp used
  - default `port` number is 21
- `telnet`
  - host is mandatory
  - default `port` number is 23
- `mailto`
  - no need for `url-path` to be specified
  - program should prompt user for message, then send using SMTP

2.8. Escaping Special URI characters

- the space character is not allowed in URIs
- the characters `/`, `#`, `?`, e.g., have special meaning in URIs
- also `&` is used to separate parameters in a query string
- so if we need any of these as an ordinary character in a URI, we use the *escaped version*
- the escaped version is the character `%` followed by the ASCII hexadecimal value of the character
- now `%` has a special meaning too
- the escaped versions of the above special characters are as follows:

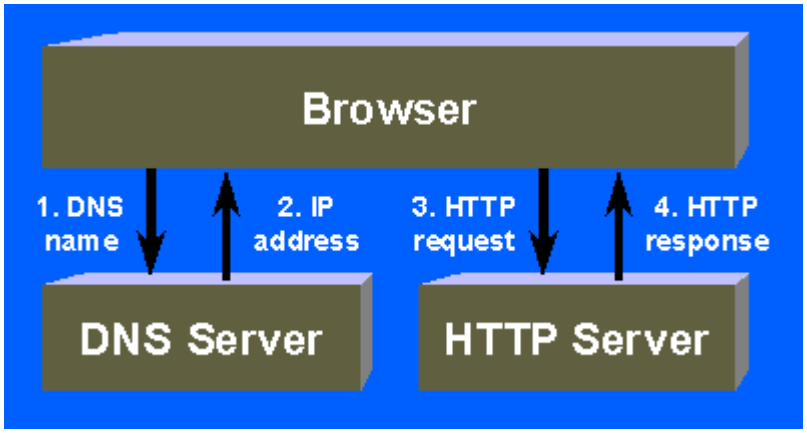
symbol	escaped version
%	`%25`
/	`%2F`
#	`%23`
?	`%3F`
space	`%20`
&	`%26`

2.9. Domain Name System (DNS)

- provides a service mapping (human-readable) DNS names to IP addresses
- browsers, mail software and most other Internet applications use DNS
- although the TCP/IP protocols themselves use only IP addresses
- DNS has two advantages:
  - easier to remember `www.w3.org` than `128.30.52.37`
  - higher level of abstraction allows simpler reorganisation
- names are organised *hierarchically*:
  - most significant part of the name on the right (specified by DNS)
  - left-most segment of a name is the name of an individual computer
- DNS is essentially
  - a distributed database implemented as a hierarchy of DNS servers
  - an application-layer protocol allowing hosts to query the database

2.10. Name Resolution

- translation of a domain name into an address is called *name resolution*
- the name is said to be *resolved* to an address
- software to perform the translation is known as a *name resolver* (or simply *resolver*)
- this software is usually built in to the application
- a resolver uses the DNS protocol to contact a DNS server on port 53
- e.g., browser uses a DNS server to map DNS name to IP address as follows:



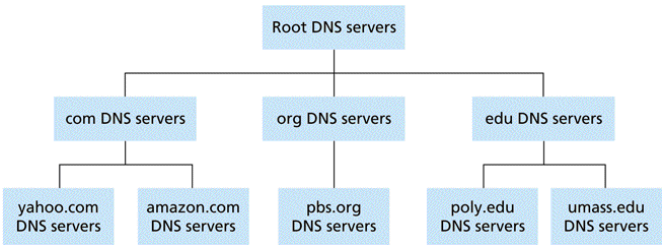
2.11. DNS Design

- why is DNS distributed?
- a simpler design would have been to have one DNS server storing all the mappings
- problems with this centralised design include:
  - it is a single point of failure
  - the need to handle huge volumes of queries
  - a single server cannot be "close" to all clients
  - it would also have to handle all updates for new hosts

2.12. Top-Level Domains

- right-most domains of the hierarchy are *top-level domains*:
  - either *country-code top-level domain (ccTLD)*
  - or *generic top-level domain (gTLD)*
- ccTLD represented by two-letter country-codes from ISO 3166, e.g., `uk`, `fr`, `de`, `ch`
- gTLD given in RFC 1591; some examples:
  - `edu`: educational institutions
  - `com`: commercial entities, i.e., companies
  - `net`: network providers
  - `org`: organisations, e.g. NGOs
  - `gov`: government agencies
  - `mil`: US military
  - `int`: organisations established by international treaties

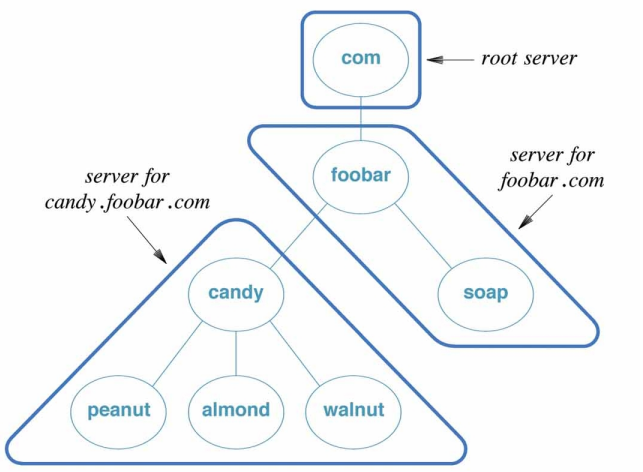
2.13. DNS Server Hierarchy



- the above figure shows a portion of the hierarchy of DNS servers
- there are 13 *root DNS servers* (each is actually a cluster of replicated servers)
    - these return IP addresses of top-level domain servers
- *top-level domain servers* are responsible for top-level domains
    - they return IP addresses of authoritative servers for organisations
- each organisation must provide an *authoritative DNS server* for its publically accessible hosts

2.14. DNS Server Model (1)

- each organisation is free to choose how to organise its servers
    - a small organisation might use an ISP to run a DNS server
    - a larger organisation might place all names on a single server
    - a large organisation might divide its names among several servers



2.15. DNS Server Model (2)

- DNS allows each organisation to
    - *assign* names to computers, or
    - *change* those names

  without informing a *central authority*
- each DNS server contains information linking it to other DNS servers up and down the hierarchy
- a given server can be *replicated*
- replication is useful for heavily used servers, such as *root servers*

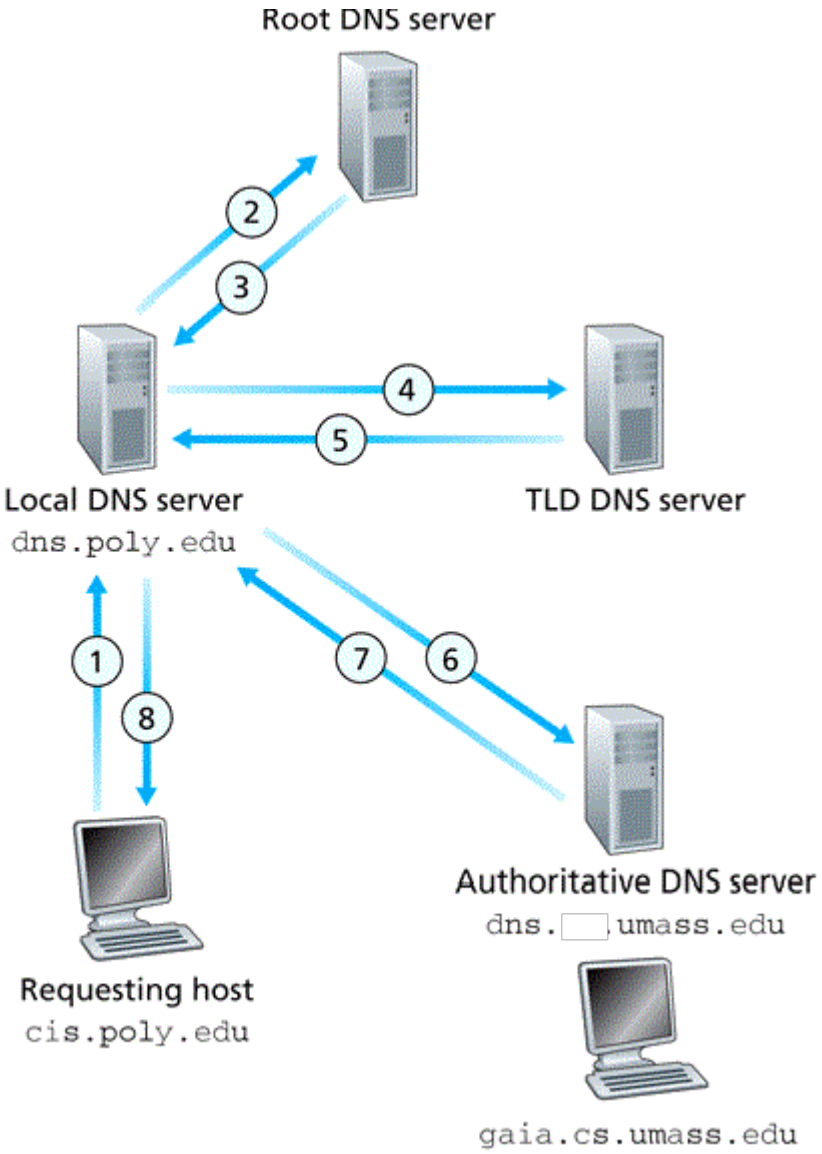
2.16. DNS Caching

- DNS servers employ *caching* in order to improve performance and reduce load
- mappings between names and addresses can be cached
- the length of time a mapping stays in the cache is given by its *time to live* (TTL)
- a mapping coming from the authoritative DNS server for a name is called an *authoritative* answer
- a mapping coming from the cache of some DNS server is called a *non-authoritative* answer
- e.g., one can use `nslookup` on Windows/Unix-based systems

```
nslookup www.dcs.bbk.ac.uk
...
Non-authoritative answer:
Name:    www.dcs.bbk.ac.uk
Address: 193.61.29.21
```

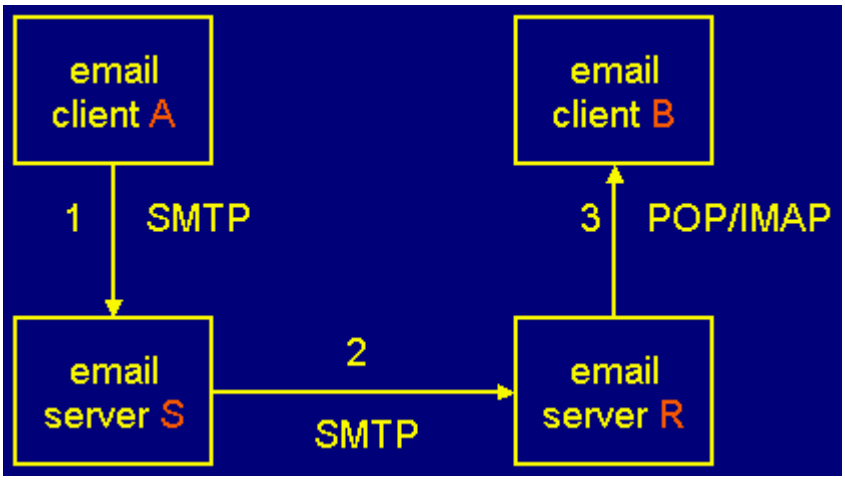
2.17. Full Name Resolution

- when nothing is cached, the local name server might have to perform full name resolution



2.18. Internet e-mail

- e-mail client responsible for
  - retrieving mail from server (POP3, IMAP4)
  - sending mail to server (SMTP)
- e-mail server responsible for
  - collecting mail from client (SMTP)
  - distributing mail to client (POP3, IMAP4)
  - relaying mail between e-mail servers (SMTP)



2.19. Sending e-mail

- *SMTP* (Simple Mail Transfer Protocol)
- defined in RFC 821 and 822 (1982), superseded by RFC 2822 (2001)
- use `mailto:` prefix in URI in browser
- uses TCP port 25
- address of recipient is of the form `name@dept.inst.ac.uk`
- uses *DNS* (Domain Name System) to map domain name to IP address

2.20. Example SMTP Session

- mail message is transferred from user `John_Q_Smith` on computer `example.edu` to two users on computer `somewhere.com`

```
Server:   220 somewhere.com Simple Mail Transfer Service Ready

Client:   HELO example.edu
Server::  250 OK

Client:   MAIL FROM:<John_Q_Smith@example.edu>
Server:   250 OK

Client:   RCPT TO:<Mathew_Doe@somewhere.com>
Server:   550 No such user here

Client:   RCPT TO:<Paul_Jones@somewhere.com>
Server:   250 OK

Client:   DATA
Server:   354 Start mail input; end with <CR><LF>.<CR><LF>
Client:   ...sends body of mail message, which can contain
Client:   ...arbitrarily many lines of text
Client:   <CR><LF>.<CR><LF>
Server:   250 OK

Client:   QUIT
Server:   221 somewhere.com closing transmission channel
```

2.21. Email Representation Standards

- two important standards exist
  - RFC (Request For Comments) 2822 mail message format
  - Multi-purpose Internet Mail Extensions (MIME)
- RFC 2822 format comprises
  - a header section
  - a blank line
  - and a body
- header lines each have the form

  ```
  keyword: information
  ```

  where keywords include `From`, `To`, `Subject`, `Cc`
- the mail message (including headers) makes up the `DATA` as sent by SMTP

2.22. Multi-purpose Internet Mail Extensions (MIME)

- SMTP originally only used the 7-bit ASCII format
- inadequate for non-English and non-textual data
- *MIME* was defined in RFCs 2045, 2046, 2047, 2048, 2049; allows
  - non-ASCII message bodies
  - extensible set of different formats for non-textual bodies
  - multi-part message bodies
  - non-ASCII textual header information

2.23. MIME Headers

- MIME headers include:
  - `MIME-Version`
  - `Content-Type`: specifies a type and subtype
  - `Content-Transfer-Encoding`: specifies auxiliary encoding for transfer
- contents of the `Content-Type` header is the *MIME type*
- examples of MIME types are `text/html`, `image/gif` and `multipart/mixed`
- example of `Content-Transfer-Encoding` is `base64`:
  - preferred encoding for 8-bit binary data
  - each group of 3 bytes (24 bits) is encoded as 4 ASCII characters

2.24. Base64 Encoding

	0x00	**0x10**	**0x20**	**0x30**
0	A	Q	g	w
1	B	R	h	x
2	C	S	i	y
3	D	T	j	z
4	E	U	k	0
5	F	V	l	1
6	G	W	m	2
7	H	X	n	3
8	I	Y	o	4
9	J	Z	p	5
A	K	a	q	6
B	L	b	r	7
C	M	c	s	8
D	N	d	t	9
E	O	e	u	+
F	P	f	v	/

- table on left is used in base64 encoding
- values in top row and leftmost column are hexadecimal numbers
- range of values is `0x00` to `0x3F` (`111111`), i.e., `0` to `63`
- example: encode `01011010, 10001010, 00011101` as follows
  1. splitting into 4 6-bit values:
     `010110, 101000, 101000, 011101`
  2. converting to hex: `0x16, 0x28, 0x28, 0x1D`
  3. use table to encode: `W, o, o, d`

2.25. Links to more information

- The RFC homepage
- FAQs on DNS From Internic, a registry of domain name registrars
- Internet Corporation for Assigned Names and Numbers (ICANN): organisation in overall charge of the DNS
- Wikipedia articles on DNS, SMTP, IMAP, POP3, MIME

Chapter 4 of [Comer] and Chapter 2 of [Kurose and Ross].