Week 11

HTTP + DNS

School of Information Technology and Electrical Engineering
The University of Queensland

CSSE 2310 7321

Outline

Today

- HTTP
- DNS

Credits:

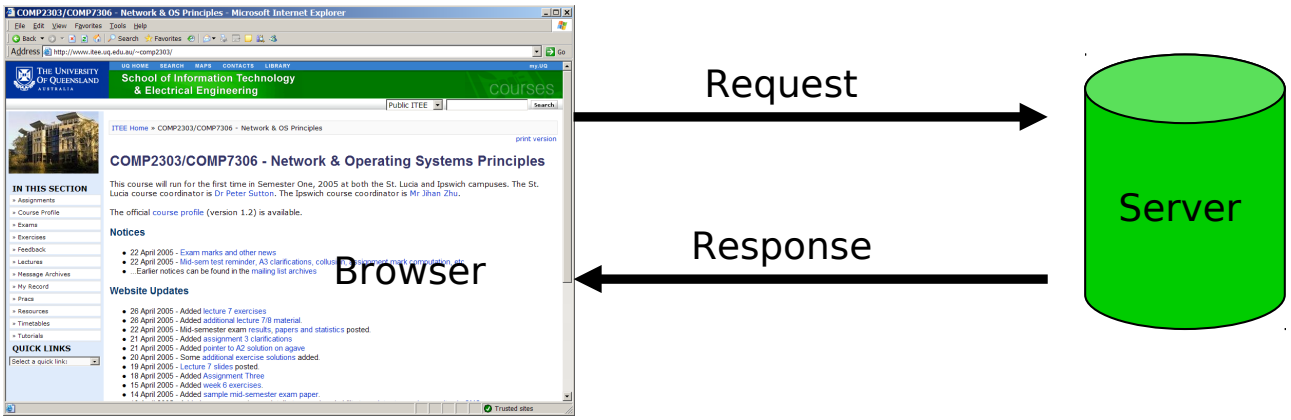
- Tanenbaum, “Computer Networks”

HTTP

- **HTTP = HyperText Transfer Protocol**
  - Not the same thing as HTML
  - Can transport other things
  - ASCII based protocol (not the body)
    - i.e. lines of text – each terminated by carriage-return newline (\r\n)
      - Most ASCII based network protocols use \r\n line endings
  - Usually on top of TCP, but standard does not require this

How does HTTP work?

- HTTP has two phases
  - Request - browser sends request
  - Response - server sends response



- ASCII based protocol
  - Can simulate this with telnet connection

HTTP Requests and Responses

- Requests and Responses both have three parts
  1. **request or response line**
  2. **header section**
  3. **body**
- HTTP Request:
  - Client (browser) first connects to the server
    - Usually on TCP port 80

HTTP Request

HTTP Request Line
Header Section
Body

- After connection, client sends a request line, consisting of
  - method
  - document address
  - HTTP version no.
- Example:
  - **`GET /~comp2303/index.html HTTP/1.0`**

Method

Document Address

HTTP Version

Example

- `telnet www.uq.edu.au 80`
  `GET / HTTP/1.0`
  (followed by two newlines)
- `echo –e "GET /~comp2303/ HTTP/1.0\r\n\r\n" | nc itee.uq.edu.au 80`

(The –e interprets escaped chars)

HTTP Request Methods:

- **GET** - tells server to retrieve entire document
- **HEAD** - requests just the header information for the response (not HTML document <HEAD>)
- **POST** - for posting data (e.g. from forms)
  - Contents of form encoded and sent to server
- **PUT** - for replacing document with data sent from the client
- **DELETE** - for removing a document from server
  - **PUT** and **DELETE** used for direct-to-web publication
- **LINK**, **UNLINK**, **OPTIONS**, **TRACE**, **...**

HTTP Request

- After request line, client can send any number of header lines
  - Mostly informational
  - Usually optional

Example:

```
Connection: Keep-Alive
Accept: image/gif, image/jpeg, */*
Accept-Encoding: gzip
Accept-Language: en
User-Agent: Mozilla/4.5 [en] (Win98; I)
```


HTTP Request Headers

- **User-Agent** - name and version of client
- **Referer** - URL of last document displayed
- **Authorization** - client's authorization to access the data (e.g. encoded name & password)
- **If-Modified-Since** - return document, only if modified since given date
- **Content-Length** - Number of data bytes
  - Mandatory for **PUT** and **POST** requests
- **Connection** - Connection options (e.g. **Keep-Alive**)
- **Host** - Virtual host to retrieve data from
- **Cookie** - Cookie(s) for that URL
- plus many others ...

HTTP Request

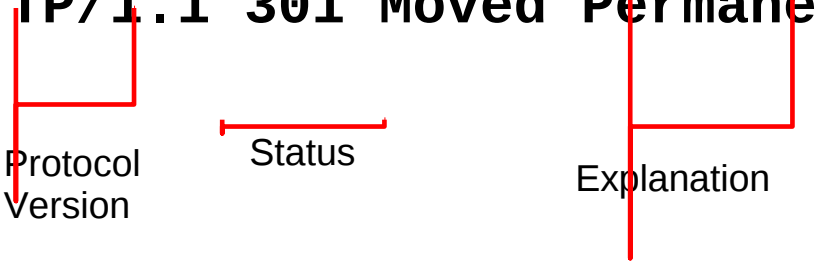
HTTP Request Line
Header Section
Body

- **Body (or Request Data)**
  - Follows the header section (and a blank line)
  - Only needed for PUT and POST requests

HTTP Response

HTTP Response Line
Header Section
Body

- First line of response consists of
  - Protocol version number
  - Status code
  - Explanation of status
- Example:
  - **HTTP/1.1 200 OK**
  - **HTTP/1.1 301 Moved Permanently**



Common Status Codes

- 100's - Informational
- 200's - Client request successful
  - 200 - OK - URL found, contents follow
  - 204 - No Response - Request OK but no data
- 300's - Redirecton, further action needed
  - 301 - Moved - URL permanently moved
  - 304 - Not Modified - Possible response to "If-Modified-Since"
- 400's - Client request incomplete
  - 401 - Unauthorised - user must produce authorisation
  - 403 - Forbidden - authorisation failed
  - 404 - Not Found - document does not exist
- 500's - Server errors

HTTP Response

HTTP Response Line
Header Section
Body

- Header lines consist of
  - Information about server
  - Information about document that follows
  - Mostly optional - except for **`Content-Type`**
- Example:

```
Date: Sat, 09 Oct 1999 11:32:15 GMT
  Server: Apache/1.3.6 (Unix)
  Last-Modified: Sat, 09 Oct 1999 11:30:06 GMT
  Content-Length: 111
  Keep-Alive: timeout=15, max=100
  Connection: Keep-Alive
  Content-Type: text/html
```

Common Response Headers

- **`Server`** - name and version of the server software
- **`Date`** - the current date
- **`Last-Modified`** - date document was last changed
- **`Location`** - new location for redirection responses
- **`Content-Length`** - No. of bytes of data
- **`Content-Type`** - MIME type of data
- **`Set-Cookie`** - contains cookie info

- Lots of others ...

HTTP Response

HTTP Response Line
Header Section
Body

- Body (or Response Data)
  - Follows the header section (and a blank line)
  - Not needed for **`HEAD`** request
  - Can be 0 bytes to billions...
- Connection
  - Usually server will close connection after data finished
  - If **`Connection: Keep-Alive`** has been specified then channel will remain open waiting for another request

HTTP - Summary

- Request
  - Method Document-Address HTTP-version
  - Request Headers
  - Body (Request Data)
- Response
  - HTTP-version Status Status-Description
  - Response Headers
  - Response Data

DNS – Domain Name System

Strings are used to name hosts

- e.g. agave.students.itee.uq.edu.au

Network only understands numbers

- Need conversion mechanism

Original ARPANET

- Hosts file – listed all hosts and IP addresses
  - Doesn't scale
  - Need to decentralize

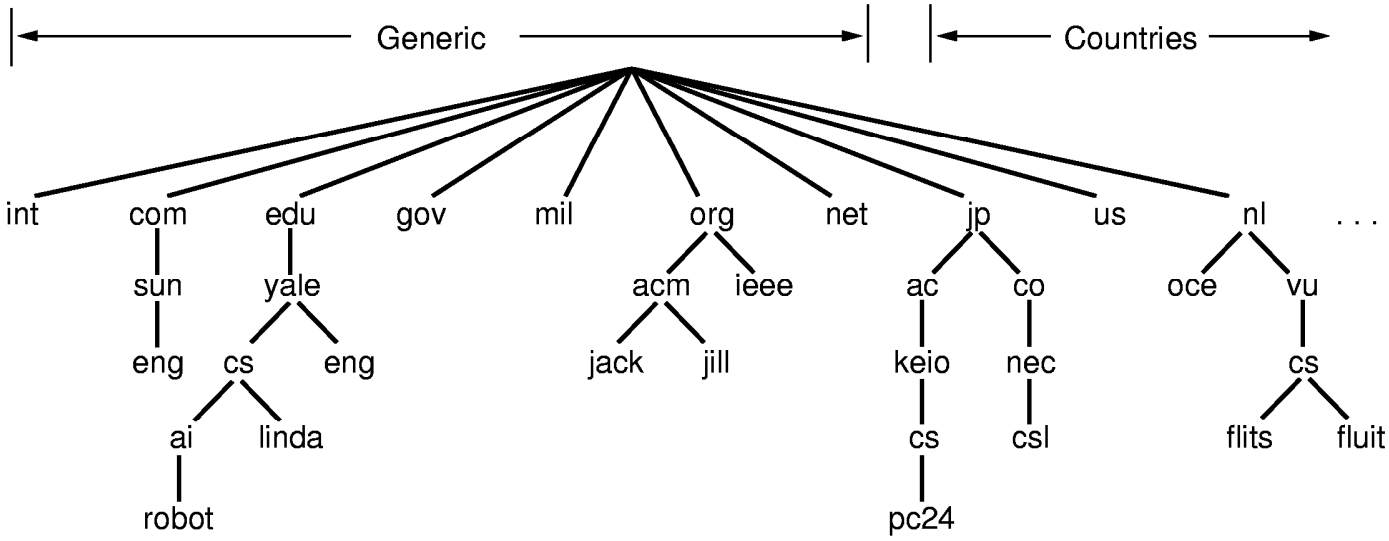
DNS Hierarchy

Subdivide and delegate authority

Repeat – end up with tree like structure

Internet – hierarchy based on structure of organisations – not on physical network connections

DNS Hierarchy (cont.)



There are additional top level domains also: .biz, .info, …

Mapping Domain Names to Addresses

Name server

- Program that supplies name-to-address translation service

Client – *name-resolver*

- Uses one or more servers

Resolution

- Start with local server
- Query passed on if answer unknown
- Answer cached locally for some time

DNS Query

DNS Queries – based on UDP

Figures to be drawn

- Recursive
- Non-recursive

nslookup demonstration