The Web

Jens Lechtenbörger

Neuland im Internet 2019

Contents

1 Introduction

1.1 Learning Objectives

- Explain message format and `GET` requests of HTTP as application protocol
- Perform simple HTTP requests via `telnet` or `gnutls-cli`

1.1.1 Recall: Internet Architecture

- "Hourglass design"

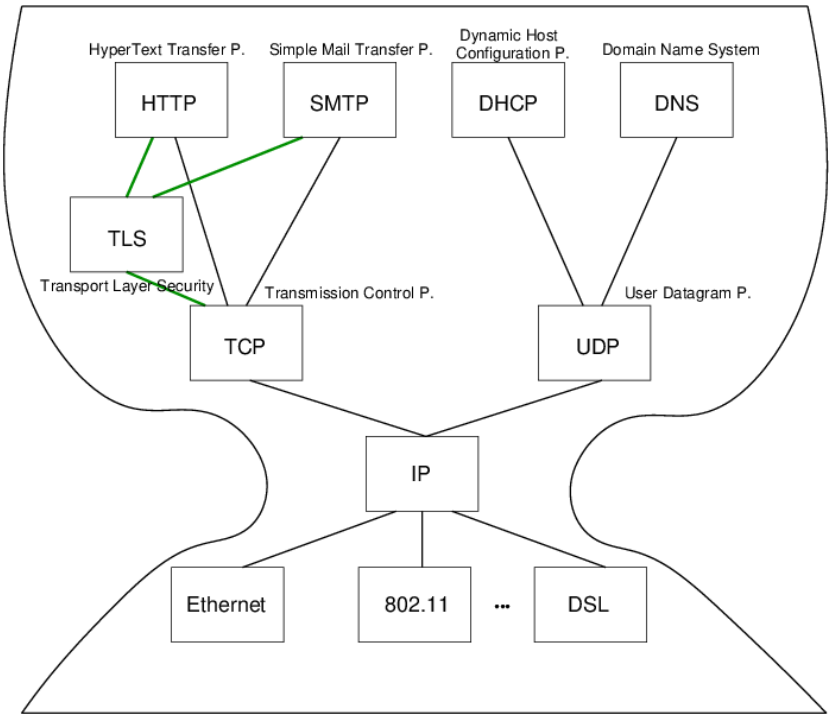

Figure 1: Internet Architecture with narrow waist

- IP is focal point
  - "Narrow waist"
  - Application independent!
    - Everything over IP
  - Network independent!
    - IP over everything
- Now: **HTTP** at application layer

1.2 Today's Core Question

- What does your browser do when you enter a URI in the address bar?

2 Web

2.1 History of the Web (1/2)

- 1945, Vannevar Bush: As we may think
  - Memex for information storage
  - Associative indexing (Hyperlinks)
- 1989, article by Tim Berners-Lee

  - Distributed hypertext system, "»web« of notes with links"
  - Initially for cooperation among physicists at CERN
- May 1991
  - Distributed information system based on HTML, HTTP, and client software at CERN
- August 1991
  - Availability of CERN files announced in `alt.hypertext`
    - `http://groups.google.com/group/alt.hypertext/msg/395f282a67a1916c`

2.2 History of the Web (2/2)

- 1992, NCSA **Web Server** available
  - National Center for Supercomputing Applications, University of Illinois, Urbana-Champaigne
- 1993, Mosaic **browser** created at NCSA
- 1994, World Wide Web Consortium (W3C) founded by Tim Berners-Lee
  - Publication of technical reports and "recommendations"
- Now
  - Web 2.0, Semantic Web, cloud computing, browser as access device

2.3 WWW/Web

- Standards
  - W3C (HTML 4 Specification)
    - "The World Wide Web (Web) is a network of information resources."
  - HTTP/1.1 Specification (RFC 7230)
    - "The Hypertext Transfer Protocol (HTTP) is a stateless application-level protocol for distributed, collaborative, hypertext information systems."
- Distributed information system
  - Client-Server architecture
    - Web browser (client) sends HTTP requests to Web server
  - Based on
    - Internet
    - URIs (Uniform Resource Identifiers, generalize URLs and URNs)
    - HTTP (now)
    - ((X)HTML)

3 HTTP

3.1 HTTP

- Hypertext Transfer Protocol
  - HTTP/1.1, RFC 7230
    - Plain text messages, discussed subsequently
  - HTTP/2, RFC 7540
    - Adds frame format with compression
    - Adoption increasing, from 15% in July 2017 to 28% in July 2018 to 33.4% (as of 2019-07-03; after peak of about 37% in June 2019)
  - HTTP/3 upcoming
- Request/response protocol
  - Specific message format
  - Several access methods
  - Requires reliable transport protocol
    - Typically TCP/IP, port 80 (or port 443 for HTTPS)

3.2 Excursion: Manual Connections

- HTTP (before HTTP/2) and SMTP are plain text protocols
  - With encrypted variants HTTPS and SMTPS (or STARTTLS)
- Enables experiments on the command line
  - Type (or copy&paste) request, see server response
  - For unencrypted connections, `telnet` can be used (preinstalled or available for lots of OSs)
  - For encrypted connections, `gnutls-cli` can be used (part of GnuTLS, which is free software)
    - TLS = Transport Layer Security
      - Successor to SSL
      - Layer between application layer and TCP, recall Internet architecture
      - Secure channels based on asymmetric cryptography

3.2.1 telnet

- Original purpose: Login to remote host (plaintext passwords)
  - Nowadays, we use Secure Shell, `ssh`, for that
- Still, `telnet` can establish **arbitrary TCP connections**
  - `telnet www.google.de 80`

    - (For variants without visual feedback possibly followed by ctrl-+ or ctrl-], `set localecho` [enter] [enter])
    - `GET / HTTP/1.1` [enter]
    - `Host:  www.google.de` [enter] [enter]
    - (Context for above lines soon)
  - `telnet wi.uni-muenster.de 25`
    - (SMTP)
  - **Beware**: Buggy telnet implementations may stop sending after first line (use Wireshark to verify)

3.2.2 gnutls-cli

- Establish TLS protected TCP connection
  - Alternative to `telnet` on previous slide
  - `gnutls-cli --crlf www.informationelle-selbstbestimmung-im-internet.de`
    - `GET /Anonymes_Surfen_mit_Tor.html HTTP/1.1` [enter]
    - `Host:  www.informationelle-selbstbestimmung-im-internet.de` [enter] [enter]
  - `gnutls-cli --crlf --starttls -p 25 wi.uni-muenster.de`
    - Type `ehlo localhost`, then `starttls`; press ctrl-d to enter TLS mode

3.3 Excursion: Browser Tools

- Modern browsers offer developer tools
  - E.g., press `Ctrl-Shift-I` with Firefox
  - Tools to inspect HTML, CSS, Javascript
  - Tools to inspect HTTP traffic (Network tab)
    - Live view on browser requests and server responses
      - With details on timing, caching, headers
  - Console with error messages
  - And much more

3.4 HTTP Messages

- Requests and responses
  - Generic message format of RFC 822, 1982 (822→2822→5322)
    - Originally for e-mail, extensions for binary data
      - Lines end with CRLF, `\r\n` below
  - Messages consist of
    - Headers
      - In HTTP always a distinguished start-line (request or status)

        - Then zero or more headers
      - Empty line
      - Optional message body
  - Sample `GET` **request** (does not have a body)
    - `GET /newsticker/ HTTP/1.1\r\n`
      `Host: www.heise.de\r\n`
      `User-Agent: Mozilla/5.0\r\n`
      `\r\n`

- Sample HTTP **response** to previous `GET` request
  - `HTTP/1.1 200 OK\r\n`
    `Date: Tue, 02 Nov 2010 13:49:26 GMT\r\n`
    `Server: Apache\r\n`
    `Vary: Accept-Encoding,User-Agent\r\n`
    `Content-Encoding: gzip\r\n`
    `Content-Length: 20046\r\n`
    `Connection: close\r\n`
    `Content-Type: text/html; charset=utf-8\r\n`
    `\r\n`
    `gzip'ed HTML code as body`

3.5 HTTP Methods

- Case-sensitive (capital letters)
  - `GET` (Request for resource, see section 4.3.1)
  - `HEAD` (Request information on resource, see section 4.3.2)
  - `POST` (Transfers entity, see section 4.3.3)
    - Annotations, postings, forms, database extensions
  - `PUT` (Creates new resource on server, see section 4.3.4)
  - `DELETE` (Deletes resource from server, see section 4.3.5)
  - `CONNECT` (Establish tunnel with proxy, see section 4.3.6)
  - `OPTIONS` (Asks for server capabilities, see section 4.3.7)
  - `TRACE` (Tracing of messages through proxies, see section 4.3.8)

3.6 Conditional GET

- `GET` under conditions
  - Requires (case-insensitive) request header
    - (Can be used by browser to check if cached version still fresh)
    - `If-Modified-Since`
    - `If-Match`
    - `If-None-Match`

- Example

- Request
  - ```
    GET /Anonymes_Surfen_mit_Tor.html HTTP/1.1
    Host: www.informationelle-selbstbestimmung-im-internet.de
    If-None-Match: "4fc5-568ed5e21e210"
    ```
- Response
  - ```
    HTTP/1.1 304 Not Modified
    Date: Mon, 16 Jul 2018 08:23:07 GMT
    additional headers
    ```

3.7 Sample Status Codes

- Three digits, first one for class of response
  - 1xx: Informational - Request received, continuing process
    - 100: Continue - Client may continue with request body
  - 2xx: Successful - Request successfully received, understood, and accepted
    - 200: OK
  - 3xx: Redirection - Further action necessary to complete request
    - 302: Found (temporarily under different URI)
    - 303: See Other (redirect to different URI in `Location` header)
    - 304: Not Modified (previous slide)
  - 4xx: Client Error - Request with bad syntax or cannot be fulfilled
    - 403: Forbidden
    - 404: Not Found
  - 5xx: Server Error - Server failed for apparently valid request

3.8 HTTP Connection Management

- Options

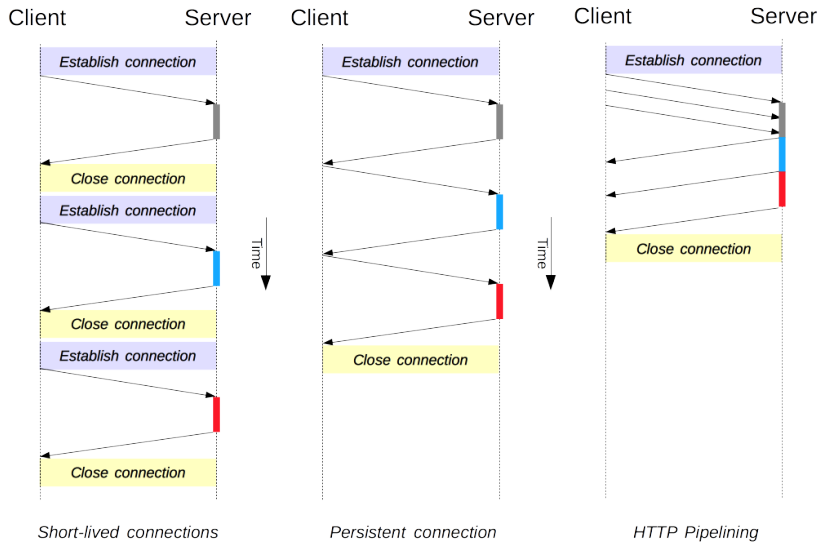

Figure 2: "HTTP/1.x connection management" by Mozilla Contributors under CC BY-SA 2.5; from MDN web docs

  - **Short-lived** connections: Each request on separate TCP connection
  - **Persistent** connections
    * TCP connection reused for multiple HTTP requests
      · HTTP/1.0: Connection: Keep-Alive
      · HTTP/1.1: Persistence by default
  - **Pipelined** connections
    * Client may send multiple requests over single TCP connection before receiving a response
    * HTTP/1.1 compliant servers support pipelining
      · Not activated in browsers by default, compatibility and performance problems

3.9 Review Question

- Did you execute `GET` requests and conditional `GET` requests on the command line? Any surprises?
- Note: Connections may **time out** (e.g., `Peer has closed the GnuTLS connection`) if you do not type fast enough
  - Also, use of backspace or cursor keys may destroy connection
  - Suggestion: Type in text editor and copy&paste into command line

4 Conclusions

4.1 Summary

- Web browsers and servers talk HTTP

- Simple message format
- More details in CACS

License Information