Website Design

CSc 47300

Week 2

Web Applications; HTTP; Simple Application Server

Web Applications

So, What's This Web Application Thing About Anyway?

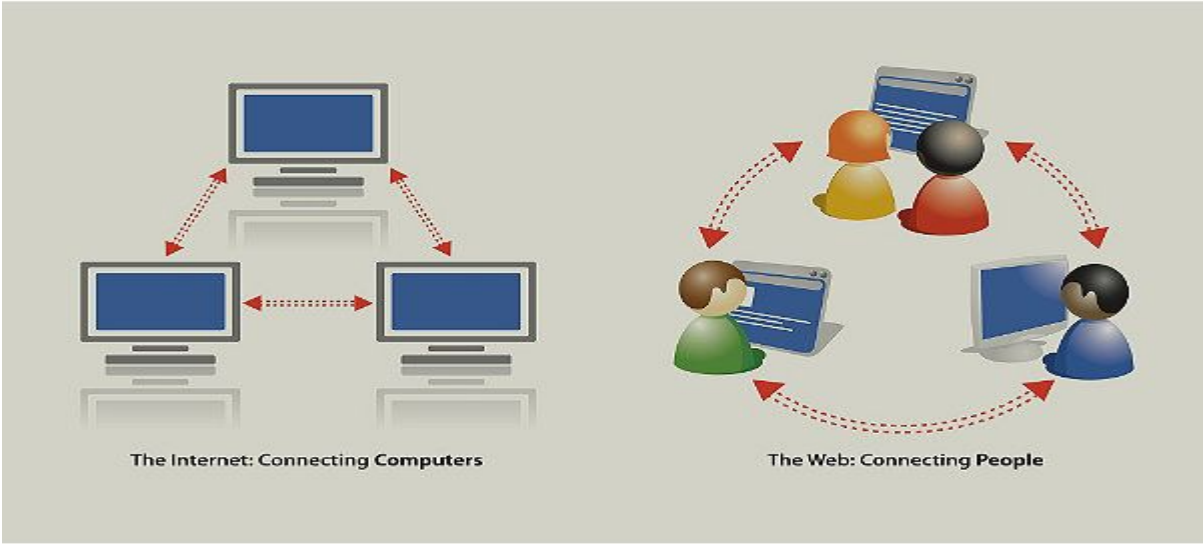
In this set of slides, we'll cover:

- Some definitions
- Simplified explanation of a couple of protocols (TCP, HTTP)
- Client / Server

Definitions

- **The internet** - global system of interconnected computer networks; a network of networks
  - Internet's underlying protocol for communication is **TCP/IP**
  - **TCP/IP** dictates how data should be packetized, addressed, transmitted, routed and received.
- **The web** - a collection of interconnected documents (web pages) and other resources (images, video, ect.), retrievable by url and connected by *hyperlinks*.
  - **HTTP** is the protocol used to allow documents and resources to be requested over a network.

Definitions (cont.)



Other Services

The web is just one of many services available on the internet…

What are some other services and protocols on the internet?

Other Services

The web is just one of many services available on the internet…

What are some other services and protocols on the internet?

- Email (SMTP)
- Chat (XMPP, OSCAR, IRC)
- File Transfer (FTP)
- Voice (SIP, Skype protocol)
- These are all examples of network protocols - ways of communicating over a network

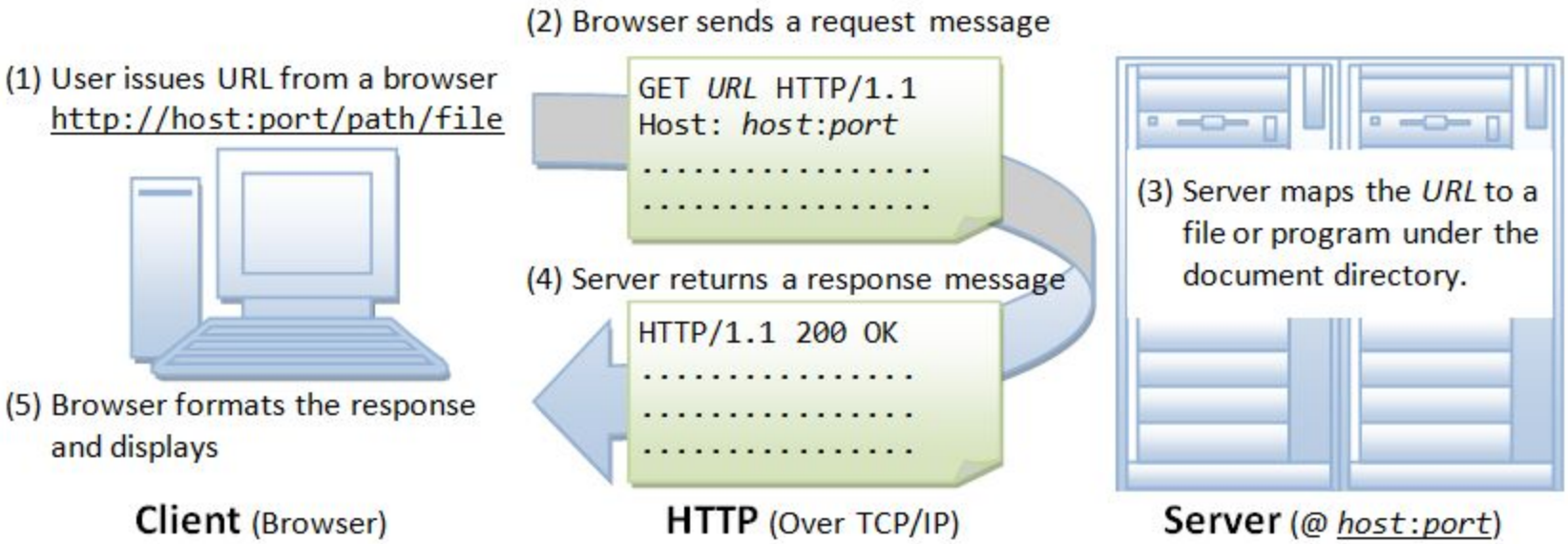
Protocols

Hm. All this talk about protocols but … what exactly is a protocol?

It's a bunch of rules and conventions for communication. Really. That's it.

For computers and communications between them, these rules may define:

- The format for exchanging messages
- A meaning (semantics) and syntax for these messages
- The process for synchronizing the communication


(1) User issues URL from a browser
http://host:port/path/file
(2) Browser sends a request message
GET URL HTTP/1.1
Host: host:port
.....
.....
(3) Server maps the URL to a file or program under the document directory.
(4) Server returns a response message
HTTP/1.1 200 OK
.....
.....
.....
(5) Browser formats the response and displays
Client (Browser)
HTTP (Over TCP/IP)
Server (@ host:port)

TCP/IP is the Underlying Protocol of the Internet

- Don't worry, that's about as in-depth as we'll get on TCP/IP
- However, just know that other application layer protocols are built on top of TCP/IP
- We're mostly interested in the web, and making web applications…
  - So we should take a look at application layer protocols
  - Specifically HTTP

The Web

What was our definition of the web again?

A collection of interconnected documents (web pages) and other resources (images, video, ect.), retrievable by url and connected by *hyperlinks*.

It All Starts With a URL

Each document or resource on the web is retrievable by a name, a URL (Universal Resource Locator).

It All Starts With a URL

Each document or resource on the web is retrievable by a name, a URL (Universal Resource Locator).

What are the parts to a URL?

It All Starts With a URL

Each document or resource on the web is retrievable by a name, a URL (Universal Resource Locator).

What are the parts to a URL?

- Scheme/Protocol - HTTP (Browsers accept schema-less, meaning you don't have to add HTTP/S)
- Domain or Actual IP Address (walmart.com)
- Port (Optional) - 80 (Default if HTTP), 443 (Default if HTTPS)
- Path - walmart.com/m/savings-spotlight
- Query String (Optional) - walmart.com/search/?query=Coffee
- Fragment Identifier (Optional) - github.com/psf/requests#supported-features--bestpractices

```
scheme://domain:port/path?query_string#fragment_identifier
```

Domains and IP Addresses

Each machine connected to the Internet gets a unique IP address.

We can map domains to IP addresses through DNS (Domain Name System)

- Both IP Addresses and domains are acceptable in a URL.
  - Google.com / 172.217.12.174 (IP obtained via `ping`)
- On OSX, Linux, and Windows, there's a file that allows you map names to IP addresses before DNS
  - Typically `/etc/hosts` or `hosts.txt`
- `localhost` maps to 127.0.0.1… which essentially is your computer

HTTP

HTTP

To retrieve documents on the web, we use HTTP (Hyper Text Transfer Protocol).

The computer/application asking for the document is the client or user-agent, and the computer responding to requests for documents is the server.

- The client (or the user-agent) is usually a browser (Chrome, Firefox)
  - There are clients other than browsers (Mobile apps, scripts, etc.)
- Generally, the server is going to be some sort of web server, like Apache or Nginx

HTTP is a request-response protocol, a very basic text-based (at least for version 1.1) communication method between computers:

- The client sends a request for some data. The server responds to the request.

HTTP (cont.)

The interaction between your browser and a web server goes something like this:

1. The browser (client) attempts to connect to the address of the server
2. If the server is listening and reachable, a TCP connection is made between the client and the server on port 80 (HTTP) or 443 (HTTPS)
3. The browser (client) sends a request message
4. On the same connection, the web server gives back a response message

A Request Message

A request consists of:

- Request line … which includes a request method and a path.
  - `GET /search/?query=Coffee`
- Request headers
  - `Host: www.walmart.com`
  - `User-Agent: curl/7.64.1`
- Here's a list of request header fields.

Request Methods

Here’s a list of available request methods. A request method (sometimes called a verb) tells the server what action to perform on the identified resource. A couple of common ones are:

Request Methods

Here's a list of available request methods. A request method (sometimes called a verb) tells the server what action to perform on the identified resource. A couple of common ones are:

- **GET**
  - The `GET` method requests a representation of the specified resource. Requests using GET should only retrieve data.

Request Methods

Here's a list of available request methods. A request method (sometimes called a verb) tells the server what action to perform on the identified resource. A couple of common ones are:

- **GET**
  - The `GET` method requests a representation of the specified resource. Requests using GET should only retrieve data.
- **POST**
  - The `POST` method is used to submit an entry to the specified resource, often causing a change in state of side effects on the server.

Request Methods

Here's a list of available request methods. A request method (sometimes called a verb) tells the server what action to perform on the identified resource. A couple of common ones are:

- **GET**
  - The `GET` method requests a representation of the specified resource. Requests using GET should only retrieve data.
- **POST**
  - The `POST` method is used to submit an entry to the specified resource, often causing a change in state of side effects on the server.
- **PUT**
  - The `PUT` method replaces all current representations of the target resource with the request payload.

Request Methods

Here's a list of available request methods. A request method (sometimes called a verb) tells the server what action to perform on the identified resource. A couple of common ones are:

- **GET**
  - The `GET` method requests a representation of the specified resource. Requests using GET should only retrieve data.
- **POST**
  - The `POST` method is used to submit an entry to the specified resource, often causing a change in state of side effects on the server.
- **PUT**
  - The `PUT` method replaces all current representations of the target resource with the request payload.
- **DELETE**
  - The `DELETE` method deletes the specified resource.

A Response Message

A response consists of:

- Status-line - which includes a status code and reason
  - `HTTP/2 200`
- Response Header Fields
  - `Content-Type: text/html`

And of course, a list of response header fields.

Status Codes

The status code that a server responds with is a numeric code that indicates the result of the request.

Some typical status codes are:

- 200 OK- request was successful
- 404 Not Found - Resource was not found
- 500 Server Error - Generic server error

Status Codes Continued

There are 5 different classes of status code:

- 1xx - Informational, request received
- 2xx - Success, request was received, understood, and accepted
- 3xx - Redirection, additional action must be taken to complete request
- 4xx - Client Error
- 5xx - Service Error

A Sample Interaction (Request)

```
curl -v -s https://www.google.com 1> /dev/null
```

```
> GET / HTTP/2
> Host: www.google.com
> User-Agent: curl/7.64.1
> Accept: */*
```

A Sample Interaction (Response)

```
curl -v -s https://www.google.com 1> /dev/null
```

```
< HTTP/2 200
< date: Tue, 08 Sep 2020 17:09:31 GMT
< expires: -1
< cache-control: private, max-age=0
< content-type: text/html; charset=ISO-8859-1
```

Tooling

curl

curl is a command line tool to transfer data to and from a server.

- Send a GET request
  - `curl -v -X GET https://www.google.com`
- Send a POST request
  - `curl -v -X POST https://www.google.com`

Chrome

Chrome comes with great tools for development!

1. Go to View → Developer → Developer Tools
2. Click on Network
3. Go to the Page
4. Watch the requests fly by!
5. (Why so many!?)

HTTP/2

HTTP/2 is out and being used.

- HTTP methods, status codes and semantics remain the same!
- However, the focus of the protocol is improved performance:
  - Binary format over plain text
  - Parallel requests can be made over the same connection
  - Server can push required resources to client even before client makes requests!
  - Compresses headers
- Check out the MDN page for the evolution of HTTP

Simple Application Server