The HTTP protocol

Michael Mrissa
michael.mrissa@upr.si

Univerza na Primorskem

Acknowledgments

- Lionel Médini
- Olivier Glück
- Emmanuel Coquery
- Pierre-Antoine Champin

HTTP - General reminder

- HTTP : Hyper Text Transfer Protocol
  - Dedicated to the Web (origin : CERN, 1990)
  - RFC 2616 (HTTP 1.1), RFC 7540 (HTTP 2.0)
  - Client / server mode
    - No server→client notifications (but extensions)
  - Standard port : 80
    - Why is port 80 important ?
    - Massive adoption, first for Web sites
    - Then for applications

HTTP - General reminder

- Stateless protocol
  - What does it mean ?
    - Warning : states do exist, on both client and server sides
  - Light transaction management
    - No information kept between 2 exchanges
    - Allows the HTTP server to scale better
      - No memory space to allocate for sessions etc.
  - Requires a mechanism to manage sessions
    - cookie, Id in URL, hidden form field...

HTTP - History

- HTTP 0.9 : first version
  - Only one method : GET
  - No headers
  - One request = one TCP connection
- HTTP 1.0 : improvements (1)
  - Headers ("meta" information)
  - Use of caches
  - Authentication methods...

HTTP - History

- HTTP 1.1 : improvements (2)
  - Default persistent connection mode
    - Several HTTP transactions (with ressources) for one TCP connection
    - The connection is maintained as long as the server or client do not decide to close it (with connection: close)
  - Virtual servers
    - Host directive in the query is necessary

HTTP - History

- HTTP 2.0 : principles
  - Based on the SPDY protocol (Google)
  - RFC 7540 (may 2015)
  - Same syntax as HTTP 1.1 (methods, status codes, headers...)
  - Additions
    - Push of necessary resources from server
    - Request multiplexing
    - Header compression
    - Security layer (TLS) mandatory *de facto*

Request format

- HTTP commands
  - Methods : GET, POST, HEAD, PUT, DELETE, TRACE, OPTIONS, CONNECT
  - URL from server root
  - HTTP Version
- Headers (set of lines)
  - Header name : value
- One empty line
- Contents (can be empty)
  - Parameters to be processed by the server

GET method

- Standard method to ask for a representation of a resource
  - Can deliver a file, an image...
  - Can activate a program by transmitting data
- The body of the request is always empty
- Parameters are added after the resource name
  - Transmitting data in the URL after a « ? »
  - Fields separate by a « & »
- GET /index.php?email=toto@site.fr&pass=toto&s=login HTTP/1.1
- Comments
  - All HTTP request are as secure as sending a postcard
  - Some security can be reached through HTTPS
  - All the data is full text in the URL, not encrypted
  - URL has a limited size of 4Ko

HEAD Method

- Similar to GET
  - Body of request always empty
  - Retrieve only the headers
- Useful for getting
  - Last modification date (caches, JavaScript)
  - Size (estimating time of reception of document)
  - type (content negociation, see later)
  - Get some information about the server
- Warning
  - Servers do not necessarily give that information

POST method

- Transmitting data in the request body
- Data is also full text

```
POST /directory/index.php HTTP/1.1
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: localhost
Accept: */*
Content-type: application/x-www-form-urlencoded
Content-length: 36

email=toto@site.fr&pass=toto&s=login
```

Some request headers

- Client identity
  - From : email address of client
  - Host : server, mandatory since HTTP 1.1
  - Referer : URL the client comes from
  - User-Agent
- Client preferences
  - Accept : list of accepted MIME types
  - Accept-Encoding : compress, gzip...
  - Accept-Langage
  - Accept-Charset

Some request headers

- Objective
  - Do not send an object that is already in the client cache
- Problem
  - Objects in cache can be obsolete
- Solution
  - Client specifies the date of the cached copy in the HTTP request
  - If-modified-since : date
  - Server sends an empty reply if the cached version is up to date

Some request headers

- Information for the server
  - Autorization (username:passwd, base64 encoded)
  - Cookie
- Reply condition
  - If-Modified-Since : useful for caches
  - If-Unmodified-Since
  - If-Match (Etag)

Response format

- Response type
  - HTTP version
  - Response code
  - Code description
- Headers (set of lines)
  - Header name : header value
- Empty line
- Possibly contents
  - Encoded according to the specified MIME type

Response codes

- Objective
  - Give request result : success or failure
  - In case of failure, the contenu of the response must always describe the reason
  - Ex : file not found, access forbidden
- Code classes
  - 100-199 : information
  - 200-299 : success
  - 300-399 : redirection
  - 400-499 : failure due to client
  - 500-599 : failure due to server
- More information
  - http://www.codeshttp.com/

HTTP/1.1 200 OK
HTTP/1.1 304 Not Modified
HTTP/1.1 403 Forbidden
HTTP/1.1 404 Not Found
HTTP/1.1 500 Internal Server error

Some response headers

- Document content
  - Content-Type : MIME type of document
  - Content-Length : indicate loading progression
  - Content-Encoding, Content-Location, Content-Langage
- The document itself
  - Last-Modified (self explanatory)
  - Allow : authorized methods
  - Expires : expiration date of the document
- General headers
  - Date : request date
  - Server : server type

A typical transaction (1)

- Client request: client => server
  - 1. Requesting the test.html document

```
GET /~mydirectory/test.html HTTP/1.1
```

  - 2. Sending header information: inform the server
    - configuration
    - Accepted documents

```
User-Agent: Mozilla/5.0 (compatible;MSIE 6.0;Windows NT 5.1)
Host: www.upr.si
Accept: image/gif, image/jpeg
```

  - 3. empty line (end of header)
  - 4. contents (empty in this example)

A typical transaction (2)

- Server response: server → client
  - 5. code for request state

    HTTP/1.1 200 OK
  - 6. sending header information : inform the client
    - Server configuration
    - document asked

    Date: Tue, 30 Sep 2008 06:11:28 GMT
    Server: Apache/1.3.34 (Debian) PHP/5.2.1
    Last-Modified: Tue, 30 Sep 2008 06:11:14 GMT
    ETag: "600593b3-61-48e1c302"
    Accept-Ranges: bytes
    Content-Length: 97
    Content-Type: text/html; charset=iso-8859-1
- 3. empty line (end of header)
- 4. contents if request was successful

Cookies

- Remember HTTP is stateless protocol
  - Needs for a means to deal with sessions → cookies
- Cookie
  - URL-encoded string of characters of 4ko max stored on client hard drive
  - Information associated to a set of URLs, used for all requests to any of them
- Cookies allow
  - To propagate access code: avoid authentication at each request
  - Facilitate access to databases on the server
  - Gives statistics to the server : visited page count, etc.
- Side note
  - Cookies are not the only way to manage session

Install a cookie on the client

- « Set-Cookie » directive in header of HTTP response (sent during first connection)

```
Set-Cookie: name=value; expires=date; path=my_path;
domain=domain_name; secure
```

  - name=value : cookie content, without any space, semi-colon or comma (only mandatory field)
  - expires : become invalid after that date
  - path=/pub : cookie is valid for all request under that path (/pub)
  - domain : domain name for which cookie is valid
  - secure : cookie is valid only for secure connection

Client-side Cookie Usage

- Before each request the client (browser) checks for a corresponding cookie
- If found, it uses the Cookie directive in the HTTP header

```
Cookie: name1=value1; name2=value2; ...
```

- The server can insert several Set-Cookie directives
- In the first cookie specification :
  - A client can store a maximum of 300 cookies
  - A maximum of 20 cookies per domain is allowed
  - Maximum size of a cookie is 4Ko
  - Meaning max. 15 domains for a size of ~1,2Mo

Chunk Transfert (HTTP/1.1)

- A server response can be sent in several chunks
  - Sometimes the server may not know the total size of the response

```
Transfer-Encoding: Chunked
```

- Each chunk is made of one line
  - Size of the chunk in hexadecimal
  - data
- After the chunks, one line
  - 0 (zero)
  - Eventually additional headers

Resource Encoding

- Problem description
  - A Web server can serve different types of resources
    - text, Web pages, images, documents, executable files...
- Each type of resource is encoded in a different way
- A client must know the resource type to be able to process it
  - View in the browser, use of plugin, external app
- Solution (HTTP & Internet in general)
  - MIME : Multi-purpose Internet Mail Extensions
  - Recognized in HTTP : since v1.0

Resource Encoding

- MIME types : a composition
  - General type : text, image, audio, video, application...
  - Sub-type : depends on general type
  - Examples : image/gif, image/jpeg, application/pdf, application/rtf, text/plain, text/html...
  - In constant evolution
- MIME types : use
  - Server sets a Content-type header
    - Ex : Content-Type: text/html; charset=UTF-8
  - Client associates each MIME type to a specific processing

Character Encoding

- Reminder : character encoding
  - Principle : assign an integer to each character of a text
  - No confusion between character encoding (charset) and type (MIME) of file
- Problem description
  - Different character sets
    - ANSI, Occidental Europe, simplified Chinese, etc.
- Different encoding norms
  - Depend mostly of OSes and their configuration
    - ASCII, Windows-1252, ISO Latin 1, Unicode 8, 16 or 32...
  - Transmission through the Web : multi-platform
    - Independent from the OS and from client or server configuration

Character Encoding

- Encoding of characters used in a resource
  - Considered as a subpart of resource encoding
    - Related to the MIME type of the resource
    - Related to the language of the resource
  - Indicated in HTTP headers of responses
    - Content-Language: en, fr
    - Content-Type: text/html; charset=ISO-8859-1

Encoding of request parameters

- Formating : URL-encoded
  - For client to encode data in URL (GET method)
  - URL syntax (RFC 2396)
    - Beginning of parameters indicated by « ? »
    - Field name and value separated by « = »
    - Field separation: « & », spaces in a field value: « + »
    - Reserved characters: ; / ? : @ & = + $,
    - Non-alphanumeric characters replaced by %xx
    - (where xx = hex. ASCII code of character)
  - Example : name1=value1&name2=value2&...
  - Fields with multiple values (selection lists)
    - list-name=value1&list-name=value2&...

Server-side Encoding

- Operation
  - The MIME type is set from file extension (/etc/mime.types)
  - Encoding is set after negotiating with the client (mod_negotiation)
  - http://httpd.apache.org/docs/2.0/content-negotiation.html
  - It is possible to set a particular encoding for parts of a site with .htaccess files
- Modules
  - Apache : mod_mime
    - http://httpd.apache.org/docs/2.0/mod/mod_mime.html
  - nginx : ngx_http_charset_module
    - http://nginx.org/en/docs/http/ngx_http_charset_module.html

Comments on Encoding

- In a browser
  - The request specifies what encodings are known by the client
    - Accept, Accept-Language, Accept-Charset, Accept-Encoding
- Like other HTTP headers, encoding can be specified in the HTML code
  - <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1"/>
- In the response, if a HTTP header and a meta element are contradictory, priority is given to the server header
- Small exercise : try to discover UPR configuration