ECE 435 – Network Engineering
Lecture 20

Vince Weaver

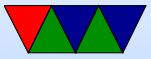
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

16 November 2016

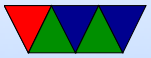
Announcements

- HW#8 Due, HW#9 will be posted
- Forgot to mention SMTP servers. Sendmail. Qmail. Postix. Exim.
- Microsoft becomes member of Linux Foundation(!)
- Don't forget Project status update



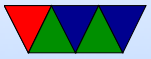
www on the client side

- A URL (uniform resource locator) specifies the document you want
  `http://web.eece.maine.edu/~vweaver/`
- Browser parses URL
- It looks up the address of web.eece.maine.edu via DNS
- DNS reports 192.168.8.99
- Browser makes TCP connection to port 80 (default) of 192.168.8.99
  how do you specify other port? web.eece.maine.edu:8080



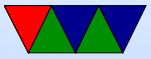
why would you want to?
security

- The client requests the file vweaver/index.html (index.html is the default). Often there's a server root, and special handling for user dirs (with the tilde) that are often in user home dirs, as in this case
- The server returns this file
- The TCP connection is closed
- The browser displays the text
- The browser might have to fetch any images linked to by the document



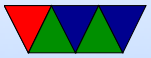
Non-HTML

- You can serve up any kind of binary file. Often have associated MIME-type like with e-mail
- Browser also often has built in support
- GIF (trouble due to patents), PNG. SVG?
- MP3 music? Movies?
- Plugins. Flash? Java? PDF?



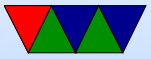
Web-servers

- famously netcraft had a list (meme netcraft reports BSD is dying)
- NCSA was first popular one
- Apache ("a patchy" version of NCSA) took over
- Microsoft IIS
- Other companies like Sun/Netscape/SGI
- nginx ("engine-x")
- lighthttpd ("lightly")

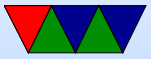


simple web server

- Listen on port 80
- Accept a TCP connection
- Get name of file requested
- Read file from disk
- Return to client
- Release TCP connection
- How do we make this faster?
  - Cache things so not limited by disk
    (also cache in browser so not limited by network)



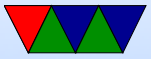
- Make server multithreaded



URLs

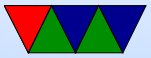
- URI (uniform resource identifier)
- URL (uniform resource locator) subset of URI, includes info on how to find the resource (protocol and server)

- URN (uniform resource name) asks for a document but from anywhere. I.e. give it something like an ISBN and returns the book
- `scheme:[//[user:password@]host[:port]][/]path[?query][#fragment]`
  `: / ? # [ ] @ reserved, must encode if use %3f`
  `query key1=value1&key2=value2 or key1=value1;key2=value2`
- protocol: http, ftp, file, news, gopher, mailto, telnet



http

- HyperText Transfer Protocol (RFC 2616)
- Make ASCII request, get a MIME-like response
- HTTP 1.0, single request was set and single response
  HTTP 1.1 supports persistent connections, allowing multiple requests to happen with one TCP connection (lowering overhead)
- Commands
  - GET filename HTTP/1.1
    get file



- HEAD

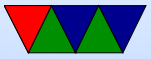
  get header (can check timestamp. why? see if cache up to date)
- PUT

  send a file
- POST

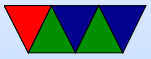
  append to a file
- DELETE

  remove file (not used much)
- TRACE

  debugging

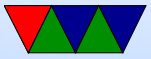


  - CONNECT, OPTIONS
- Responses are three digit status codes
  - 1xx – informational – not used much
  - 2xx – Success – 200 = page is OK
  - 3xx – Redirect – 303 = page moved
  - 4xx – Client Error – 403 = forbidden, 404 = not found
  - 5xx – Server ERror – 500 = internal, 503 = try again
- Additional request headers. A lot are possible, included after the GET.
  - User-Agent (browser info). Can you lie? Can you leak info?

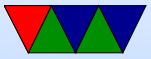


- Accept-*: type of documents can accept, compression, character set
- Host: server you are requesting
  Can configure browser to open up helper util for this (for example, run Office if it's a word file)
- Authorization: if you need special permissions/login
- Cookie: deals with cookies
  Statelessness – how do you remember setting, logins, shopping cart, etc. "cookies". Expire. Can be misused.

- Additional response headers.
  - Content-Encoding,Language,Length,Type

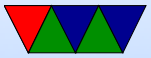


- Last-Modified: helps with caching
- Location: used when redirecting
- Accept-Ranges: partial downloads (downloading a large file, interrupted, can restart where left off)



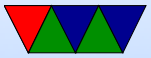
Do you need a browser?

```
telnet www.maine.edu 80
GET / HTTP/1.1
Host: www.maine.edu
control-[
close
```



HTTP/2

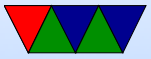
- 2015. RFC 7540
- Google push through, extension of their SPDY (speedy) Microsoft and Facebook giving feedback
- Leaves a lot of high level things the same
- Decrease latency of rendering web pages:
  - compress headers
  - Server can push data the browser didn't request yet but it knows it will need (like images, etc)
  - pipeline requests



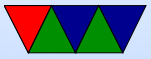
Send multiple requests without waiting for response
good on high-latency links (FIFO on 1.1, new makes it asynchronous)

- multiplex multiple requests over one TCP connection
- head-of-line blocking problem?
  line of packets held up by processing of first
  FIFO first requests waits until done until next, can't run in parallel

- Page load time 10-50% faster
- While can use w/o encryption, most browsers say will only do with encryption

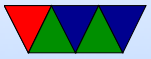


- Criticism: was rushed through. Is way complex. Does own flow control (has own TCP inside of TCP) Re-implements transport layer at application layer



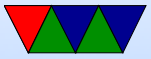
What if Server Overloaded?

- Slashdot effect
- caching/proxy – squid
- Content Delivery Network – akami
- Server farms



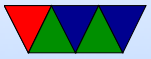
Security

- SSL – Secure Socket Layer
- Replaced by TLS (Transport Layer Security)
- Port 443 for https
- Public key encryption.



Setting Up a Web-server

- Apache



Web Seach

- Web-bots index the web. robots.txt file
- Altavista, Hotbot, Excite, Inktomi, etc.
- Curated search like Yahoo (people organize links rather than automatically search)
- Google (1996 some machine in Stanford, 1997-1998)
- MSN search 1999, rebranded Microsoft Bing 2009