Application-Layer Protocols:
The Web & HTTP

Jasleen Kaur
January 29, 2019

Application-Layer Protocols
Outline

◆ Example client/server systems and their application-level protocols:
  » The World-Wide Web (HTTP)
  » Reliable file transfer (FTP)
  » E-mail (SMTP & POP)
  » Internet Domain Name System (DNS)
◆ Example p2p applications systems:
  » BitTorrent
◆ Other protocols and systems:
  » Streaming media — DASH
  » Content delivery networks (CDNs)
Applications and Application-Layer Protocols

Overview

◆ Applications:
  » Communicating, distributed processes
  » Running in network hosts in “user space”
  » Exchange messages

◆ Application-layer protocols
  » One “piece” of an application
  » Defines messages exchanged and actions taken
  » Uses services provided by lower layer protocols

Application-Layer Protocols
The Web

◆ User agent (client) for the Web is called a browser:
  » MS Internet Explorer
  » Google Chrome
  » Apple Safari
  » Mozilla Firefox

◆ Server for the Web is called a Web server:
  » Apache (public domain)
  » MS Internet Information Server (IIS)
Application-Layer Protocols
Web Terminology

◆ Web page:
  » Addressed by a URL
  » Consists of “objects”

◆ Most Web pages consist of:
  » Base HTML page
  » Embedded objects

Web Terminology
URLs (Universal Resource Locators)

Optional server port (Default = port 80)

www.someSchool.edu:8080/someDept/pic.gif

Server domain name
Object path name

◆ Each object on the web is addressable by a URL
  » RFC 1738 & 3986
◆ URL components
  » Server address
  » (Optional port number)
  » Path name
Web Terminology
The Hypertext Transfer Protocol (HTTP)

- Web’s application layer protocol
- Client/server model
  - client: browser that requests, receives, and displays Web objects
  - server: Web server sends objects in response to requests

The Hypertext Transfer Protocol
HTTP Overview

- HTTP uses TCP sockets
  - Browser initiates TCP connection to server (on port 80)
- HTTP messages (application-layer protocol messages) exchanged between browser and Web server
- HTTP/1.0: RFC 1945
  - One request/response interaction per connection
- HTTP/1.1: RFC 2616
  - Persistent connections
  - Pipelined connections
- HTTP/2.0: RFC 7540
  - Pipelined, encrypted by default

- HTTP is “stateless”
  - Server maintains no information about past browser requests

Aside
- Protocols that maintain “state” are complex!
  - Past history (state) must be maintained
  - If server or client crashes, their views of “state” may be inconsistent and must be reconciled
The Hypertext Transfer Protocol

HTTP example

- User enters URL `www.someSchool.edu/someDept/home.index`
  - Referenced object contains HTML text and references 10 JPEG images
- Browser sends an HTTP “GET” request to the server `www.someSchool.edu`
- Server will retrieve and send the HTML file
- Browser will read the file and sequentially make 10 separate requests for the embedded JPEG images

HTTP 1.0 Example

URL `www.someschool.edu/someDept/home.index`

0) Server process at host `www.someschool.edu` waiting for TCP connections on port 80

1) Browser initiates TCP connection to server at `www.someSchool.edu`
   - Port 80 is "well known" for server

2) Server "accepts" connection

3) Client writes an HTTP GET request message (containing path) to TCP connection socket

4) Server reads request message, forms response message containing requested object, writes message to socket

5) Server closes TCP connection
HTTP 1.0 Example
URL www.someschool.edu/someDept/home.index

6) Browser reads response message containing the HTML file. Ten references to JPEG objects are found during the HTML parse.

7) Browser initiates TCP connection to server at www.someschool.edu

TCP 3-way handshake

8) Server "accepts" connection

Steps 1-6 are repeated for each of the 10 JPEG objects

The Hypertext Transfer Protocol
HTTP message format

- Two types of HTTP message formats: request and response messages
  - ASCII (human-readable format)

- HTTP request message:
  - Request line
  - Optional header lines
    - Present only for some methods (e.g., POST)
  - Entity body

HTTP 1.0 Example
URL www.someschool.edu/someDept/home.index

6) Browser reads response message containing the HTML file. Ten references to JPEG objects are found during the HTML parse.

7) Browser initiates TCP connection to server at www.someschool.edu

TCP 3-way handshake

8) Server "accepts" connection

Steps 1-6 are repeated for each of the 10 JPEG objects

The Hypertext Transfer Protocol
HTTP message format

- Two types of HTTP message formats: request and response messages
  - ASCII (human-readable format)

- HTTP request message:
  - Request line
  - Optional header lines
    - Present only for some methods (e.g., POST)
  - Entity body
The Hypertext Transfer Protocol
HTTP message format

- Method types:
  - GET
  - POST — Form input sent in entity body to path specified in URL field
  - HEAD — Just headers, no object
  - DELETE — Delete file specified in URL field

- HTTP/1.1 adds
  - POST — Uploads in entity body to path specified in URL field

- HTTP Message Format
  - Mozilla & MS Explorer request examples

  HTTP/1.1 adds
  - POST — Uploads in entity body to path specified in URL field
  - DELETE — Delete file specified in URL field

How does Mozilla process:

GET /~jasleen HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.74 [en] (WinNT, U)
Host: dove.cs.unc.edu:80
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, 
       */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: SITESERVER=ID=8a064b7855a043146e45991174a3d970

- List of standard header fields:

- Chrome -> Inspect -> Network -> Headers
HTTP Message Format
Mozilla & MS Explorer request examples

GET /~jasleen HTTP/1.0
Connection: Keep-Alive
User-Agent: Mozilla/4.74 [en] (WinNT; U)
Host: dove.cs.unc.edu:80
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
image/png, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Cookie: SITESERVER=ID=8a064b7855a043146e45991174a3d970

GET /~jasleen HTTP/1.1
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/msword, application/vnd.ms-excel,
application/vnd.ms-powerpoint, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 4.0)
Host: dove.cs.unc.edu:80
Connection: Keep-Alive

HTTP Message Format
General response message format

◆ Response messages
  » ASCII (human-readable format)

◆ Message structure:
  » Response line
  » Optional header lines
  » Requested object, error message, etc.

version <SP> code <SP> phrase <CR><LF>
header field name "." value <CR><LF>

entity body
HTTP Message Format
HTTP response status codes

- Sample response codes:
  - **200 OK**
    - Request succeeded, requested object later in this message
  - **301 Moved Permanently**
    - Requested object moved, new location specified later in this message (Location:)
  - **400 Bad Request**
    - Request message not understood by server
  - **404 Not Found**
    - Requested document not found on this server
  - **505 HTTP Version Not Supported**

Typical Response Headers

```
http://example.com
```

- **Connection**: Keep-Alive
- **User-Agent**: Mozilla/4.74 [en] (WinNT; U)
- **Host**: dove.cs.unc.edu:80
- **Accept**: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*
- **Accept-Encoding**: gzip
- **Accept-Language**: en
- **Accept-Charset**: iso-8859-1,*,utf-8
- **Cookie**: SITESERVER=ID=8a064b785a043146e4599174a3d97

- **Date**: Fri, 02 Feb 2009 19:10:11 GMT
- **Server**: Apache/1.3.9 (Unix) (Red Hat/Linux)
- **Last-Modified**: Tue, 30 Jan 2009 21:48:14 GMT
- **Etag**: "1807135e-67-3a77369e"
- **Accept-Ranges**: bytes
- **Content-Length**: 103
- **Connection**: close
- **Content-Type**: text/plain
**HTTP Message Format**

**Telnet example**

| Connect to HTTP server port | > telnet www.cs.unc.edu 80 |
|———|——|
| Telnet output | Trying 152.2.131.240... |
| | Connected to rock.cs.unc.edu. |
| Type GET command plus blank line | Escape character is '{ }'. |
| HTTP response status line | GET /~jasleen/foo.txt HTTP/1.0 |
| | HTTP/1.1 200 OK |
| | Date: Tue, 12 Feb 2013 01:43:25 GMT |
| | Server: Apache/2.2.15 (Red Hat) |
| | Last-Modified: Tue, 12 Feb 2013 01:43:02 GMT |
| | ETag: "44d2e7a0-6e-4d57d25675580" |
| HTTP response headers plus blank line | Accept-Ranges: bytes |
| | Content-Length: 110 |
| | Connection: close |
| | Content-Type: text/plain; charset=UTF-8 |
| | ** This test file is stored in the UNIX file system at ** |
| | ** /afs/cs.unc.edu/home/jasleen/public_html/foo.txt ** |
| Telnet output | Connection closed by foreign host. |

**Telnet example (2)**

| Connect to HTTP server port | > telnet www.msn.com 80 |
|———|——|
| Telnet output | Trying 207.46.179.134... |
| Type GET command plus blank line | Connected to www.msn.com. |
| Escape character is '{ }'. |
| HTTP response status line | GET /~index.html HTTP/1.0 |
| | HTTP/1.1 404 Object Not Found |
| | Server: Microsoft-IIS/5.0 |
| | Date: Mon, 11 Feb 2002 18:33:15 GMT |
| HTTP response headers plus blank line | Content-Length: 1638 |
| | Content-Type: text/html |
| Object content | <HTML> <HEAD>... |
| | ... |
| | Error type 404 - Object Not Found |
| | </body> </html> |
| Telnet output | Connection closed by foreign host. |