Welcome to GitHub Pages

Temporary While I figure out how to add a nav bar.

Week 3 Lecture 2

HTTP (HyperText Tranfer Protocol)

Web Terminology

- Webpage (document): Consists of objects
- Objects: files (html, jpeg, mp3, mp4 etc) that are addressable by a single url.
- If a web page contains html text and five images, then the webpage has 6 objects
- http://www.someschool.edu/someDepartment/picture.gif
- www.someschool.edu : the hostname
- /someDepartment/picture.gif : path name
- Web browsers implement the client side of HTTP (so web browser = client)
- Socket: Acts like a door between the client process and the TCP or UDP etc connection. Also door between the server and the TCP connection.
- RTT (Round Trip Time) :
- HTTP is implemented in two programs: The Client and the Server. They communicate with each other through HTTP messages. HTTP defines the structure of these messages.
- User requests a web page, browser sends HTTP request message for the objects on a webpage, server receives the requests and responds with a HTTP response message containing the objects.
- The client and server send the HTTP requests and responses into their socket information.
- Since TCP provides reliable data transfer service to HTTP, the request or response will eventually arrive intact on the other side without needing to worry about it.
- HTTP is a stateless protocol as it maintains no information about clients. e.g. If a client request the same objects again after a couple seconds the server will resend the entire object without remembering it already did that a second ago.

Non-Persistent and Persistent Connections

Usually a client and server are connected for a long period of time, and the client often makes a lot of requests to the server etc. The series of requests may be made back-to-back, periodically or intermittently (it depends on the application). The application developer needs to make the decision about whether each request/response is sent over different TCP connections or over the same one?

- Non-persistent Connections: Each request/response is sent over a different TCP or UDP connection. It goes through the whole process to initiate the TCP connection, sending the request message, reponse message, closing of the TCP connection etc for every single message that needs to be sent.

> - RTT : Three way handshake takes place. Then the client client sends through the HTTP request which requires another RTT. Thus the total response time is 2 RTT's plus the trasmission time.
> - For each connection, TCP buffers must be allocated and stored on the client and server. This can burden the web server which could be serving 100's of requests.

- Persistent Connections: Each of them are sent over the same connection. TCP connection remains open after sending a response. Subsequent requests and responses are sent over the same connection. A webpage and multiple webpags can be serviced over the same connection back to back without needing to wait for piplining. HTTP closes a connection when it isn't used for certain time.

HTTP Message Format

Request Message

- The first line of a HTTP request message is called the request line, all subsequent lines are called the header lines.
- The request line has three fields: Method Field, URL field and HTTP version field.
- The methods field can take on several different values (GET, POST, HEAD, PUT and DELETE).
- After the header lines there is the entity body. With the GET method, the body is empty, but with POST it is used. e.g. POST is used when a user fills out a form.

Response Message

- Contains the status line, six header lines and the entity body.
- Connection: Specifies whether to close or to keep the connection alive after the message is sent.
- DATE: Indicates the time and date when the HTTP response was created and sent by the server.
- Content Length: Indicates the number of bytes in the object being sent
- Content-Type: Indicates whether the body is html text or an image or etc.
- Common status codes:

> - 200 OK: Request succeeded and info is returned
> - 301 Moved Permanently: Object has been moved permanently. The client software will automatically retrieve the new URL.
> - 400 Bad request: Generic error code saying request was not understood
> - 404 Not Found: Domain does not exist on this server
> - 505 HTTP Version Not Supported: the requested HTTP protocol is not supported by the server.

Cookies

- HTTP server is stateless, so cookies are used to identify users and to track them.
- Has four components
    - cookie header line in the HTTP response message
    - cookie header line in the HTTP request message
    - cookie file kept on the user's end system
    - back-end database at the website

- When we visit a website like amazon, the server creates a unique ID and creates an entry in its back end database with the ID as its index.
- Set-cookie:
    - Webserver includes a Set-cookie header in its response message with and ID number
    - When the browser (client) receives the response message, it appends a line to the cookie file it manages with the hostname of the server and the ID number in the Set-cookie header.
    - Each time that website is visited the browser consults the cookie file and extracts the ID number for the site.

Web Caching (aka proxy server)

- Satisfies HTTP requests on behalf of an orgin server.
- Has it's own disk storage and keeps copies of recently requested objects.
- The browser can be configured so that all of the HTTP requests are first directed to the Web cache.
- When making a request
    - Browser establishes a TCP connection with the cache and sends a HTTP request
    - Web Cache checks to see if it has a local copy. If it does then it returns the object.
    - If it doesn't the web cache opens ip a TCP connection to the origin server and sends a HTTP request for the object
    - When the web cache receives the object it stores a local copy and then sends a HTTP response message to the client browser.
- The cache is both a server and a client.
- They substantially reduce traffic

The Conditional GET

- Although caching can reduce user-preceived response times, the copy of an object residing in the cache may be stale. The object on the webserver may have been modified.
- The request message uses the GET method and includes an IF-MODIFIED-SINCE header line.
- Process:
    - Browser requests an object
    - The cache would send a request to the server
    - Server sends response message with the requested object
    - Cache forwards the response message to the browser and caches the object locally.
    - Cache stores the last-modified date and the object
    - Blah blah sometimes late
    - Browser requests the same object
    - Since this could be modified, the cache ensures that it's contents are up to date by issuing a conditional GET.
    - The conditional GET tells the server to send to object only if the object has been modified since the specified date.
    - If not modified it will respond with "304 Not Modified". In this response the web server still sends a response message by not the requested object. This saves bandwidth and reduces user preceived time.

HTTPS

- HTTP is insecure

- HTTPS: HTTP over a connection encrypted by the TLS (Transport Layer Security). Before the actual get requests are sent, the TCP connection is established and over that the TLS connection is also established between the browser and the server.

  Issues with HTTP

  - HTTP headers are big. As they are all text, they can be quite large.
  - Objects have different dependencies (requiring extra RRT) If we have many small objects (images etc)
  - Head of Line Blocking "Slow objects" Delay later requests. Since objects are serviced by objects sequentially, if the first object requires a lot of time to fetch, and this would mean that any subsequent requests cannot be sent back because the server is still waiting to service the first request.
  - Browsers often open parallel TCP connections to the servers, which places load on the network and servers and increases the throughput. This does reduce HOL blocking. But this is not very desireable due to its downside.

  HTTP2

  - Binary instead of text
    - efficient to parse
    - more compact
    - less error prone.
  - Responses are multiplexed over a single TCP connection
    - Fast objects bypass slow objects (avoids HOL blocking)
    - fewer handshakes
  - Server can push things
    - Rather than sending the base file and then waiting for the client to request the embedded objects, the server can just push those embedded objects to the client.

Electronic Mail

- Three major components
  - User agents (browser basically)
  - Mail servers
    - Mailbox contains incoming messages for a user
    - Message Queue: outgoing mail messages
  - Simple Mail Transfer Protocol: Protocol used to exchange mails between the mail servers.

- SMPT
  - Older than HTTP
  - Uses TCP
  - Three phases
    - handshaking
    - Transfer of messages

        - closure
    - Proceedure:
        - Email is written with end users email address
        - User agent sends the message to the mail server where it is placed in the message queue
        - Client SMPT sees the message on the message queue and opens up a TCP connection to an SMPT server on the recipients mail server
        - SMPT handshaking takes place, and the mail gets sent into the TCP connection
        - At the recipients mail server, the SMPT receives the message
        - Mail server places the mail into the mailbox
        - Recipient invokes the user agent and reads the message
    - Direct transfer, SMPT does not use intermediate mail servers, and only communicates directly.
    - SMPT is persistent
    - Uses ASCII
    - Uses CRLF and CRLF to determine the end of the message
    - Comparisons with HTTP
        - HTTP: pull
        - SMTP: Push
        - Both have ASCII command/response interaction and status codes etc
        - HTTP each object is encapsulated in its own response msg
        - SMPT: Multipart message

Mail Access Protocols

- When the user is actually pulling the email, that takes place over a different protocol.
    - POP (post office protocol):
    - IMAP (Internet mail access protocol):
    - HTTP(S): Gmail, Yahoo...
- Why do we need to have the sender's mail sever? What is the catch? A user agent can directly connect with the recipients mail server without the need of the senders mail server?
    - ANS: The recipients server might not be ready to receive the email. Our mails could get lost etc
- Why do we have a separate Receivers mail server? Can't the recipient run the mail server on their own end system?
    - ANS: THe end users server may not always be on.