

CS290i - Lecture 3

HTTP

Scalable Internet Services and Systems, Spring 2001

Thorsten von Eicken
Department of Computer Science
University of California at Santa Barbara

HTTP 0.9

† The Original HTTP as defined in 1991

- † This document defines the Hypertext Transfer protocol (HTTP) as originally implemented by the World Wide Web initiative software in the prototype released. This is a subset of the full HTTP protocol, and is known as HTTP 0.9.
- † No client profile information is transferred with the query. Future HTTP protocols will be back-compatible with this protocol.
- † This restricted protocol is very simple and may always be used when you do not need the capabilities of the full protocol which is backwards compatible.
- † The definition of this protocol is in the public domain (see policy).
- † The protocol uses the normal internet-style telnet protocol style on a TCP-IP link.

2

HTTP 0.9 (cont)

† Connection

- † The client makes a TCP-IP connection to the host using the [domain name](#) or [IP number](#), and the [port number](#) given in the address.
- † If the port number is not specified, 80 is always assumed for HTTP.
- † The server accepts the connection.
- † Note: HTTP currently runs over TCP, but could run over any connection-oriented service. The interpretation of the protocol below in the case of a sequenced packet service (such as DECnet(TM) or ISO TP4) is that the request should be one TPDU, but the response may be many.

3

HTTP 0.9 (cont.)

† Request

- † A line of ASCII chars terminated by a CR LF pair.
- † A well-behaved server will not require the CR.
- † Request: "GET", a space, the document address.
- † The document address will consist of a single word (ie no spaces)
- † The search functionality of the protocol lies in the ability of the addressing syntax to describe a search on a named index .
- † A search should only be requested by a client when the index document itself has been described as an index using the ISINDEX tag.

4

HTTP 0.9 (cont.)

† Response

- † The response is a message in HTML. This is a byte stream of ASCII characters.
- † Lines shall be delimited by an optional CR followed by a mandatory LF.
- † The format of the message is HTML. It also allows for plain ASCII text to be returned following the PLAINTEXT tag .
- † The message is terminated by the closing of the connection by the server.
- † Well-behaved clients will read the entire document as fast as possible. The server may impose a timeout of the order of 15 seconds on inactivity.
- † Error responses are supplied in human readable text in HTML syntax. There is no way to distinguish an error response from a satisfactory response except for the content of the text.

5

HTTP 0.9 (cont.)

† Disconnection

- † The TCP-IP connection is broken by the server when the whole document has been transferred.
- † The client may abort the transfer by breaking the connection before this, in which case the server shall not record any error condition.
- † Requests are idempotent. The server need not store any information about the request after disconnection.

6

HTTP 1.0

† URL = Uniform Resource Locator

- † Resource ? file

† Format of requests and responses

- † Initial request/reponse line
- † Zero or more header lines
- † Blank line (CRLF)
- † Optional message body

† Initial request line:

- † GET /path/to/file/index.html HTTP/1.0
- † Other requests: POST, HEAD

7

HTTP 1.0 (cont.)

† Initial response line:

- † HTTP/1.0 <code> <text>
- † Code is error code:
 - † 1xx: informational
 - † 2xx: success, e.g. 200 OK
 - † 3xx: redirect, e.g. 301 Moved Permanently, 302 Moved Temporarily
 - † 4xx: error by client, e.g. 404 Not Found
 - † 5xx: error by server, e.g. 500 Server error

† Headers:

- † <header-name>: <value>, <value>, ...
- † Example:
Header1: some-value-1a, some-value-1b
HEADER1: some-value-1a,
some-value-1b

8

HTTP 1.0 (cont.)

† Message body

- † The `Content-Type`: header gives the MIME-type of the data in the body, such as `text/html` or `image/gif`.
- † The `Content-Length`: header gives the number of bytes in the body

† HEAD request

- † Returns same headers as GET, but no message body

† POST request

- † Send data to the server: message body
- † Content-length & Content-type headers

9

HTTP 1.1

† Timeline

- † Nov 95: first internet draft (before HTTP/1.0 draft)
- † Jan 97: RFC2068 finished
- † Nov 98: RFC2616 “draft standard” approved

† New features

- † Persistent connections
- † Caching/proxy cleanup
- † IP address conservation (Host header)
- † Partial transfers
- † Content negotiation
- † Compression
- † Digest authentication

10

New Features

† Persistent connections

† IP address conservation

† Partial transfers

† Compression

† Digest authentication

11

Persistent Connections

- † Connections are persistent by default:
 - † Send request 1, receive response 1
 - † Send request 2, receive response 2
 - † Note: requires well-delimited requests & responses (content-length & chunked encoding)
- † Termination:
 - † Server: `Connection: close`, or close connection (e.g. idle)
 - † Client: close connection (or `Connection: close`)
 - † Note: client must handle aborted connections by retrying
- † Pipelining:
 - † send request 2 before response 1 rec'd
 - † Strictly in-order responses
 - † Careful with non-idempotent requests

12

Host+ Range header

† Host header

† Problem: 1 server but 1000 domain names (www.thorsten.com)

† Solution: host header is required

‡ GET /index.html HTTP/1.1

Host: www.thorsten.com

‡ As opposed to assigning 1000 IP addresses to the server

† Range header

† Problem: aborted connections, fetching prefixes

† Solution: specify requested byte range

‡ GET /bigimage.jpg HTTP/1.1

Range: bytes=12345-

13

Jeff Mogul's gripes

† Data model, MIME miasma

† Extensibility

† Caching

† Header categories

† Status and error codes

† Transport issues

† Content negotiation

† Cookies & social issues

14

Data Model & MIME

† See Mogul's slides 15-24

† #16: HTTP-OO "vision": bogus from the start...

† #18: "Entity": **1 a** : independent, separate, or self-contained existence **b** : the existence of a thing as contrasted with its attributes; **2** : something that has separate and distinct existence and objective or conceptual reality

† #23: Example:

‡ What is the server supposed to respond to a request with:

† Content-Type: text/plain

† Content-Encoding: compress

† Transfer-Encoding: gzip, chunked

† Range: bytes=1024-2047

15

HTTP types/codings

† §3.5: "Content coding values indicate an encoding transformation that has been applied to an entity. Content codings are primarily used to allow a document to be compressed or otherwise usefully transformed without losing the identity of its underlying media type and without loss of information. Frequently, the entity is stored in coded form, transmitted directly, and only decoded by the recipient."

† §3.6: "Transfer-coding values are used to indicate an encoding transformation that has been applied to an entity-body in order to ensure "safe transport" through the network. This differs from a content coding in that the transfer-coding is a property of the message, not of the original entity."

† §7.2.1: "When an entity-body is included with a message, the data type of that body is determined via the header fields Content-Type and Content-Encoding. These define a two-layer, ordered encoding model:
entity-body := Content-Encoding(Content-Type(data))
Content-Type specifies the media type of the underlying data. Content-Encoding may be used to indicate any additional content codings applied to the data, usually for the purpose of data compression, that are a property of the requested resource."

† §14.35.1: "Byte range specifications in HTTP apply to the sequence of bytes in the entity-body (not necessarily the same as the message-body)."

16

HTTP types/codings (cont.)

† Questions:

- † In the following scenarios, what should your server send and what headers should it use?
- † 1: request to transfer bytes 1024-2047 of a file called foo.html.gz
- † 2: request to transfer bytes 1024-2047 of a file called foo.html and the client indicates it can handle a gzip transfer-coding
- † 3: request to transfer bytes 1024-2047 of foo.html.gz when only foo.html exists, but your server is “smart” and can gzip on the fly
- † 4: request to transfer bytes 1024-2047 of foo.gif when only foo.jpg exists, but your server is “smart” and can transform on the fly

17

Extensibility

† See Mogul's slides 25-29

- † #26: Multi-version chains are possible:
HTTP/1.0 200 OK
Via: 1.1 fred, 1.0 Lucy
- † #26: for example byte ranges are optional
- † #29: when to GET vs. POST?

18

Caching

† Cache locations:

- † Server
- † Public proxies
- † Private proxies
- † User-agent (browser)

† See Mogul's slides 30-36

- † #33: e.g. can //www.expertcity.com/index.html invalidate //63.251.224.189/index.html?
- † Age calculation, expiry date, last-modified date, “max-age=3600”, if-modified-since
- † Variants: request “Accept-Language: fr, en;q=0.5”, got en-br version in cache, can it be returned?

19

Other

† Content negotiation

- † See Mogul's slides 37-38
 - † Request headers: Accept[-type], Accept-Charset, Accept-Encoding, Accept-Language, Upgrade
 - † Accept-Encoding applies to Content-Encoding, not Transfer-Encoding! (TE is for latter, so what's the point?)
 - † Response headers: Accept-Ranges, Connection

† Status codes

- † See Mogul's slides 39-40

† Transport issues

- † See Mogul's slides 41-46
 - † #45: max size of a URI? Of headers? Failure indication?

20

Cookies & social issues

† See Mogul's slides 47-48

† Cookies not part of HTTP/1.1! See RFC2965...

† Headers: Set-Cookie2, Cookie

† Fields:

- † Domain=.expertcity.com
- † Path=/myaccount/
- † Port="80,8080"
- † Discard (when u-a terminates)
- † Max-age=3600
- † Secure

21

Summary

† Why is it so hard?

- † Each detail is simple
- † The collection is unmanagable
- † Need more layers? More orthogonality?

† How can you implement it?

- † What do you expect clients to implement?
- † What do you expect servers to implement?
- † What do you expect proxies to implement?
- † What do you expect all these to get wrong?
- † How do you test your own implementation?

† How come it works at all?

- † Nobody uses the full spec?
- † The real spec is IE & NS?

25