Computer Networks 2

Application Layer

Responsibilities

- Writing data to the network
- Reading data from end user
- Providing applications to end user
- Internet is what it is today due to applications
- Eg: WWW, Instant messaging

Analogy to post

- Carry package to post office

Application Architectures

Client Server Architecture

- Client side and server side processes
- Communicate via messages

Server

- Owns the service. Eg: Website
- For big websites, sitting in data centres

Clients

- Consume content

P2P Architecture

- Applications communicate with each other.
- No dedicated server or large data centre
- Each peer is both server and client
- Eg: Bittorrent

Sockets

- Application is a program.
- It runs as a process
- And spans multiple threads
- Interface between network and process is socket.

Ephemeral ports

- From browser a process can use any ephemeral port to make calls

- Server needs to have fixed for connection

HTTP

- Client server protocol
- Specifies how requests should be made and how responses should be sent
- Uses TCP for guaranteed ordered delivery.

Stateless Protocol

- By default servers don’t store anything about the client

Objects

- Web is nothing but a series of objects being fetched
- Nothing but a file.
- Each object is at a location. Found via URL
- Normally fetching starts from a root html file and then others are recursively found.

URL

- Uniform Resource Locator
- Protocol | Hostname | Port | Location | Arguments

Types of HTTP Connections

Non Persistent

- Use TCP per request.

Persistent

- Single TCP connection for all requests in a session.
- Connection is cut after a time period of inactivity and is configurable.

HTTP Requests

```
GET /path/to/file/index.html HTTP/1.1
Host: www.scaler.com
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
Accept: text/html

```

- Each line ends with a \r and \n
- Messages in plain ASCII
- First line is request
- Other lines are headers
- Request line contains METHOD, URL, VERSION

HTTP Methods

- GET: Request data
- POST: Put something on the server without knowing the location. Server responds with the location. Technically it requests a page based on entered data.
- HEAD: Requests data but no response, only headers. Used for testing/ debugging/ checking if something exists.
- PUT: Upload at a URI. Replaces if something already exists there.
- DELETE: Delete object at a URI.

Interesting case of POST

- Unlimited data can be sent
- Often used for search queries etc as well for same reason

HTTP Headers

- Host: for whom is the request
- Connection: Persist the connection or not.
- User-agent: which type of device is making request
- Accept-Language: Preferred language of response
- Accept: Type of object (html etc.) that will be accepted

DEMO: Look a real HTTP Request

- Show in Developer Tools -> Network
- Reload Page
- Click anny entry

HTTP Responses

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15: 44 : 04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

[The object that was requested]
```

- Status Lines
- Header Lines
- Entity Body

Status Lines

- HTTP Version, Status Code

Status Code

- 1xx: info
- 2xx: Success
- 3xx: Redirection
- 4xx: Client Error
- 5xx: Server Error

Common Ones:

- 200 OK
- 404 File Not Found
- 500 Internal Server Error

Response Headers

- Date: at which response was generated
- Server: Server software
- Last Modified: Date of the object being sent
- Content Length: Length of the object being sent in bytes
- Content Type: Type of object. Not determined by file extension but this header field.

Curl

```
curl https://scaler.com --head
```

Cookies

- HTTP is otherwise stateless.
- How to maintain session states like:
    - Logged in or not
    - What is in your cart
- Cookies allow server to keep a track of these
- Can be used by third party to track you as well

How Cookies Work

- Unique string identifiers stored in the client
- Set by server through HTTP Headers
    - Set-cookie: value header
- Sent along with subsequent HTTP Requests
- Allows a server to know who is contacting it
- Server internally has a database of cookie to user

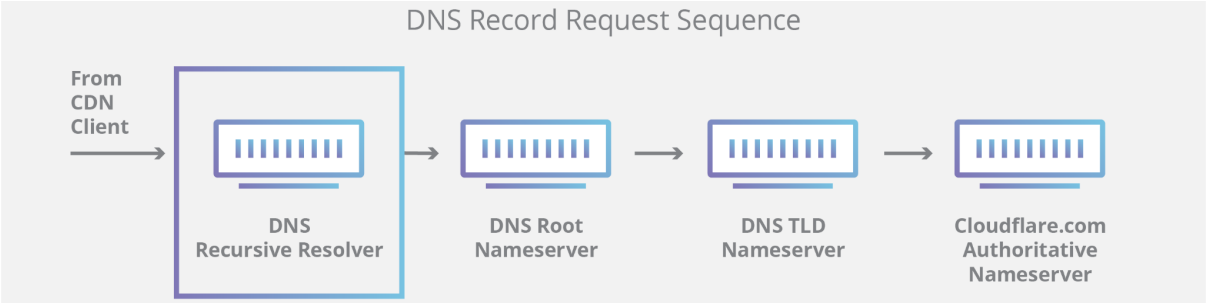
- Show cookies in developer tools -> storage tab.

DNS

- Internet uses IP addresses to find things
- But people don't remember numbers (how many phone numbers)
- We remember names
- Like google.com

- Need a system that maps name to ip.
- A single server can’t work as:
    - Single point of failure
    - Massive traffic

Hierarchical Database



- Root Servers return IP of TLD
- Point to authoritative NS
- Authoritative NS has info about that website
- Caching happens at a lot of places, including OS

DEMO: DNS

```
host -t ns google.com
```

```
cat /etc/resolv.conf
```