What is the Internet

Internet: networked computers

Each computer has an address - like mail or deliveries

- Local traffic goes to a local hub
- Local hub sends stuff for outside to a higher hub
- Until a hub has the destination "inside"
- Sends to the right local hub

Much redundancy, not just one path (unlike mail)

Internet was a US Defense Dept exercise

- To survive interruptions (Like nukes 🤯)

Internet Delivery

Internet Protocol **(IP) addresses** are not for humans

Domain Name System (**DNS**) is name for an IP address

Humans use names like `www.google.com`

Each "dot" separates a level in reverse

- ".com" knows all the domains inside it
- "google.com" knows all the domains inside it

Three domain parts is common, but more are possible

- **https://repository.library.northeastern.edu/**

Traffic Protocols

DNS lookups are one kind of traffic over the Internet

There are many others. Examples:

- Email
- Database
- Many online games
- Web

Internet is down!

"The internet is down" - Probably not

Could be

- Their local internet routing is down
- Some local web issue is down
- A common provider (example: Amazon) is broken
    - Preventing traffic to/from a lot of web sites

Very rare for a big part of the Internet to be "down"

Don't be that person

But don't be that person

The one that tells others they are "wrong"

We all know what they mean

What is a server?

"It Depends"

- **https://jvns.ca/blog/2019/12/26/whats-a-server/**

A server can be

- A program that responds to requests
- The machine that runs that program
- A virtual machine running on a physical machine

A server can run a server running a server

Generally, for this course a server is the program

Web Request/Response

For the Web

- A **client** program makes a **request**
    - "Client" is a program, not a user!
    - Client not always a browser!
    - Often a program running in a server!
- A **server** gets the request and gives a **response**

Fundamental Web: Each request gets one response

- No response without a request
- More complex options aren't basic HTTP
    - Ex: Websockets

Poor cases for basic web

- A stock-ticker app that is told when stocks change
- A weather app that is told when weather changes

Both of these worked very poorly on the early web

- Client can't be told of changes
- Client had to ask (**request**) repeatedly
    - A lot of traffic with no news
- Nowadays things may be fast enough to not care
    - Or we use newer, more advanced techniques
- Understand the request/response nature of web

Original Web was for linking scientific papers

- Text (**First Website**)
- Linking back and forth (Hypertext!)
- Readable on different platforms

Not WYSIWYG (What You See Is What You Get)

- Web described content, not what it looked like
- Even less visual than Wikipedia
    - But very informative and useful

HTTP - Hyper Text Transfer Protocol

Web provided unique benefits

- Common port (80 for HTTP, 443 for HTTPS)
    - Meant once you got through a firewall, you had access to everything
- Not tied to a particular appearance
    - Could work on different types of computer
- Tolerant of bugs/typos
    - Easy for unskilled programmers to use
- Human readable
- Indirectly Searchable
    - Stable!

How is the Web searchable?

- A program (crawler, spider, bot) reads a page
- Makes a list of all the links on that page
    - Adds any new links to list of pages to crawl
- Reads the text of the page and saves info (index)
- Repeats with next link on list

Users go to site with index, enter search terms

- Website gets search terms
- Website uses index to get matching links
- User sees list of matching links

You don't "Search" the web!

- You search someone's database of links
- "Search Engines" are about
    - Having a big database of links
    - Keeping that database up-to-date
    - Figuring out which links to offer

A Search Engine is just another Web Site!

- And the crawler is a web client program
    - Making requests
    - Parsing and Saving the pages in responses
- User makes requests to Search Engine Website
    - **https://google.com/search?q=smug+cats**
- Site returns web page of links

Web is stateless

Each request is considered by itself

- Without checking/knowing previous requests

Can go straight to any link

- Without passing through other pages

Response is based only on info in THIS request

- Response for Request is "stateless"
- **state** is a term we will use a lot
    - in different contexts

What about requiring login?

Isn't login stateful? (not stateless)

Yes and no. The *protocol* doesn't enforce that.

- Request comes in
- Based on info IN REQUEST, server decides:
    - send you elsewhere (redirect)
    - show you alternate content (login screen)
    - show you the requested material (content)

Request must contain the info to let server decide

There is no state in the **handling of** the request

Browser Rendering

Not every web client is a browser

- Ex: a spider is a web client and not a browser

Browsers decide what to do with content

- Often this means **rendering** an HTML page

Other options include:

- Displaying an image
- Playing a sound file
- Showing a PDF
- Saving a file

What is a URL?

A Uniform Resource Locator (**URL**)

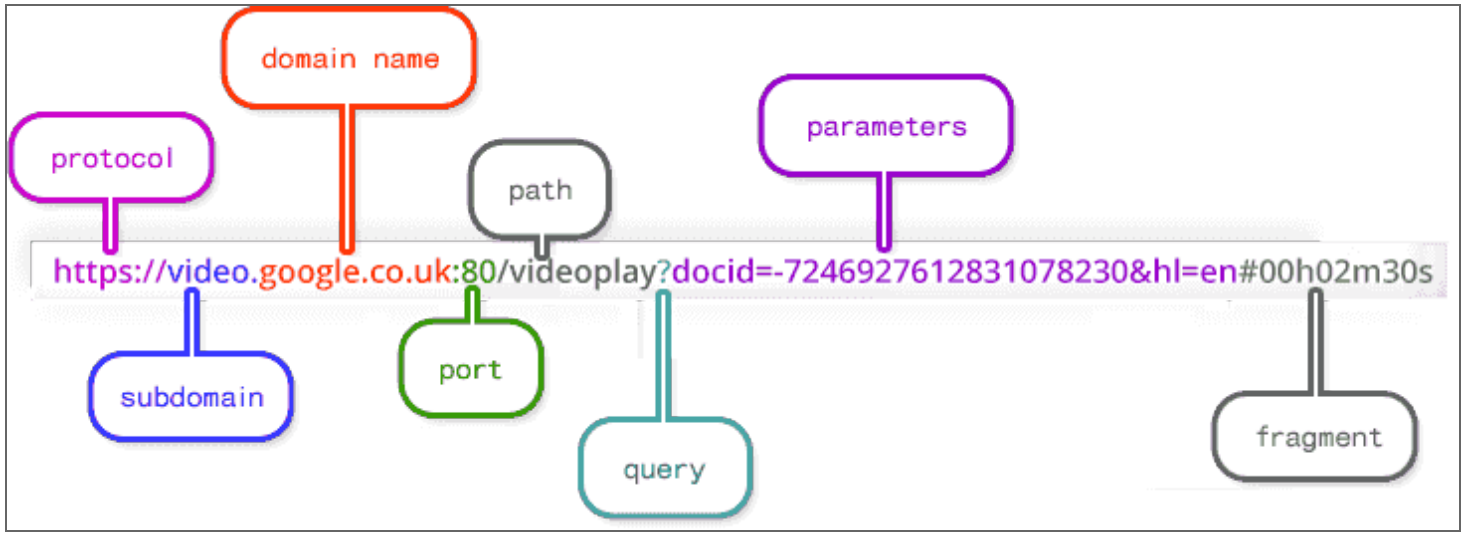
- Where something on the internet can be found

http://catalog.northeastern.edu/graduate/engineering/multidisystems-msis/

Often called a `Web Address`

- But URLs are not limited to the web

Parts of a URL



From **https://doepud.co.uk/blog/anatomy-of-a-url**

Fully Qualified

A URL with all required parts is "Fully-Qualified"

- Protocol
- Domain
- Path

Without all the parts, it might just be a path

That path can be "**absolute**" or "**relative**"

- Fully Qualified URLs have only absolute paths

Absolute Path

Absolute Paths are different paths on the same server

- Absolute Paths always begin with `/`

Absolute path is taken from some **"root"** of the server

- This is NOT the "root" of the file system

The **document root** is how the web server treats requests for the "root"

```
<img src="/images/complex_url.png">
```

Relative Path

A **Relative Path** is based on path of **current page**

- Relative Paths do NOT begin with `/`
- `<img src="cat.png"/>`
- `<img src="images/cat.png"/>`
- `<img src="../images/cat.png">`

Paths

Two pages:

- `http://example.com/foo/index.html`
- `http://example.com/bar/images/index.html`

What is different when the urls below are loaded?

- `<img src="cat.png"/>`
- `<img src="images/cat.png"/>`
- `<img src="/images/cat.png"/>`
- `<img src="../images/cat.png"/>`
- `<img src="images/../cat.png"/>`

Paths - Answered

`http://example.com/foo/index.html` vs
`http://example.com/bar/images/index.html`

- `<img src="cat.png"/>`
  - `/foo/cat.png` vs `/bar/images/cat.png`
- `<img src="images/cat.png"/>`
  - `/foo/images/cat.png` vs `/bar/images/images/cat.png`
- `<img src="/images/cat.png"/>`
  - Both `/images/cat.png` (absolute)

Paths - More Answers

`http://example.com/foo/index.html` vs
`http://example.com/bar/images/index.html`

- `<img src="../images/cat.png"/>`
    - `..` means "go up one directory"
        - Can't go earlier than root/document root
    - `/images/cat.png` vs `/bar/images/cat.png`
- `<img src="images/../cat.png"/>`
    - `/foo/cat.png` vs `/bar/images/cat.png`

Summary - Part 1

- Internet vs Web
- Internet routing
- DNS/**Domain names**/subdomains
- Web is **request**/**response**
- Web is **stateless**
- Searching isn't built in
- Searching is easy because stateless

Summary - Part 2

- URLs can be **fully qualified** or not
- A path can be an **absolute path** or a **relative path**
- Paths in URLs based on **document root**
- Browsers **render** a web page after getting the data
- Not all **clients** are browsers
- Not all data is rendered