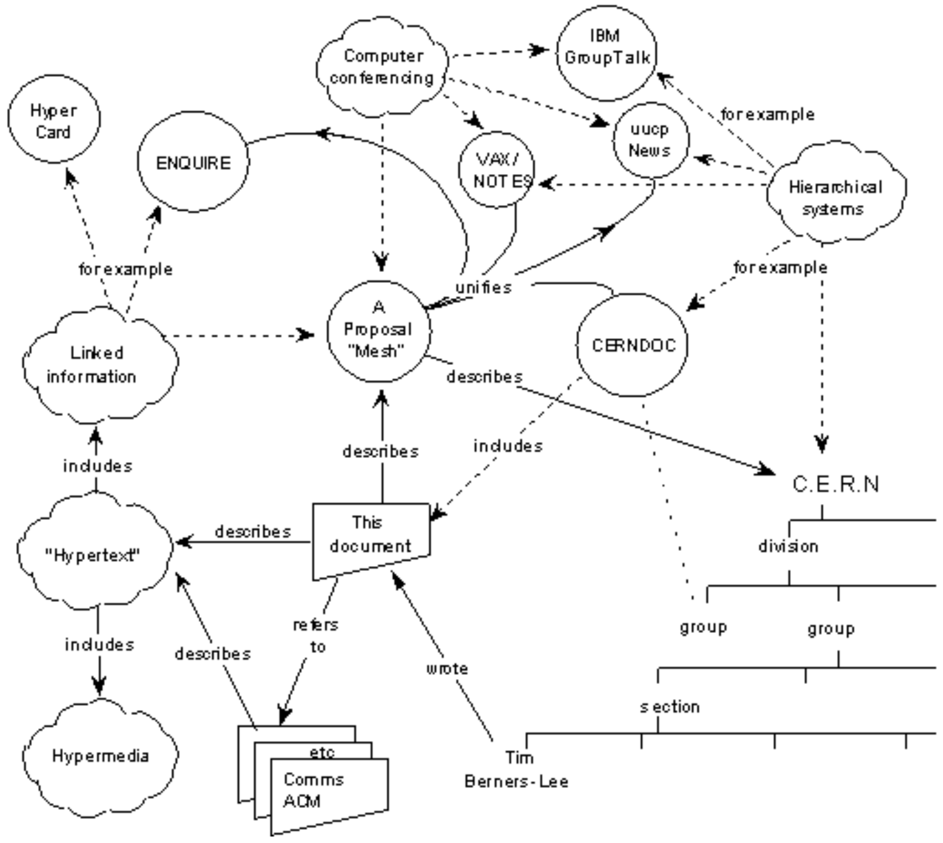
CHAPTER 2

The Web

HTTP Protocol

The World-Wide Web

- The Worldwide Web was invented by Tim Berners-Lee in 1991
- The web originally wasn't designed to deal with dynamic content.
- It was a web of linked documents, that only changed through manual editing.



- The internet preceded the web by about 10 years (in terms of broad availability).
- The web made the internet usable and became its primary user interface.
- The web began to be used for things it wasn't originally intended for.
- As a result the web became more like a remote application GUI rather than a collection of statically linked documents.

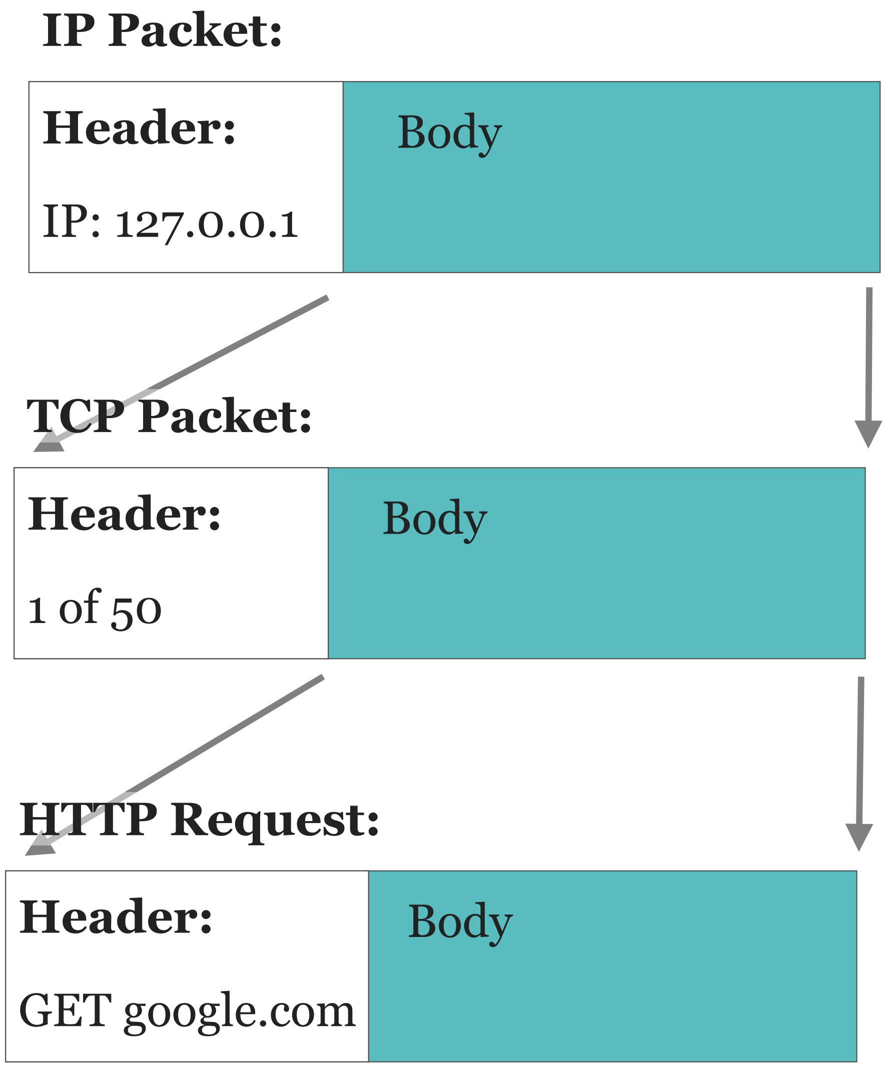
Web Protocols

- **Internet Protocol**
  - The underlying protocol of the internet
  - All information is stored in packets
  - The header contains the IP address of the destination

- Routers ensure that the packet gets to the destination
- Packets may arrive out of order
- Packets may be lost
- Packets are maximum 64KB
- Larger amounts of data must be split to fit into packets
- However, IP has no way of reconnecting these packets...

- **TCP/IP**
  - Transmission Control Protocol
  - TCP is a protocol that runs on top of IP
  - This means that the TCP headers are inside the IP packet body
  - TCP is connection-oriented
  - Three-way handshake to establish connection
  - Every packet received is acknowledged
  - If no acknowledgement is received then it is resent.
  - Large amounts of data can be split into smaller packets and reassembled and reordered at the receiver
  - TCP is a reliable form of communication
  - Most data transfer on the web happens within the TCP protocol
  - Being connection-oriented the connection remains open until one party closes the connection
  - Regular 'keepalive' packets are sent to check if the other party is still responding (typically every 1 sec to 1 min)
  - An IP address can have multiple servers by each server having a port number

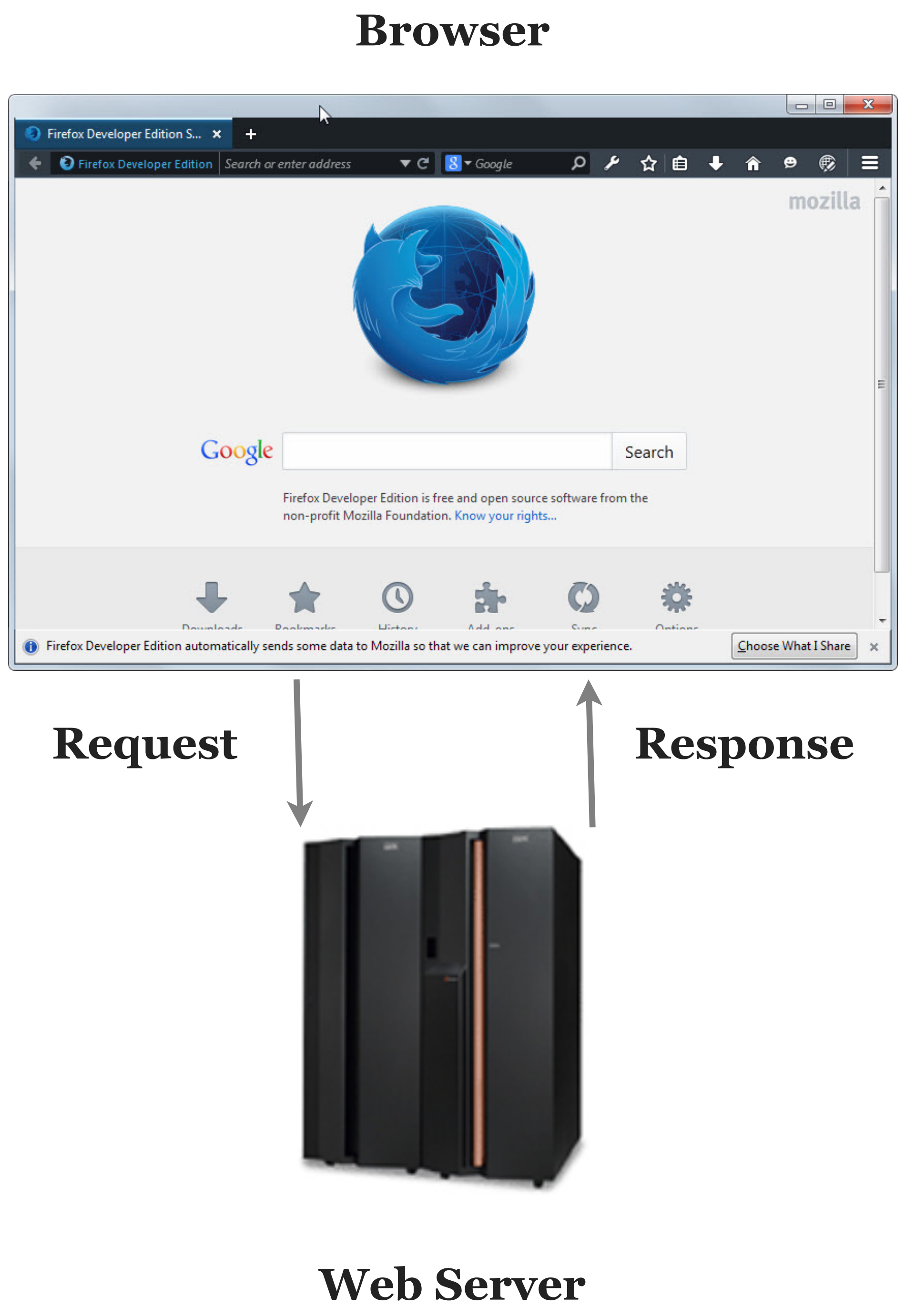
- **HTTP**
  - Hypertext Transfer Protocol runs on top of TCP
  - It is simpler than TCP:
    - Request is sent from client to server
    - Response is sent from server to client
  - TCP looks after opening the connection, keeping the connection alive, splitting packets, combining and ordering packets, and acknowledging packets.
  - In addition TCP is a binary protocol whereas HTTP is a text protocol which is easier to generate and read.
  - HTTP servers default to port 80



SECTION 1

HTTP Protocol

- The HTTP protocol largely revolves around ***requests*** and ***responses***, however there are some modern additions such as streaming content.
- The browser makes a request to the web server and the web server returns a response.



Firefox Developer Edition is free and open source software from the non-profit Mozilla Foundation. Know your rights...

Firefox Developer Edition automatically sends some data to Mozilla so that we can improve your experience.

cURL

- cURL is a command-line utility that allows us to send HTTP requests to servers without a browser
- It is useful for debugging HTTP requests.
- We can use cURL to access a website, e.g.:

```
curl google.com
```

- This will display the raw HTML from the web server, e.g.:

```
<HTML><HEAD><meta http-
equiv="content-type" content="text/
html;charset=utf-8">
<TITLE>302 Moved</TITLE></
HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
```

- The above result isn't important at this stage, but Google is actually indicating that it wants the browser to redirect to another URL (we will cover this in detail later).

Enabling Verbose (-v) Output

- The **-v** option stands for verbose and will print out the request and response headers:

```
curl -v google.com
```

- The following is returned:

```
* Rebuilt URL to: google.com/
* Hostname was NOT found in DNS
cache
*   Trying 74.125.237.167...
* Connected to google.com
(74.125.237.167) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.37.1
> Host: google.com
> Accept: */*
>
< HTTP/1.1 302 Found
< Cache-Control: private
< Content-Type: text/html;
charset=UTF-8
```

```
< Location: http://
www.google.com.au/?
gfe_rd=cr&ei=7pdhVNDnLamN8Qe1nYHgBg
< Content-Length: 262
< Date: Tue, 11 Nov 2014 05:00:30
GMT
* Server GFE/2.0 is not blacklisted
< Server: GFE/2.0
< Alternate-Protocol: 80:quic,p=0.01
```

- Lines starting with * are cURL status messages
- Lines starting with > is data cURL is sending to the server
- Lines starting with < is data cURL is receiving from the server
- The data sent to the server (in green) is the request message
- The data received from the server (in red) is the response message

- As you can see the packets are quite readable (i.e. are not in binary)
- We will now look at the basic format of the request and response packets.

HTTP Request

- This is the minimum HTTP header:

```
GET path HTTP/1.1
Host: host
```

- An example which would be equivalent to typing http://google.com in a browser:

```
GET / HTTP/1.1
Host: google.com
```

HTTP Methods

- The first token in the request is the HTTP Method.

- Valid HTTP Methods:

METHOD	DESCRIPTION
GET	Return data at URL
HEAD	Only return headers
POST	Send data in request body
PUT	Store data in body at URL (semantic)
DELETE	Delete data at URL (semantic)
TRACE	Return request message back to sender
CONNECT	Reserved (unused)

- Typical websites use **GET** to request data and **POST** to send data
- PUT and DELETE are now also commonly used in web applications which adopt the REST approach (we will cover this later in the course)

Request path

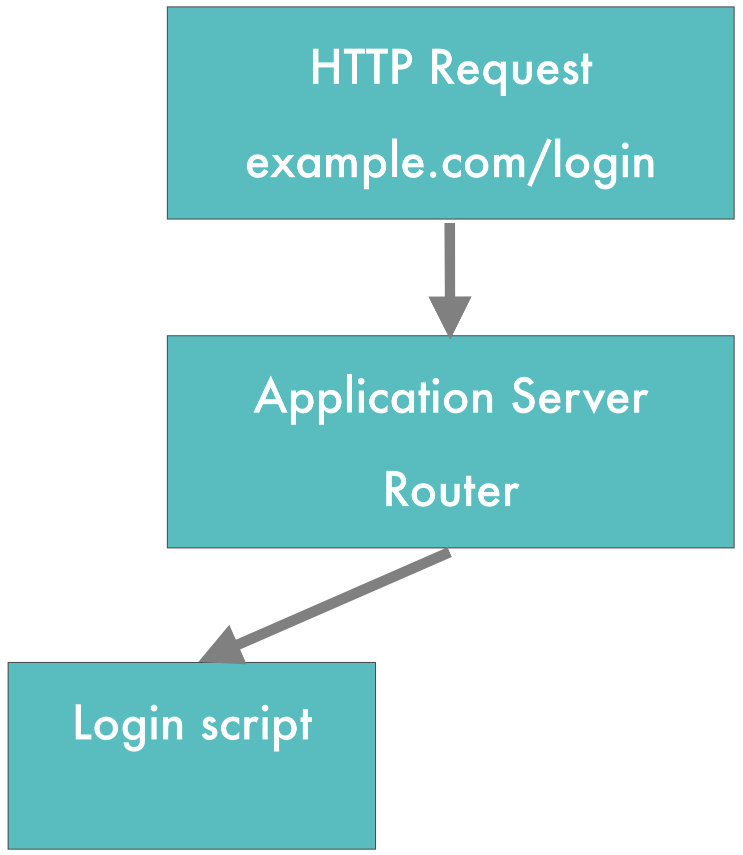
- The next component in the request is the path
- In the example above it is simply the root '/'
- Notice how cURL first rebuilds the url 'google.com' to include a slash at the end?
- Historically the path pointed to an actual file
- The primary file type web browser's supported was HTML ending with .html (or .htm)
- Generally web server's didn't give browsers access to the full file system.
- For example, if '/' truly accessed the root of the file system the browser would have access to the entire filesystem!

- Instead the web server generally had a subdirectory which was the web server root.
- For example a directory named 'public_html' or 'www' might be the root for the web server, only files within these directories could be accessed by browsers.
- **An example:**
  - A website, example.com, has its files stored in the path **/Users/web/public**
  - If the following URL is accessed:
    - http://example.com/docs/help.html
  - It would be accessing:
    - /Users/web/public/docs/help.html
- These days it is actually quite rare for a) files to be stored directly in .html files and b) for URLs to refer directly to files.

Modern Web Application Request Architecture

- Modern web applications rarely store files in static .html files, why?
- Tim Berners-Lee's original vision of the web was a web of linked documents.
- Now, however the web is being used as remote application GUI.
- Application GUIs are dynamic instead of the static nature of .html files, this is for two reasons:
  - Application GUIs update when users interact with them
  - Application GUIs often display data from somewhere else, often a database
- This is a very different architecture to what Tim Berners-Lee had originally envisioned, however it can still utilise the HTTP protocol and the HTML standard.

- In modern web applications the request URL doesn't necessarily need to link to a specific file.
- In such a setup all requests are sent to an application server which then uses a ***router*** to determine how to respond to the requested URL.



- This architecture is closer to what is known as a **Remote Procedure Call (RPC)**
- In essence the web browser is invoking code on the server and getting a response, i.e. a procedure is running remotely on the server, as opposed to simply retrieving a static HTML file.
- Therefore the URL in the HTTP request doesn't need to reflect an actual file on the server.

HTTP Protocol Version

- There are only two versions of HTTP: 1.0 and 1.1.
- HTTP/1.1 was broadly adopted in 1996 making it the only version is wide spread use today.
- A version of HTTP/2 is being worked on but has not been released.

Header fields

- The HTTP protocol allows for header fields.
- In the earlier example there is the ***Host*** header field
- This is a way for the client to communicate with the HTTP server.
- For example the client can indicate what data formats it supports through the ***Accept*** header field as shown in the earlier cURL example.
- The Accept header field specifies MIME types that can be received, in the example */* indicates that accepts all MIME types.

Sending Data

- There are two ways to send data via HTTP:
  - URL parameters
  - Message body

URL Parameters

- The '**?**' character in a URL indicates that the following data is a list of URL parameters, e.g.:

```
example.com/login?username=bob
```

- URL's are sensitive to punctuation and should be URL escaped, replacing non-basic latin characters with their hexadecimal character codes, e.g. a space ' ' is replaced with %20, the character code for a space is 32 decimal, which is 0x20 hexadecimal:

```
example.com/login?user=bob smith
```

URL escaped:

```
example.com/login?user=bob%20smith
```

Message Body

- Data can also be sent in the message body
- The message body follows the header fields and requires a blank line between the header fields and message body.
- The URL Parameters example could be sent in the body using the following request:

```
GET /login HTTP/1.1
Host: example.com

username=bob
```

- The message body can contain other data types and these can be indicated by using a ***Content-Type*** header field, for example to upload a JPEG image.

HTTP Response

- The HTTP response is similar to the HTTP request, however it also needs to communicate if an error occurred through a ***status code***
- A simple response would be:

```
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 36

<html><body><p>Hi!</p></body></html>
```

Status Codes

- Some common status codes:

STATUS CODE	MESSAGE
200	OK
301	Moved Permanently
302	Found
303	See Other
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
408	Request Timeout
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

- Most status codes are purely semantic, however some expect action from the browser, for example 301, 302, and 303, expect the browser to send a new request to the provided URL in the ***Location*** header.
- This can be seen with the cURL example where Google is redirecting the request from google.com to google.com.au
- Web applications commonly use redirects to handle form input, we look at this in more detail later in the course.

SECTION 2

Web Browser

- The web browser communicates with the HTTP server and receives the following files:
    - HTML
    - CSS
    - JavaScript
    - Images, Video, other media
    - Data (e.g. XML, JSON)
- Below is a brief description of the files received from the server:

HTML

- HTML is an XML-like format which allows text to be marked up for presentation purposes.
- Ideally HTML should communicate the ***structure*** and ***semantics*** of web content
- The ***presentation*** or look of the web page should be placed in CSS files (it can be placed in HTML but should be placed in CSS)
- HTML files can be stored as ***static files*** on the server
- Most HTML content however is now ***dynamically generated*** as the content of the web page changes.
- To dynamically generate content an ***application*** needs to be running on the web server which produces the HTML.

- The generated HTML is identical to static HTML from the browser's perspective.

CSS

- CSS is used to ***style*** and ***position*** the HTML content.
- It generally isn't generated by the application.
- Instead the application will modify the HTML file, such as class or id, to invoke a different style.
- However certain tools such as LESS and SASS add functionality to CSS and 'generate' CSS as a result.

JavaScript

- Like CSS, JavaScript generally isn't generated by the application.
- JavaScript runs in the web browser and provides functionality without requiring code to run on the server.
- JavaScript can communicate with the server directly without requiring a page reload, often loading data in JSON format

Images, Video, and other media

- Servers often return images, video, and other media
- It is increasingly common for the media to be dynamically returned.
- For example, a profile image will be different depending on the user logged in.

Data (e.g. JSON, XML)

- XML was an early structured data format, however JSON is becoming increasingly popular because it is JavaScript syntax and maps well to the JavaScript object structure.

- Data is used to update the web page without causing a page reload.
- An example might be a live table of stock prices.

SECTION 3

Application Server

- Applications run on the web server to dynamically generate content.
- The language we will be using in this course is PHP which is a scripting language.
- Other common scripting languages used for web applications include:
    - Python
    - Ruby
    - Perl
    - JavaScript
- Compiled languages can also be used for web applications, such as:
    - Java
    - C#
    - Scala
- The application server performs the following functions:
    - Interacts with a database to provide dynamic content.
    - Performs server-side logic for user authentication and other processes.
- Application servers primarily return HTML which then references static CSS, JavaScript, and image files.
- However, increasingly application servers also send and receive JSON data from JavaScript running in the browser.

- In such cases the application server no longer needs to communicate in HTML as the primary HTML can be static and dynamic data retrieved via AJAX calls.
- This approach can be appealing but has issues with the time to load the first page, static HTML pages appear to be faster, as a result some web sites take a hybrid approach where the first load of the page is dynamically generated HTML and subsequent updates are via AJAX.
- Servers which primarily communicate in JSON or XML become effectively a web-based API, we refer to these as web services.
- A web service consists of URLs that accept and return data.
- Web pages and also non-HTML applications such as mobile apps can communicate with web services.
- Even though mobile apps may not use HTML for the presentation, they still use the HTTP protocol to communicate with the servers.

SECTION 4

Revision

- What is the protocol that is unique to the web?
- What is the commonly used version of HTTP?
- GET, POST, and DELETE, are known as ...?
- What is the typical HTTP response message and code for a successful HTTP request?
- What two protocols do HTTP packets reside within?
- What are the 3 tiers in the 3-tier architecture?