HTTP: Hypertext Transfer Protocol

Lecture 11
HTTP

- Hypertext Transfer Protocol
- History
  - Developed at CERN, Tim Berners-Lee
  - Early 90’s
  - 1996: version 1.0
  - 1999: version 1.1 (current version!)
- Simple request/response
  - Client sends request to (web) server
  - (Web) server responds
  - “stateless” protocol
An HTTP request (or response) consists of

1. Header: meta information
2. Body (sometimes): the payload

The header consists of

1. Method (requests) / Status (response)
2. Header fields (separated by newlines)
3. Blank line
Protocol: Request, Response

Request

Method
Header field 1
Header field 2
Body

Response

Status
Header field 1
Header field 2
Header field 3
Body
Request Header: First Line

- Syntax of first line:
  
  \textit{verb path version}

  - Verb: GET, HEAD, POST, PUT, DELETE, ...
  - Path: part of URL (path-to-document)
  - Version: HTTP/1.1

- Example:
  
  - URL
    
    http://news.osu.edu/
  
  - Request begins
    
    GET / HTTP/1.1
Request Header: Header Fields

- Each field on its own line, syntax: 
  ```
  name: value
  ```

- Examples
  ```
  Host: cse.ohio-state.edu
  Accept: text/*
  Accept: image/gif
  If-Modified-Since: Sat, 12 May 2014 19:43:31 GMT
  Content-Length: 349
  User-Agent:
  ```

- Followed by blank line
"Nobody knows you're a dog"

GET / HTTP/1.1
Host: news.osu.edu
User-Agent: Mozilla/5.0 (X11; Ubuntu;...etc

Request

$ curl

$ telnet
Header Fields cont’d

- **Host**
  - Only required field
  - Q: Why is host needed?

- **Accept**
  - Browser preference for MIME type(s) to receive

- **If-Modified-Since**
  - Send payload only if changed since date
  - Date must be GMT

- **Content-Length**
  - Required if request has a body
  - # number of bytes in body
Demo

- Send this HTTP request to host web.cse.ohio-state.edu (on port 80):
  
  ```
  GET /~paolo/ HTTP/1.1
  Host: web.cse.ohio-state.edu
  ```

- At console
  
  ```
  $ telnet web.cse.ohio-state.edu 80
  ```

  Then type what to send (see above)
HTTP Response Anatomy

- Recall, four parts
  1. Status (one line)
  2. Header fields (separated by newlines)
  3. Blank line
  4. Body: the payload

- Parts 1-3 are the “header”

- Part 1, status line syntax:
  \textit{http-version status-code text}

- Examples
  
  \begin{verbatim}
  HTTP/1.1 200 OK
  HTTP/1.1 301 Moved Permanently
  HTTP/1.1 404 Not Found
  \end{verbatim}
## Taxonomy of Status Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1xx</td>
<td>Informational</td>
</tr>
<tr>
<td>2xx</td>
<td>Success</td>
</tr>
<tr>
<td>3xx</td>
<td>Redirection</td>
</tr>
<tr>
<td>4xx</td>
<td>Client Error</td>
</tr>
<tr>
<td>5xx</td>
<td>Server Error</td>
</tr>
</tbody>
</table>
Some Common Status Codes

- **200 OK**
  - All is good!
  - Response body is the requested document

- **301 Moved Permanently**
  - Requested resource is found somewhere else (please go there in the future)

- **304 Not Modified**
  - Document hasn’t changed since date/time in If-Modified-Since field of request
  - No response body

- **404 Not Found**
  - Server could not satisfy the request
  - It is the client’s fault (design-by-contract?)

- **500 Internal Server Error**
  - Server could not satisfy the request
  - It is the server’s fault (design-by-contract?)
Response Header: Header Fields

- Each field on its own line, syntax:
  
  \[
  \text{name: value}
  \]

- Examples

  Date: Mon, 22 Sep 2014 14:51:38 GMT
  Server: Apache/2.2.3 (Red Hat)
  Content-Type: text/html; charset=iso-8859-1
  Content-Length: 333

- Followed by blank line
Demo: Using Terminal

- Use telnet to retrieve
  
  `http://web.cse.ohio-state.edu/~paolo`
  
  - Fails (see status code)
  
  `http://web.cse.ohio-state.edu/~paolo/`
  
  - Body is incomplete (no images)
  
  - Body is chunked

- Use curl to retrieve
  
  - Handles headers, redirection, chunking,...

  `$ curl -L http://web.cse.ohio-state...`
Demo: Using Firefox

- Developer > Network
- One GET results in many requests
  - http://www.cse.osu.edu/~paolo
- For each request, see:
  - Request headers
  - Response status code
  - Response headers
  - Response (and preview)
Demo: Using Ruby

- Mechanize: A Ruby gem for HTTP
  ```ruby
  require "mechanize"
  ```

- Create an agent to send requests
  ```ruby
  agent = Mechanize.new do |a|
    a.user_agent_alias = "Mac Safari"
  end
  ```

- Use agent to issue a request
  ```ruby
  page = agent.get "http://www.osu.edu"
  ```

- Follow links, submit forms, etc
  ```ruby
  page.link_with(text: "Carmen").click
  ```
Request Methods

- GET, HEAD
  - Request: should be *safe* (no side effects)
- PUT
  - Update (or create): should be *idempotent*
- DELETE
  - Delete: should be *idempotent*
- POST
  - Create (or update): changes server state
  - Beware re-sending!
- HTTP does not enforce these semantics
Passing arguments (GET)

- Arguments are key-value pairs
  - Mascot: Brutus Buckeye
  - Dept: CS&E

- Can be encoded as part of URL
  - `scheme://FQDN:port/path?query#fragment`

- `application/x-www-form-urlencoded`
  - Each key-value pair separated by `&` (or `;`)
  - Each key separated from value by `=`
  - Replace spaces with `+` (arcane!)
  - Then normal URL encoding
  - `Mascot=Brutus+Buckeye&Dept=CS%26E`
Examples

☐ Wikipedia search
  http://en.wikipedia.org/
  w/index.php?
  search=ada+lovelace

☐ OSU job postings
  http://news.osu.edu/
  articles-by-category.html?
  tag=Press+release&
  key=article_category

☐ Random numbers
  http://random.org/passwords
  ■ Use FF Developer to submit, edit request
Passing Arguments (POST)

- Encoded as part of the *body*

- Advantages:
  - Arbitrary length (URLs are limited)
  - Arguments not saved in browser history
  - Slightly more secure (not really)

- Content-Type header field specifies
  - `application/x-www-form-urlencoded`
    - Same encoding scheme as seen with GET
  - `multipart/form-data`
    - Better for binary data (else 1 byte → 3 bytes)
  - More options too:
    - `application/xml`, `application/json`, ...
Summary

- HTTP: request/response
- Anatomy of request
  - Methods: GET, PUT, DELETE, POST
  - Headers
  - Body: arguments of POST
- Anatomy of response
  - Status Codes: 200, 301, 404, etc
  - Headers
  - Body: payload
- Tools
  - Curl, FF Developer, Mechanize