

# HTTP 5

---

## HTTP 1.1 Features

(see previous handout)

Keep-alive connection

Connection: keep-alive -- in request (it's the default anyway)

Connection: close

Multiple requests "in flight" at a time

Virtual hosts -- "host:cse.stanford.edu" header in request to indicate intended server

Chunked transfer encoding

Another speedup

Send the document back in chunks. Each chunk is preceded by its size (in hex) followed by \r\n (CRLF), followed by the data

There is no content-length header, so the server can start sending data as soon as it has anything

The end of the document is marked by a length=0 chunk.

## HTTP 1.1 Pipelined Example

Here I have hacked up the web client to request /0.html, /1.html, .. in very quick succession. Only 0.html actually exists, the others generate 404.s

Demonstrates multiple, pipelined requests on a keep-alive connection

Notice that the 404 responses come back chunked (when to chunk and when not is up to the server)

## Pipelined Client Code

```
## ask for /0.html /1.html ... without waiting for response
## -- demonstrates keep-alive / pipelining
my($i, $req, $count);
$count = 5;
for ($i=0; $i<$count; $i++) {
    $req = "GET /$i.html HTTP/1.1$EOL";
    $req .= "host:$host$EOL";

    ## Ask for keep-alive, except the last time
    ## (in reality, we should check the server response
    ## to see if it is actually keeping the connection open)
    my($mode);
    if ($i==$count-1) { $mode = "close"; } else { $mode="keep-alive"; }
    $req .= "connection:$mode$EOL";

    $req .= "$EOL";

    print SOCK $req;
```

```

    print $req;
}

## Read back and print all the responses

my($line);

while ($line = <SOCK>) {
    $line =~ s/\015|\012//g; ## strip off line endings
    print $line, "\n"; ## print with our local line ending
}

close(SOCK);

exit(0);

```

## Pipelined Example

```

elaine0:~/my193i> webclient2.pl cse.stanford.edu
GET /0.html HTTP/1.1
host:cse.stanford.edu
connection:keep-alive

GET /1.html HTTP/1.1
host:cse.stanford.edu
connection:keep-alive

GET /2.html HTTP/1.1
host:cse.stanford.edu
connection:keep-alive

GET /3.html HTTP/1.1
host:cse.stanford.edu
connection:keep-alive

GET /4.html HTTP/1.1
host:cse.stanford.edu
connection:close

HTTP/1.1 200 OK
Date: Fri, 26 Apr 2002 18:47:15 GMT
Server: Apache/1.3.23 (Darwin)
Last-Modified: Fri, 26 Apr 2002 18:40:53 GMT
ETag: "115bb-dc-3cc99f35"
Accept-Ranges: bytes
Content-Length: 220
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html

<!doctype html public "-//w3c//dtd html 4.0 transitional//en">
<html>
<head>
    <title>Test</title>
</head>

```

```
<body bgcolor="#FFFFFF">
```

```
<h1>
```

```
Test</h1>
```

```
<p>Just a little test doc.
```

```
<p>Just a little test doc.
```

```
</body>
```

```
</html>
```

```
HTTP/1.1 404 Not Found
```

```
Date: Fri, 26 Apr 2002 18:47:15 GMT
```

```
Server: Apache/1.3.23 (Darwin)
```

```
Keep-Alive: timeout=15, max=99
```

```
Connection: Keep-Alive
```

```
Transfer-Encoding: chunked
```

```
Content-Type: text/html; charset=iso-8859-1
```

```
111
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

```
<HTML><HEAD>
```

```
<TITLE>404 Not Found</TITLE>
```

```
</HEAD><BODY>
```

```
<H1>Not Found</H1>
```

```
The requested URL /1.html was not found on this server.<P>
```

```
<HR>
```

```
<ADDRESS>Apache/1.3.23 Server at cse.stanford.edu Port 80</ADDRESS>
```

```
</BODY></HTML>
```

```
0
```

```
...
```

```
111
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

```
<HTML><HEAD>
```

```
<TITLE>404 Not Found</TITLE>
```

```
</HEAD><BODY>
```

```
<H1>Not Found</H1>
```

```
The requested URL /4.html was not found on this server.<P>
```

```
<HR>
```

```
<ADDRESS>Apache/1.3.23 Server at cse.stanford.edu Port 80</ADDRESS>
```

```
</BODY></HTML>
```

```
0
```

## Client Side

Write a little "echo" server that listens for HTTP requests on port 8181, and then just echos it back, so we can see what our browser is doing...

## Echo Server Code

```
## Wait for incoming client connections
my $client_addr;
while ($client_addr = accept(CLIENT, SERVER)) {           ## accept() blocks

    autoflush CLIENT, 1;

    ## Read the client request ... until blank line
    my($line, $request);
    $request = "";
    while (1) {
        $line = <CLIENT>;
        $line =~ s/\015|\012//g;
        $request = $request . $line . "\n";
        if ($line eq "") { last; }
    }

    print "Got from the client: $request\n";

    ## Spit back a minimal HTTP response, echoing the client response back to them...
    ## We don't put in content-length, but the client seems to deal ok
    ## To generate proper HTML, we would need to worry about characters
    ## like < and & that we dump out
    print CLIENT "HTTP/1.0 200 OK$EOL";
    print CLIENT "Content-type: text/html$EOL";
    print CLIENT "Server: home grown$EOL";
    print CLIENT "Connection: close$EOL";
    print CLIENT "$EOL";
    print CLIENT "<html>Here's what we got from the client:\n<pre>\n";
    print CLIENT $request;
    print CLIENT "</pre></html>\n";

    close(CLIENT);
}
```

## Echo Server Demo

Run echo server ... click on link in course page

<li>A <a href="http://elaine0.stanford.edu:8181/foo/bar.html?pi=3.15#fragment">echo link</a> (only works when the echo server is running)

```
elaine0:~/my193i> webecho.pl
Got from the client: GET /foo/bar.html?pi=3.15 HTTP/1.0
Referer: http://www.stanford.edu/class/cs193i/
Connection: Keep-Alive
User-Agent: Mozilla/4.79 (Macintosh; U; PPC)
Host: elaine0.stanford.edu:8181
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: SUNetCookieBrowser=TRUE; SITESERVER=ID=3af7a299a1be81a423146cc31ff92448

## Client Request Features

### User-Agent

The browser version and OS of the client  
 Can use to special-case the HTML to work around browser bugs -- generally a bad idea. Better to write out correct HTML, and let incorrect browsers fail.

### Referer

The page that contained the link that lead to this request  
 Allows you to see who is pointing to you  
 See google example below  
 Note: referer is misspelled in the spec, but we're stuck with it (note: inertia)

### Accept

MIME types the client can handle -- image/png is there, in this case, since my browser has a plug-in that allows it to handle png  
 These are just suggestions -- the server can send whatever MIME type, and the client will do the best it can.

### Accept-Encoding

The client indicates that they can deal with the gzip encoding -- HTML compresses really well, so sending the HTML gzip compressed can be a 50% bandwidth savings, and gives better performance as well.  
 The server will indicate the encoding in the response header.

## Web Log Example

Major fields: IP addr, request, result code, length, referer  
 Notice: see the IP addr and time of the request  
 Notice: request for HTML, followed by req for GIF in that page  
 Notice: IE bug where it requests the PDF three times in a row  
 Notice: the google referer  
 Notice: the favicon.ico -- the little icon for bookmarking

```
128.205.181.102 - - [26/Apr/2002:11:58:55 -0700] "GET / HTTP/1.1" 200 6964
"http://www.cse.buffalo.edu/~alphonce/Courses/Spring2002/cse116/resources.html"
128.205.181.102 - - [26/Apr/2002:11:58:56 -0700] "GET /smallbinky.gif HTTP/1.1" 200
7648 "http://cslibrary.stanford.edu/"
128.205.181.102 - - [26/Apr/2002:11:59:09 -0700] "GET /112/ HTTP/1.1" 200 2297
"http://cslibrary.stanford.edu/"
128.205.181.102 - - [26/Apr/2002:11:59:09 -0700] "GET /112/screen.jpg HTTP/1.1" 200
20070 "http://cslibrary.stanford.edu/112/"
141.153.141.5 - - [26/Apr/2002:11:59:36 -0700] "GET /103/ HTTP/1.1" 200 2544
"http://cslibrary.stanford.edu/"
```

```

141.153.141.5 - - [26/Apr/2002:11:59:48 -0700] "GET /103/LinkedListBasics.pdf
HTTP/1.1" 200 32768 "http://cslibrary.stanford.edu/103/"
141.153.141.5 - - [26/Apr/2002:11:59:49 -0700] "GET /103/LinkedListBasics.pdf
HTTP/1.1" 206 45556 "-"
141.153.141.5 - - [26/Apr/2002:11:59:51 -0700] "GET /103/LinkedListBasics.pdf
HTTP/1.1" 206 17888 "-"
128.12.106.149 - - [26/Apr/2002:11:59:53 -0700] "GET /108/EssentialPerl.pdf
HTTP/1.1" 304 - "http://smi.stanford.edu/projects/helix/bmi214/"
128.205.181.102 - - [26/Apr/2002:11:59:59 -0700] "GET /112/JTetris.jar HTTP/1.1" 200
31498 "http://cslibrary.stanford.edu/112/"
128.205.181.102 - - [26/Apr/2002:12:00:52 -0700] "GET /112/Readme.txt HTTP/1.1" 200
5090 "http://cslibrary.stanford.edu/112/"
143.166.99.177 - - [26/Apr/2002:12:01:01 -0700] "GET /105/ HTTP/1.0" 200 3212
"http://www.google.com/search?hl=en&q=linked+list"
209.49.118.20 - - [26/Apr/2002:12:01:02 -0700] "GET /105/ HTTP/1.0" 200 3212
"http://learnfree.stanford.edu/103/"
143.166.99.177 - - [26/Apr/2002:12:01:11 -0700] "GET /105/LinkedListProblems.pdf
HTTP/1.0" 200 32768 "http://cslibrary.stanford.edu/105/"
128.205.181.102 - - [26/Apr/2002:12:01:23 -0700] "GET /112/ HTTP/1.1" 304 -
"http://cslibrary.stanford.edu/"
128.205.181.102 - - [26/Apr/2002:12:01:23 -0700] "GET /112/screen.jpg HTTP/1.1" 304
- "http://cslibrary.stanford.edu/112/"
141.153.141.5 - - [26/Apr/2002:12:02:19 -0700] "GET /favicon.ico HTTP/1.1" 404 296
"-
129.110.47.188 - - [26/Apr/2002:12:04:23 -0700] "GET / HTTP/1.1" 200 6964
"http://www.google.com/search?hl=en&q=linked+list+uses"
129.110.47.188 - - [26/Apr/2002:12:04:24 -0700] "GET /smallbinky.gif HTTP/1.1" 200
7648 "http://cslibrary.stanford.edu/"
129.110.47.188 - - [26/Apr/2002:12:04:37 -0700] "GET /103/ HTTP/1.1" 200 2544
"http://cslibrary.stanford.edu/"
129.110.47.188 - - [26/Apr/2002:12:04:41 -0700] "GET /103/LinkedListBasics.pdf
HTTP/1.1" 200 46681 "http://cslibrary.stanford.edu/103/"
129.110.47.188 - - [26/Apr/2002:12:04:42 -0700] "GET /103/LinkedListBasics.pdf
HTTP/1.1" 206 32768 "-"
129.110.47.188 - - [26/Apr/2002:12:04:43 -0700] "GET /103/LinkedListBasics.pdf
HTTP/1.1" 206 37760 "-"

```

## Caching

Cache data closer to the client -- potential significant savings in overall Internet bandwidth.

## Browser Cache

Browser cache -- keep copies of HTML, GIV, ... seen recently

Back button...

Could just show the cached content, or could do an if-modified-since request...

## If-Modified-Since request

Use this field in the request, specifying the last mod date of the cached copy

The Server can respond with "304 not modified", if the content has not changed.

The client can just use the cached copy they already have.

Saves bandwidth, but still requires a little 2\*latency interaction

## Cache Control

May be used in either request or response

Cache-Control: no-cache      ## forbid caching

Cache-Control: max-age=600      ## limit caching to the given number of secs

Expires: *date*      ## a time, or "0" = no cache, this is the old way

## Cache Failure

Because of earlier bad cache implementations, many sites specify no-caching.

They got tired of tech-support calls where people were seeing "old" content

This is unfortunate -- it's making browsing slower than it should be, but once it's out of favor, it's hard for it to get back.

## HTTP Proxy

Another possibility is to specify a proxy server to the client.

The client makes its request to the proxy, the proxy turns around and makes the request to the real server

The proxy can cache common pages, and return them immediately

The proxy needs to understand the HTTP protocol, and the browser must be configured to use the proxy.

## Proxy Tricks

Could use the proxy for purposes other than caching

Filtering for objectionable content in a library

Enabling people in China to get to forbidden web sites. Suppose

www.maosucks.com is blocked from inside China. From inside China, make a request to proxy.stanford.edu, and it in turns makes the request to www.maosucks.com and relays back the content.

## http-equiv

In the HEAD section of the HTML

A low-budget way of communicating HTTP like things to the browser

```
<head>
```

```
  <meta http-equiv="cache-control" content="no-cache">
```

```
</head>
```

## Description and Keywords

Search engines may be able to use this meta content to better categorize the content...

```
<meta name="description" content="Stanford CS Education Library: a fun 3
  minute video that explains the basics features of pointers.">
```

```
<meta name="keywords" content="pointer, pointee, reference, pointer fun, basic
  pointer, pointer introduction, video, animation, animated">
```

## Refresh

Read by the client

```
<META HTTP-EQUIV="Refresh" CONTENT="seconds; URL=url">
```

Can re-load the page on some schedule (URL is optional)

Not an official part of HTML -- a cheesy extension

Or can "bounce" to another page -- a quick-n-dirty technique if you need to bounce the client to another page. Not as clean as a real 301 re-direct, but much easier, since it only requires HTML.

This technique can mess up the function of the back button -- bad style

## .shtml

Some server support this.

Directives embedded in HTML comments

Processed by server at request time -- slows things down a bit

e.g. insert mod time of file...

```
<!--#flastmod virtual="foo.html" -->
```