

HTTP

HTTP

The most important protocol of the Internet – it **is** the Internet at present.

Not radical technology at all, just collects some good ideas. However, it created tremendous network effect n^2 value. (the fact that one guy working part time did it suggests that the technology itself is not the key part of the story.)

Tim Berners-Lee working at CERN – created a document distribution protocol, the Hypertext Transfer Protocol.

"Hypertext" was an well established notion of having links to a document embedded in other documents. See the "Xanadu" project – www.xanadu.net – (1960) an example of being so ambitious, that the more ordinary implementation (HTTP) just passes you by. Network effect then makes the widespread (HTTP) dominate, regardless of technical merit.

From the original HTTP justification draft

(<http://www.w3.org/Protocols/WhyHTTP.html>)...

Why a new protocol?

Existing protocols cover a number of different tasks.

- * Mail protocols allow the transfer of transient messages from a single author to a small number of recipients, at the request of the author.
- * File transfer protocols allow the transfer of data at the request of either the sender or receiver, but allow little processing of the data at the responding side.
- * News protocols allow the broadcast of transient data to a wide audience.
- * Search and Retrieve protocols allow index searches to be made, and allow document access. Few exist: Z39.50 is one and could be extended for our needs.

The protocol we need for information access (HTTP) must provide

- * A subset of the file transfer functionality
- * The ability to request an index search
- * Automatic format negotiation.
- * The ability to refer the client to another server

Tim Berners-Lee (1991)

See...

<http://www.w3.org/Protocols/>

<http://www.w3.org/Protocols/rfc2616/rfc2616.txt>

HTTP Client/Server Picture

We will start with a top-level view of HTTP, and zoom in for more detail in later lectures.

HTTP Client / Browser

Has URL → makes *request* to server

Gets back data (*response*)

Displays that data (HTML, JPEG, ...)

Processes, clicks, etc. to make further requests

Typical client side: the display of the data, the processing of clicks, scrolling, typing, etc. is handled on the client side so it's nice and fast.

URL

Identifies a document available through http

e.g. `http://www.stanford.edu/class/cs193i/index.html`

Protocol aka "scheme"

`http`

Host - server to contact

`www.stanford.edu`

Path - identifies document on that host

`/class/cs193i/index.html`

URN, URI

Future directions - URLs identify a document, but they depend on the name of the server and other details of where the document is stored. URNs identify a document regardless of where it is stored.

For the present, we have lots of URLs and few URNs.

URI - the catch-all category of URLs + URNs.

HTTP Server

Listens on Port 80

Gets HTTP requests, sends back responses

In the simplest case, the connection is closed at the end of a single request/response pair.

Document Root on Server

Usually, the HTTP server has a "document root" that points to the directory in the filesystem that contains the documents to serve up.

Suppose `/etc/htdocs/` is the document root for an HTTP server.

(htdocs is short for hyper-text documents)

There are subdirectories `a`, `b`, and `c`: (`"/etc/htdocs/a/"`, `"/etc/htdocs/b/"`, `"/etc/htdocs/c/"`), and each of those contains an `"x.html"` file.

The server gets request paths like `/a/x.html` and `/b/x.html`, and maps them into the filesystem.

e.g. the request path `/a/x.html` maps into the filesystem document
`/etc/htdocs/a/x.html`

Client Request

The client looks at the URL, contacts that host, and sends a short request that specifies the desired the path.

Request Example

So for the URL `http://foo.com/a/x.html`

Connect to `foo.com` on port 80

The request is of the form `"GET path HTTP/1.0\r\n\r\n"` where the path is the desired path from the URL

e.g. `GET /a/x.html HTTP/1.0\r\n\r\n`

Server Response

HTTP: response from server back to client

Using doc root, maps the path into the local file system...

`/class/cs193i/index.html` → `/usr/class/cs193i/WWW/index.html`

The server reads that file and sends it back to the client as the HTTP response

In the simplest case, the connection then closes after the single request/response transaction.