CA169: Week 10

HTTP

HTTP – HYPERTEXT TRANSFER PROTOCOL

- Invented in 1989 at CERN by Tim Berners-Lee
- Application layer protocol
- The protocol of the world wide web
- Latest RFC 7230- https://tools.ietf.org/html/rfc7230

HTTP – APPLICATION LAYER

- HTTP lives in the application layer
- Focuses on program to program communication
    - E.g. Web Server -> Google Chrome
- The underlying network architecture and protocols are abstracted

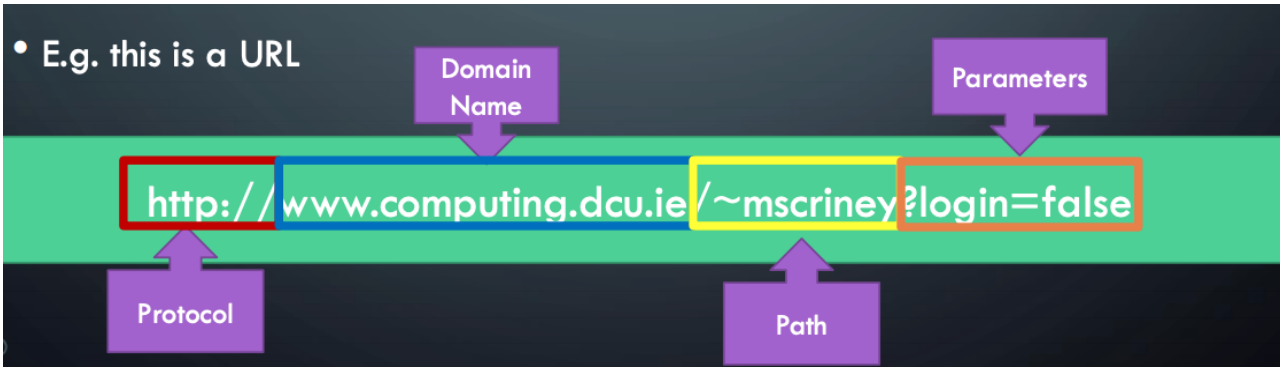
HTTP – RFC 7230

- Standards document for HTTP
- Describes
    - formats for all legal HTTP requests and responses
    - Formats of URLs
    - Controls for caching web pages
    - Persistence of connections
- Allows independent developers to develop their own web servers and clients and have them inter- operable

HTTP – URLS

- HTTP requires a URL (Uniform Resource Locator) to be passed to it
- It is more than just a domain name or IP address
- In addition to these, it specifies a protocol and a path and parameters
- E.g. this is a URL

    http://www.computing.dcu.ie/~mscriney?login=false



HTTP – OTHER URL COMPONENTS

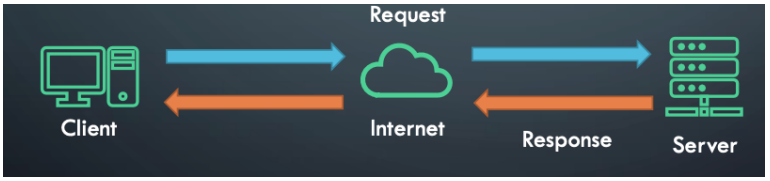
- The port – http://192.168.0.1:8000/hello.html
- User authentication – ssh://myuser:mypassword@192.168.0.1
- HTML Fragments – https://www.computing.dcu.ie/~mscriney/#teaching

HTTP - HTML

- HTTP was developed with HTML
- HTML – Hypertext Markup Language
- Hypertext – text containing links to other text documents
- We are currently on HTML5 RFC 7992 - https://tools.ietf.org/html/rfc7992

HTTP – REQUEST / RESPONSE

- HTTP works on a request/response mechanism
- The client makes a request
- The server provides the response



HTTP – REQUESTS

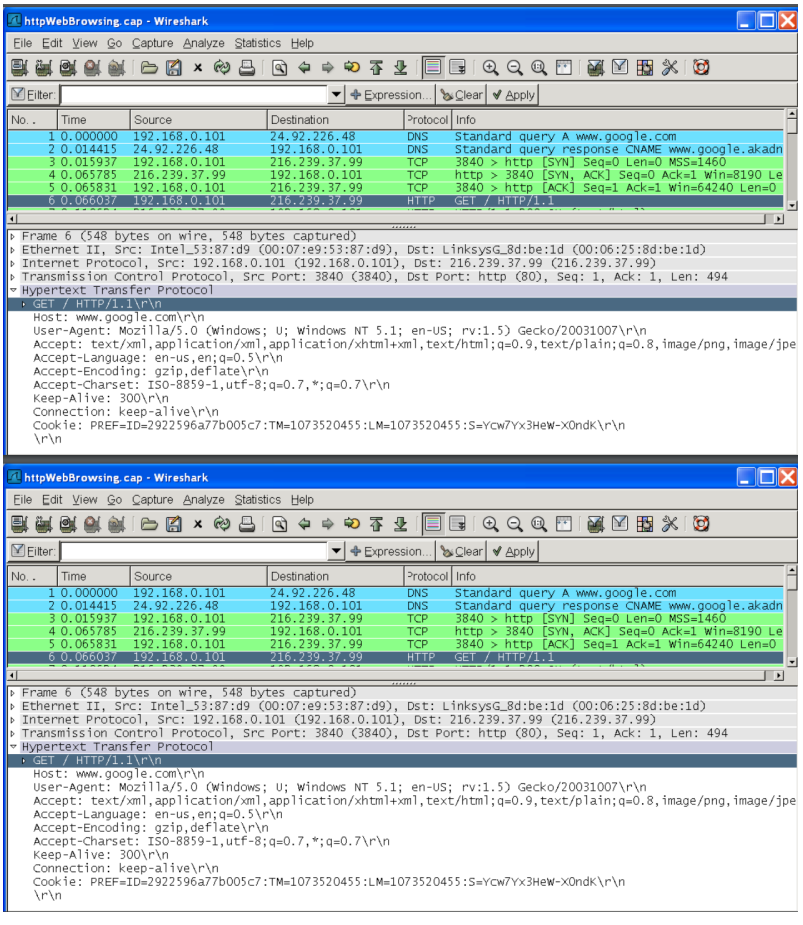
- HTTP Requests are made up of:
- The URL
- HTTP Headers
- Additional Data (e.g. username/password)

HTTP RESPONSE

- A response is made up of
  - Headers
  - Data (whatever you asked for, webpage, picture etc…)
  - A status code

HTTP REQUEST - WIRESHARK

- Open Wireshark and go to www.google.com
- See `http1.cap` on Loop
- The first 2 packets resolve the domain name (A DNS query!)
- Now a TCP connection is established, packets 3-5.
- Packet 6 sends "`GET / HTTP1.1\r\n`"



HTTP – GET REQUEST

- `"GET / HTTP1.1\r\n"`
- GET something, a file called "/" (default)
- \r is carriage return and \n is a line feed
    - Old typewriter terminology, used to separate one header from the next, each successive header has one, check it out.
- GET is one of the HTTP VERBS

HTTP HEADERS

- The name www.google.com\r\n specifies which webserver is being contacted.
- IP addressed machines may support several web-servers
- User-agent describes web browser and client machine making the request (this setup is quite old, a Mozilla browser, forerunner of Firefox and the
- Windows NT operating system

HTTP REQUEST METHODS

- GET – the most common method
    - Request a resource from a server
- POST
    - Submit data to be processed
    - Main use in web forms
- HEAD – Same as get but don't return any data

HTTP METHODS

- PUT – Update a resource with a new version
- PATCH – Update a resource (specify how it should be updated)
- DELETE – Remove a resource

HTTP HEADERS

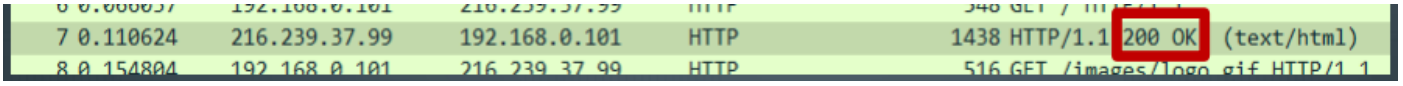
- Accept headers
- Server may support several languages, encodings, character sets, these tell server which is preferred.
- Keep alive and connection headers tell about the TCP connection being used, whether connection should be kept open and for how long.
- Most connections are persistent, allowing multiple requests from same client to server. This improves performance greatly of HTTP 1.1 over HTTP 1.0

HTTP RESPONSE

- Packet 7 the response…
- First HTTP 1.1 is fine with the server
- Headers…
- Cache-control: whether to store copies for future reference. Private here means that this is a specially generated and can be cashed by the user, but not by a group of users on a "shared proxy cache"
- Lists the types of content and encodings it can accept, text/html and gzip (compressed)
- GWS identifies itself as google's own webserver
- Content length is 1216 long and we get the date.

HTTP RESPONSE CODES

- On packet 7 we can see the response code



- A response code of 200 means a successful request
- There are many others
- The first digit specifies the class of the response
- 1xx – Information
  - 101 Switching protocols
- 2xx Success
  - 200 – OK
  - 204 – No content
  - Usually if your code makes a web request you look at the code
  - If code !=200 //Something has gone wrong
- 3xx – Redirection
  - 301 – Moved permanently (used to redirect to a new resource)
  - 304 – Not modified
- 4xx – Client error
  - 404 – not found (because of UCC!)
  - 403 – forbidden (request is ok, but you're not allowed to view the contents)
  - 401 – Unauthorized (username/password incorrect)
  - 410 – Gone – Resource unavailable now and in the future (tell search engines to remove)
- 5xx – Server Error
  - 500 - Internal server error – Generic error , something is wrong on the server
  - 501 – Not implemented – The request cannot be fulfilled
  - 503 – Service unavailable – Web server is overloaded or down for maintenance
- And the weird ones
- 418 – I am a teapot
  - https://tools.ietf.org/html/rfc2324
- 420 – Enhance your calm
  - Sent by twitter if you are being rate limited (similar to real error code 429, too many requests)
  - https://developer.twitter.com/en/docs/basics/response-codes

HTTP – MULTIPLE GET

- Only 1 GET request in packet 8
- Second request generated by the HTML source sent back for processing at the client.
  - `GET /images.logo.gif HTTP/1.1\r\n`
- It asks for the Google logo
- Several requests may be daisy-chained like this
- Multiple requests are very sophisticated now

GNU

- GNU is a famous open source and licensing organisation on the web
- From packet 21 a similar interaction can be seen sorting out where it is
- Packet 26 onwards contains the interactions with it.

WIRESHARK – FOLLOW TCP STREAM

- Wireshark allows us to follow particular interactions.
- Looking at the full interaction for Google
    - Select <Analyze> menu, and <Follow TCP Stream> from the menu. You will see every interaction between your client and the server. Each are coloured differently.
- A filter has been automatically entered for you
    - (tcp.stream eq 0)
    - This is very useful

MULTIPLE TCP STREAMS

- Packets 35, 41 and 42 open second TCP connection, same IP address as the first, same port (80), but local client port is different, 3842 instead of 3841.
- This gives rise to a second, parallel connection which speeds up transfer.

QUESTIONS

- Isolate the requests sent by the browser to the server
- Visit Google.com
    - <View ><Page Source>
    - Copy it into a file called test.html using Notepad application
    - Open a web browser and drag the file into the browser
    - What website do you see and what is missing and why?
    - Write a colour filter to highlight all of the HTTP requests in the trace
    - Write a colour filter to highlight all of the HTTP responses
    - Combine the two above, HTTP requests and responses only
- Visit three websites, one in DCU, another in Ireland and one abroad
    - compute the average response time for each. Describe how you did the calculation
    - What is the IP address of each