

CS631 - Advanced Programming in the UNIX Environment

HTTP; Code Reading

Department of Computer Science
Stevens Institute of Technology
Jan Schaumann

`jschauma@stevens.edu`

`https://stevens.netmeister.org/631/`

HTTP

Hypertext Transfer Protocol

RFC1945 (HTTP/1.0), RFC2616 (HTTP/1.1),
RFC7540 (HTTP/2)

A simple request/response protocol.

The Hypertext Transfer Protocol

HTTP is a request/response protocol:

1. client sends a request to the server
2. server responds

The Hypertext Transfer Protocol

HTTP is a request/response protocol:

1. client sends a request to the server
 - request method
 - URI
 - protocol version
 - request modifiers
 - client information
2. server responds

HTTP: A client request

```
$ telnet www.google.com 80
Trying 173.194.75.147...
Connected to www.google.com.
Escape character is '^]'.
GET / HTTP/1.0
```

The Hypertext Transfer Protocol

HTTP is a request/response protocol:

1. client sends a request to the server
 - request method
 - URI
 - protocol version
 - request modifiers
 - client information
2. server responds
 - status line (including success or error code)
 - server information
 - entity metainformation
 - content

HTTP: a server response

HTTP/1.0 200 OK

Date: Mon, 05 Nov 2018 02:52:21 GMT

Content-Type: text/html; charset=ISO-8859-1

Server: gws

X-XSS-Protection: 1; mode=block

X-Frame-Options: SAMEORIGIN

Set-Cookie: 1P_JAR=2018-11-05-02; expires=Wed, 05-Dec-2018 02:52:21 GMT; path=/; domain=

Set-Cookie: NID=144=XovVY1BCcjaIYkyum0C3F_7jB7J8jpvabZR_0-ORyWqhmEflAcNKrVdhSOT0144bF

```
<!doctype html><html itemscope="itemscope" itemtype="http://schema.org/WebPage">
<head><meta content="Search the...
```

The Hypertext Transfer Protocol

Server status codes:

- **1xx** – Informational; Request received, continuing process
- **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

HTTP: A client request

```
$ telnet www.cs.stevens.edu 80
Trying 155.246.89.84...
Connected to www.cs.stevens-tech.edu.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 301 Moved Permanently
Date: Mon, 05 Nov 2018 03:20:31 GMT
Server: Apache
Location: https://www.cs.stevens.edu/
Content-Length: 235
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
```

HTTP: A client request

```
$ openssl s_client -crlf -servername www.cs.stevens.edu -connect www.cs.stevens.edu:443
```

```
[...]
```

```
GET / HTTP/1.1
```

```
Host: www.cs.stevens.edu
```

```
HTTP/1.1 302 Found
```

```
Date: Mon, 06 Nov 2017 19:02:10 GMT
```

```
Server: Apache
```

```
Location: https://www.stevens.edu/ses/cs
```

```
Vary: Accept-Encoding
```

```
Content-Length: 214
```

```
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

HTTP: A client request

```
$ openssl s_client -crlf -servername www.stevens.edu -connect www.stevens.edu:443
[...]
GET / HTTP/1.1
Host: www.stevens.edu

HTTP/1.1 301 Moved Permanently
Date: Mon, 06 Nov 2017 19:09:21 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Set-Cookie: __cfduid=def9b13568803339571c6eff22b37f43c1509995361;
expires=Tue, 06-Nov-18 19:09:21 GMT; path=/; domain=.stevens.edu; HttpOnly
Last-Modified: Mon, 06 Nov 2017 13:27:15 GMT
Location: https://www.stevens.edu/schaefer-school-engineering-science/departments/computer-science
Via: 1.1 varnish-v4
Server: cloudflare-nginx
Strict-Transport-Security: max-age=15552000
```

HTTP: A client request

[...]

```
GET /schaefer-school-engineering-science/departments/computer-science HTTP/1.1
```

```
Host: www.stevens.edu
```

```
HTTP/1.1 200 OK
```

```
Date: Mon, 06 Nov 2017 19:11:20 GMT
```

```
Content-Type: text/html; charset=utf-8
```

```
Set-Cookie: __cfduid=dfa773452cb45ba58a5b356568cb3715e1509995480;
```

```
expires=Tue, 06-Nov-18 19:11:20 GMT; path=/; domain=.stevens.edu; HttpOnly
```

```
Last-Modified: Mon, 06 Nov 2017 15:44:53 GMT
```

```
Via: 1.1 varnish-v4
```

```
X-Generator: Drupal 7 (http://drupal.org)
```

```
X-Varnish: 12117551 14890606
```

```
Strict-Transport-Security: max-age=15552000
```

```
Server: cloudflare-nginx
```

```
<!DOCTYPE html>
```

```
<html lang="en" class="no-js">
```

HTTP: A client request

The screenshot shows a web browser displaying the Stevens Institute of Technology Department of Computer Science page. The browser's developer tools are open to the Network tab, showing a waterfall chart and a table of network requests.

The waterfall chart shows the following requests:

Name	Status	Type	Initiator	Size	Time
www.cs.stevens.edu	301	text/html	Other	295 B	485 ms
www.cs.stevens.edu	302	text/html	www.cs.steve...	286 B	429 ms
cs	301	text/html	www.cs.steve...	544 B	337 ms
computer-science	200	docum...	/ses/cs	116 KB	167 ms
d0b486e3-1465-45e2-b0d2-2f285e3dd330.css	200	stylesh...	computer-scie...	2.0 KB	495 ms

HTTP - more than just text

HTTP is a *Transfer Protocol* – serving *data*, not any specific text format.

- Accept-Encoding client header can specify different formats such as *gzip*, *Shared Dictionary Compression over HTTP (SDCH)* etc.
- corresponding server headers: Content-Type and Content-Encoding



HTTP - more than just static data

HTTP is a *Transfer Protocol* – what is transferred need not be static; resources may generate different data to return based on many variables.

- CGI – resource is *executed*, needs to generate appropriate response headers
- server-side scripting (ASP, PHP, Perl, ...)
- client-side scripting (JavaScript/ECMAScript/JScript,...)
- applications based on HTTP, using:
 - AJAX
 - RESTful services
 - JSON, XML, YAML to represent state and abstract information

Code Reading

Let's take a look at some sample implementations:

- mathopd: <http://www.mathopd.org/download.html>
- Null httpd: <http://nullhttpd.sourceforge.net/httpd/>
- muhttpd: <http://inglorion.net/software/muhttpd/>

Walk us through the code:

- networking setup (`socket(2)`, `bind(2)`, ...)
- request handling
- header parsing
- CGI execution

HTTP in your final project

- protocol versions supported: 1.0
- request methods supported: GET, HEAD
- request headers supported: If-Modified-Since
- response headers included: Date, Server, Last-Modified, Content-Type, Content-Length

HTTP in your final project

- accept connections, read input from client
- timeout idle connections
- parse and validate input
- generate proper HTTP codes
- log connection information
- send headers
- send response
- close connection

HTTP in your final project

Client data parsing:

```
method uri protocol  
<optional headers>
```

- *method* not in GET, HEAD? => 400
- *protocol* != "HTTP/1.0"? => 505
- *uri* too long? => 400

HTTP in your final project

URI processing:

- begins with "~/something"? => userdir handling
- begins with "/cgi-bin/"? => cgi handling
- avoid breaking out of docroot via "../.." etc.
- ends in "/"? => look for 'index.html' or generate index on the fly

HTTP in your final project

Let's read the RFC and collect test cases...

Reading

HTTP etc.:

- RFC 1945, 2616, 2818, 3875, 7540
- <https://httpd.apache.org/docs/current/>
- <https://www.w3.org/Protocols/>
- REST: <https://is.gd/1eSvGa>