

CSE 123b Communications Software

Spring 2002

Lecture 11: HTTP

Stefan Savage

Some slides courtesy Srinu Seshan

Project #2

- On the Web page in the next 2 hours
- Due in two weeks
- Project reliable transport protocol on top of routing protocol assignment
- New fishhead option (packet loss probability)
- You will turnin assignment using your assignment ☺

May 16, 2002

CSE 123b – Lecture 11 – DNS

2

HTTP Basics

- HTTP layered over TCP
- Interaction
 - Client sends request to server, followed by response from server to client
 - Requests/responses are encoded in text
- How to mark end of message?
 - Size of message → Content-Length
 - » Must know size of transfer in advance
 - Delimiter → MIME style Content-Type
 - » Server must “byte-stuff”
 - Close connection
 - » Only server can do this

May 16, 2002

CSE 123b – Lecture 11 – DNS

3

HTTP Messages

- Four parts
 - START LINE <CRLF>
 - » Request or response
 - MESSAGE HEADER <CRLF> <CRLF>
 - » 0 or more of these; meta data
 - MESSAGE BODY <CRLF>
 - » Actual content

May 16, 2002

CSE 123b – Lecture 11 – DNS

4

HTTP Request

- Request line
 - Method
 - » GET – return URL
 - » HEAD – return headers only of GET response
 - » POST – send data to the server (forms, etc.)
 - URL
 - » E.g. `http://www.cs.ucsd.edu/index.html`
 - » `index.html` by default
 - HTTP version
- Example
GET `http://www.cs.ucsd.edu/index.html` HTTP/1.1

May 16, 2002

CSE 123b – Lecture 11 – DNS

5

HTTP Request

- Request headers
 - Authorization – authentication info
 - Acceptable document types/encodings
 - From – user email
 - If-Modified-Since
 - Host
 - Referrer – what caused this page to be requested
 - User-Agent – client software

May 16, 2002

CSE 123b – Lecture 11 – DNS

6

HTTP Request Example

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows
NT 5.0)
Host: www.cs.ucsd.edu
Connection: Keep-Alive
```

HTTP Response

- Status-line (response)
 - HTTP version
 - 3 digit response code
 - » 1XX – informational
 - » 2XX – success
 - » 3XX – redirection
 - » 4XX – client error
 - » 5XX – server error
 - Reason phrase

HTTP Response

- Headers
 - Location – for redirection
 - Server – server software
 - WWW-Authenticate – request for authentication
 - Allow – list of methods supported (get, head, etc)
 - Content-Encoding – E.g x-gzip
 - Content-Length
 - Content-Type
 - Expires
 - Last-Modified

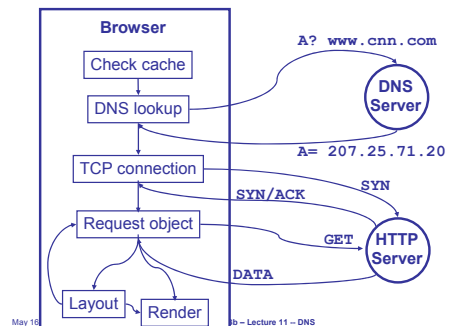
HTTP Response Example

```
HTTP/1.1 200 OK
Date: Tue, 27 Mar 2001 03:49:38 GMT
Server: Apache/1.3.14 (Unix) (Red-Hat/Linux)
      mod_ssl/2.7.1 OpenSSL/0.9.5a DAV/1.0.2
      PHP/4.0.1pl2 mod_perl/1.2.4
Last-Modified: Mon, 29 Jan 2001 17:54:18 GMT
ETag: "7a11f-10ed-3a75ae4a"
Accept-Ranges: bytes
Content-Length: 4333
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html
```

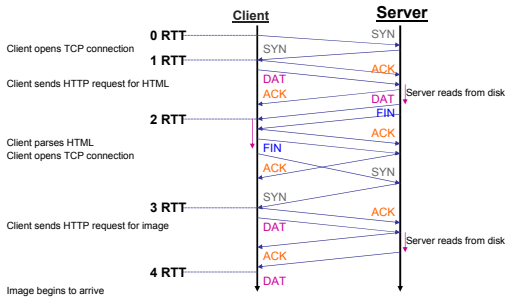
How HTTP is used

- Browser requests main page
- Browser parses main page for inline images and content (e.g. <imgsrc=http://k.com/pics/bullet>)
 - Typically 4-5 of these
 - Issues additional requests to server (or other server) for additional data items
- Browser computes location of objects on page (layout)
- Browser draws each object on the screen (rendering)
- If user clicks on a link (href) then restart process

The Way Web Surfing Works...



Single Transfer Example



Problems

- Short transfers are hard on TCP
 - Stuck in slow start
 - Loss recovery is poor when windows are small. Why?
- Lots of extra connections
 - Increases server state/processing
- Server also forced to keep TIME_WAIT connection state
 - Why must server keep these?
 - Tends to be an order of magnitude greater than # of active connections

May 16, 2002

CSE 123b - Lecture 11 - DNS

14

Netscape Solution

- Use multiple concurrent connections to improve response time
 - Different parts of Web page arrive independently
 - Can grab more of the network bandwidth than other users
- Doesn't necessarily improve response time
 - TCP loss recovery can be timeout dominated because windows are small

May 16, 2002

CSE 123b - Lecture 11 - DNS

15

HTTP 0.9/1.0

- One request/response per TCP connection
 - Simple to implement
- Disadvantages
 - Multiple connection setups → three-way handshake each time
 - » Several extra round trips added to transfer
 - Multiple slow starts

May 16, 2002

CSE 123b - Lecture 11 - DNS

16

Persistent Connection Solution

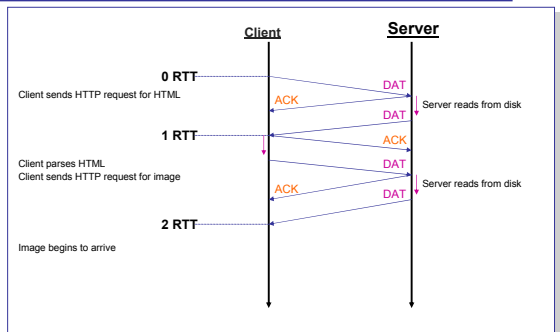
- Multiplex multiple transfers onto one TCP connection
 - Serialize transfers → client makes next request only after previous response
- How to demultiplex requests/responses
 - Content-length and delimiter
 - Block-based transmission – send in multiple length delimited blocks
 - Store-and-forward – wait for entire response and then use content-length

May 16, 2002

CSE 123b - Lecture 11 - DNS

17

Persistent Connection Example



Persistent connection Solution

- However, serialized requests do not improve interactive response
- Pipelining requests
 - Getall – request HTML document and all embeds
 - » Requires server to parse HTML files
 - » Doesn't consider client cached documents
 - Getlist – request a set of documents
 - » Implemented as a simple set of GETs

May 16, 2002

CSE 123b – Lecture 11 – DNS

19

Persistent Connection Performance

- Benefits greatest for small objects
 - Up to 2x improvement in response time
- Server resource utilization reduce due to fewer connection establishments and fewer active connections
- TCP behavior improved
 - Longer connections help adaptation to available bandwidth
 - Larger congestion window improves loss recovery
- How long to keep connection open?

May 16, 2002

CSE 123b – Lecture 11 – DNS

20

Remaining Problems

- Application specific solution
- Stall in transfer of one object prevents delivery of others
- Serialized transmission
 - Much of the useful information in first few bytes
 - Can “packetize” transfer over TCP
 - » HTTP 1.1 recommends using range requests
 - » MUX protocol provides similar generic solution
- Other solution: solve the problem at the transport layer
 - Fix TCP so it works well with multiple simultaneous connections

May 16, 2002

CSE 123b – Lecture 11 – DNS

21

TCP modifications for HTTP

- Key idea:
 - Have different connections share congestion window information
 - Only do slow start once, recover from congestion losses together
- Tricky issues
 - How aggressive should it be vs using multiple connections
 - What is the difference between
 - » One packet lost among four separate small TCP connections
 - » One packet lost in one larger TCP connection

May 16, 2002

CSE 123b – Lecture 11 – DNS

22

Typical Workload

- Multiple (typically small) objects per page
- Object sizes vary significantly
 - One measurement 1946 byte median, 13767 byte mean
 - Why so different?
 - Heavy-tailed distribution (Pareto)
 - Also true for number of embedded objects
- Popularity
 - Also heavy-tailed (Zipf)
 - Small number of very popular objects; increasingly large number of less popular objects
- Bursty interarrival pattern

May 16, 2002

CSE 123b – Lecture 11 – DNS

23

HTTP Caching

- Clients often cache documents
 - Challenge: update of documents
 - If-Modified-Since requests to check
 - » HTTP 0.9/1.0 used just date
 - » HTTP 1.1 has file signature as well
- When/how often should the original be checked for changes?
 - Check every time? each session? Day? Etc?
 - Use Expires header
 - » If no Expires, often use Last-Modified as estimate

May 16, 2002

CSE 123b – Lecture 11 – DNS

24

Example Cache Check Request

```
GET / HTTP/1.1
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
If-Modified-Since: Mon, 29 Jan 2001 17:54:18 GMT
If-None-Match: "7a11f-10ed-3a75ae4a"
User-Agent: Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)
Host: www.cs.ucsd.edu
Connection: Keep-Alive
```

May 16, 2002

CSE 123b - Lecture 11 - DNS

25

Example Cache Check Response

```
HTTP/1.1 304 Not Modified
Date: Tue, 27 Mar 2001 03:50:51 GMT
Server: Apache/1.3.14 (Unix) (Red-Hat/Linux)
  mod_ssl/2.7.1 OpenSSL/0.9.5a DAV/1.0.2
  PHP/4.0.1pl2 mod_perl/1.24
Connection: Keep-Alive
Keep-Alive: timeout=15, max=100
ETag: "7a11f-10ed-3a75ae4a"
```

May 16, 2002

CSE 123b - Lecture 11 - DNS

26

Web caching in general

- Caching in client
 - Keep unchanging portion of a page
- Caching in server
 - Keep popular content in memory so disk access not needed
- Caching in proxy
 - Between client and server
 - Responsible for sharing content between all users of proxy
 - Example: once one person at UCSD downloads CNN page, then everyone else can simply fetch it from proxy cache
 - Hit rates ~ 50%

May 16, 2002

CSE 123b - Lecture 11 - DNS

27

So what makes the Web slow?

- Client delay (parsing, layout, rendering)
- Server delay (page generation)
- DNS Lookup/loss (local, domain, root lookup, loss)
- Propagation delay (client to server, RTT/MSS)
- Queuing delay (sum per message + impact of variability)
- Connection setup & failure (server load, client timeout)
- Congestion + TCP congestion response (1/sqrt(p))
- Impact of fragmentation/packet size (RTT/MSS)
- Receiver buffering (flow control)
- Sender buffering (sender resource allocation)
- Application serialization (waiting for gif?)
- **Will the real culprit please stand up?**

May 16, 2002

CSE 123b - Lecture 11 - DNS

28

Answer is still unclear...

- Browser delay (layout/rendering) can be significant for some host/browser combinations...
- For highly loaded servers, server delay is an important component
- For high capacity networks, slow start is the limiting factor
- For networks with loss TCP congestion control can be the limiter
- For pages with many objects, the # of connections can be a factor

May 16, 2002

CSE 123b - Lecture 11 - DNS

29

Best case TCP transfer time

TCP Slow start time

$$RTT \cdot \left\lceil \log_{1.5} \left(\frac{B}{2W \cdot MSS} + 1 \right) \right\rceil$$

RTT: round-trip time
B: bytes to be transferred
W: initial window
MSS: bytes in a packet

May 16, 2002

CSE 123b - Lecture 11 - DNS

30

Impact of Congestion control

- Very simple model for steady state TCP transfer time

$$RTT \cdot \left(\frac{B \cdot \sqrt{p}}{MSS} \right)$$

RTT: round-trip time
B: bytes to be transferred
MSS: bytes in a packet
p: packet loss rate

- The models get more complicated
 - Limited receiver windows, timeouts, loss patterns, when first loss occurs, etc...

Summary

- HTTP built on top of TCP
 - Request response
 - Negotiates capabilities
- Connection per request is a limitation
 - Persistent connections
 - Pipelining, MUX
- Web caching
 - Check timestamp on cached copy against host/proxy
 - Caching at all levels
- Web is still slow