CS144: HTTP

Basic interaction

Example: Q: http://www.youtube.com interaction. What is going on behind the scene?

- Q: What entities are involved in this interaction?
- Q: What is the role of each entity?
  - Q: What runs on server? client? network? Who keeps track of what is being done?
- Q: There are many Web servers on the Internet. How can the Web browser reach and communicate with the YouTube server?
- Q: Many things are exchanged over Internet. Email, instant messaging, file transfer, etc. How does the server know that this client wants a "Web page"?
- Q: Only bytes are transferred. How do they communicate pages that contain text and rich, dynamic multimedia content?

Basic Internet Standards

- TCP/IP (transmission control protocol and internet protocol)
  - internet routing and transportation protocol
- DNS (domain name service)
  - internet protocol to map domain names to IPs
  - ICANN manages TLD (top-level domains)
- HTTP (hypertext transportation protocol)
  - communication protocol between web servers and web clients
- MIME (multipurpose internet mail extensions)
  - internet standard to specify the type of data being exchanged
- Text encoding
  - standard to represent text as a sequence of bytes
- HTML (hypertext markup language)
  - page structure standard

HTTP

- HTTP/2 is most recent, but HTTP/1.1 is most popular
- Request & response paradigm
  - all interactions start with a client's request

```
    -- request -->
client           server
    <- response --
```

- Stateless: every request is handled independently from others
  - Q: what are pros/cons of stateless protocol?

- Example:
  - Simple request and response example: telnet to http://oak.cs.ucla.edu/classes/cs144

- Real request example: http://oak.cs.ucla.edu/classes/cs144/examples/show_request/

- HTTP message = request/status line + header + body
- HTTP request
  - the bare minimum HTTP request (can be issued through telnet):

```
GET / HTTP/1.0
```

  - More realistic example

```
GET /cs144/examples/form.html HTTP/1.1  /* request line */
Host: oak.cs.ucla.edu /* beginning of header */
User-Agent: Mozilla/5.0 ...
Referer: http://oak.cs.ucla.edu/cs144/
Accept:text/xml,text/html;q=0.9,text/plain;q=0.8,image/png
   ,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Cookie: __utmz=125574670.1174236576.14.14... /* end of
   header */
```

  - request line: the actual request
    - `METHOD PATH PROTOCOL_VERSION`
    - more on the GET method later
  - header: additional information for the request
    - Host: the name of the web server
      - Q: why do we need the "Host:" field? Aren't we already contacting it?
    - User-Agent: information on the client software

  * Referer: The page linking to the requested page.
    - Q: where can Referer be used?

  * Accept ... : media/content type the client can accept q=... specifies the degree of preference of a particular type
  * Keep-Alive, Connection: in case we want to make multiple requests through one connection
    - Q: why do we want to make multiple requests per connection?

  * Cookie: more on this later

- HTTP response
  - e.g.

```
HTTP/1.1 200 OK   /* status line */
Date: Wed, 04 Apr 2007 03:20:33 GMT /* beginning of header
    */
Server: Apache
Last-Modified: Wed, 04 Apr 2007 03:19:25 GMT
ETag: "15b63b-af-ebdb0940"
Content-Length: 175
Connection: close
Content-Type: text/html  /* end of header */

<html> /* beginng of body. header and body is separated by
    an empty line */
<head><title>Example page</title></head>
...
```

  - Status line:
    * 2xx: Success - The action was successfully received, understood, and accepted

    * 3xx: Redirection - Further action must be taken in order to complete the request
    * 4xx: Client Error - The request contains bad syntax or cannot be fulfilled
    * 5xx: Server Error - The server failed to fulfill an apparently valid request
  - ETag: a unique tag that is the same only if the body is the same
    * Q: when will it be useful?
  - Content-Length: length of the body
  - Content-Type: the type of the content html, flash, pdf, etc.

- Looking at request and response using Chrome Developer console
- HTTP/2
  - Current standard (approved on Feb 17, 2015), but not yet widely deployed
  - Design rationale
    * Many objects need to be fetched to display a single page
      · ~ 100 objects, ~ 2MB
    * Web is often accessed through high-latency mobile connections
  - Makes it possible to
    * Send multiple objects through a single TCP connection
    * Reduces latency and overload
  - HTTP2
    * Uses binary format (not text)
    * Works with TLS (encryption) in most implementations
    * "Multiplexed streams" with priority specification
      · Indicates which stream to prioritize if resource constrained
    * Enables HTTP header compression
    * Enables "server push" (allows predictive cache "push" by server)
    * But its wide-scale adoption is still uncertain
    * More detail at https://daniel.haxx.se/http2/

References

- HTTP/1.1: RFC 7230 – RFC 7237
- HTTP/2: RFC 7540