Experiential Learning Workshop on HTTP

Mar 27, 2018

Dr. Ram P Rustagi
ram@rprustagi.com

Exploration Topics - Day 1

- Overview
- Overview of Tools Needed
- Hands-on 1: using networking tools
- Understanding Network Delays
- Hands-on 2: Experiencing Network delays
- Overview of HTTP
- Understanding HTTP Req/Resp and Headers
- Hands-on 3: HTTP Basics, 200, 403, 404
- Persistent Connections, Cache-control,
- Hands-on 4: Keepalive, cache, 304, E-tags
- Summary

Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Auth, Redirect, Status 3xx,4xx
- Hands-on 5: 206, Compression, Accept-Lang
- Hands-on 6: 301/302, 400, 401
- TCP/UDP : Streaming and reliability
- Hands-on 7: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 8: Slow urls, prefetch, sharding
- Summary

Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Hands-on 5: 206, Compression, Accept-Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx
- Hands-on 6: 301/302, 400, 401, Multi-site
- TCP/UDP : Streaming and reliability
- Hands-on 7: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 8: Slow urls, prefetch, sharding
- Summary

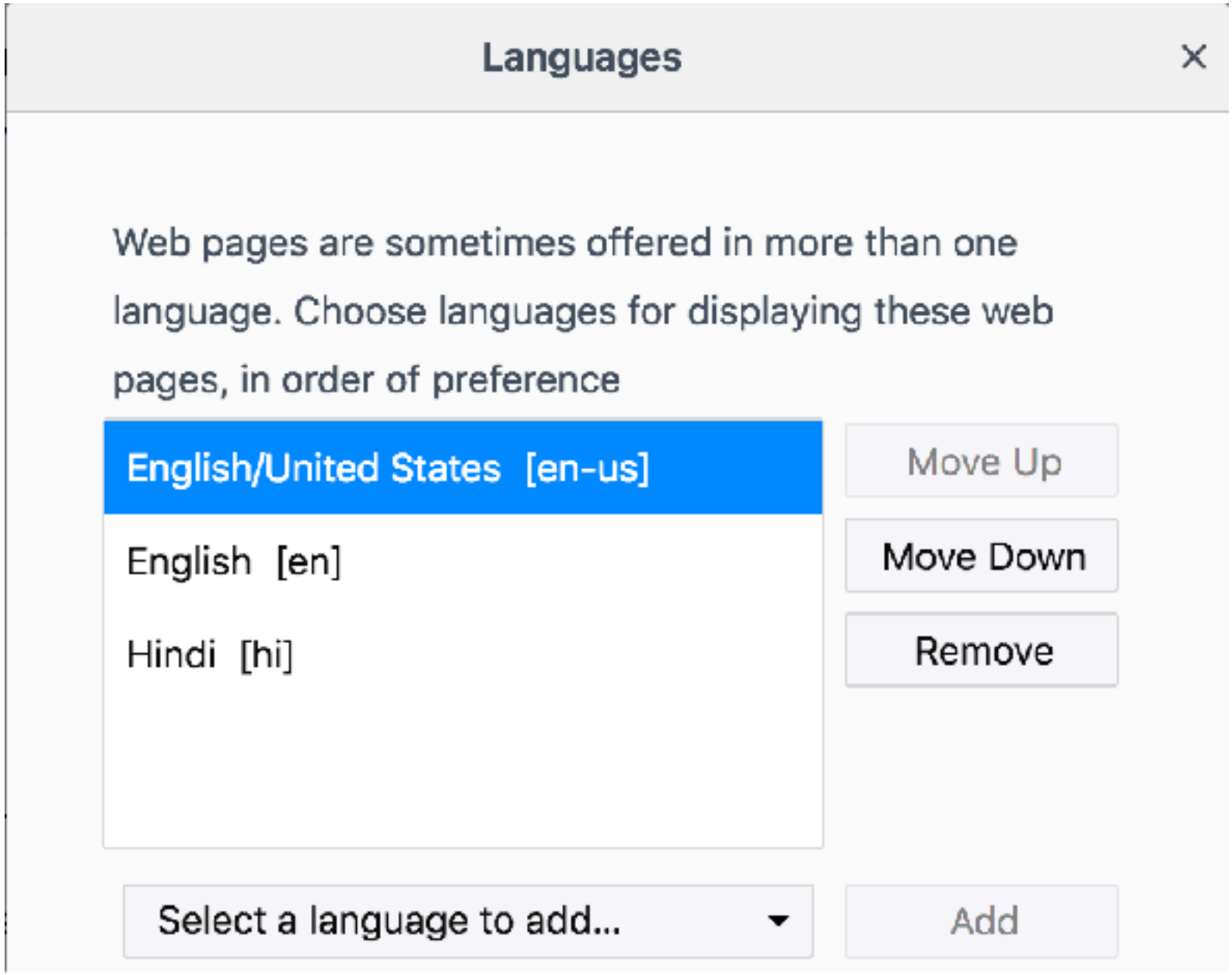
Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Hands-on 5: 206, Compression, Accept-Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx
- Hands-on 6: 301/302, 400, 401, Multi-site
- TCP/UDP : Streaming and reliability
- Hands-on 7: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 8: Slow urls, prefetch, sharding
- Summary

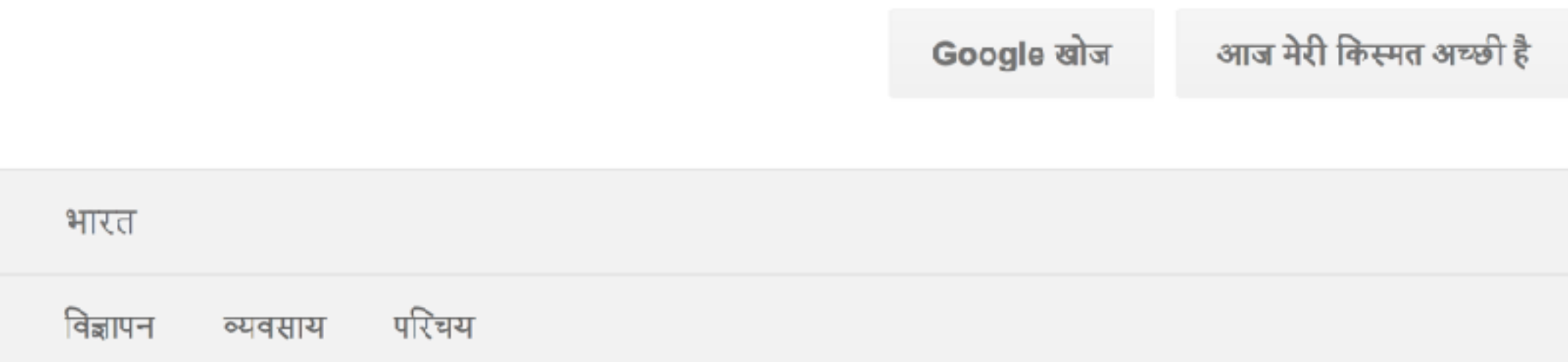
Languages

- Preferred language of user
  – Firefox: Preferences->Languages
- Header `Accept-Languages: en-US, hn-IN`
- Change your preferred language.
- Access google.com
- Web page will be in your preferred language

Firefox: Language Setting



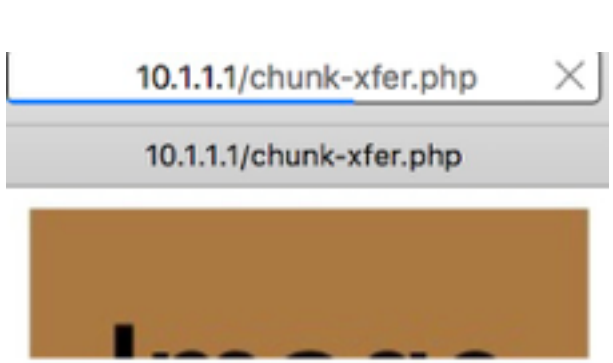
Google main in Hindi



Chunk Delivery

- Header `Transfer-encoding: chunked`
  - Continuous output and display
  - <host>/transfer-encoding-chunked.php
  - Useful for large HTML content
  - Useful for incremental image display.
  - URL: rprustagi.com/bites/chunk-xfer.php

Incremental Image Display



Incremental Image Display

```
HTTP/1.1 200 OK
Date: Sun, 25 Mar 2018 05:41:36 GMT
Server: Apache
Transfer-Encoding: chunked
Content-Type: image/jpeg

3e8
......JFIF.............Exif..MM.*.........
.................J..........R.(.
:
3e8
<?xpacket end="w"?>....8Photoshop
3.0.8BIM.....8
:
```

Sample Server Program

```
header('Content-Type: image/jpeg');
header('Transfer-Encoding: chunked');
$chunkSize = 1000;
$handle = fopen('../img/img-07.jpg', 'rb');
while (!feof($handle)) {
    $buffer = fread($handle, $chunkSize);
    #chunk size in hex, content, new line
    echo sprintf("%x\r\n", $chunkSize);
    echo $buffer; echo "\r\n";
    flush();
    usleep(500000); # mimic network latency
    }
fclose($handle)
```

Partial Content Request

```
GET /pictures.html HTTP/1.1
   Host: 10.211.55.9
   User-Agent: Mozilla/5.0 …
   Accept: text/html,application/
   xhtml+xml
   Accept-Language: en-us,en;q=0.5
   Range: bytes=100-200
```

Partial Content Response

```
HTTP/1.1 206 Partial Content
Date: Sun, 25 Mar 2018 05:56:57 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 18 Mar 2018
08:00:31 GMT
Accept-Ranges: bytes
Content-Length: 68
Vary: Accept-Encoding
Content-Range: bytes 78-145/395
Content-Type: text/html

<img src="/img/img-01.jpg" />
<img src="/img/img-02.jpg" />
```

Partial Content Access

- `$ curl -H "Range: bytes=0-199" -o chunk1.html http://myweb.com/mypage.html`
- `$ curl -H "Range: bytes=200-399" -o chunk2.html http://myweb.com/mypage.html`
- `$ curl -H "Range: bytes=400-" -o chunk3.html http://myweb.com/mypage.html`
- `$ cat chunk1.html chunk2.html chunk3.html >response.html`
- open `response.html` in browser

Compression

- HTTP Header
  - `Accept-encoding: gzip, deflate`
- When web server supports compression, response will be compressed using specified compression mechanism
- HTTP Response headers are
  - `Vary: Accept-Encoding`
  - `Content-Encoding: gzip`

Partial Content Compressed Response

```
HTTP/1.1 206 Partial Content
Date: Sun, 25 Mar 2018 05:58:36 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Sun, 18 Mar 2018
08:00:31 GMT
Accept-Ranges: bytes
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Range: bytes 78-145/153
Content-Length: 68
Content-Type: text/html

?&37]??(?VI?a]C???t%}?Fxe??ɵ??5?+k?W???
```

HTTP Headers - Authentication

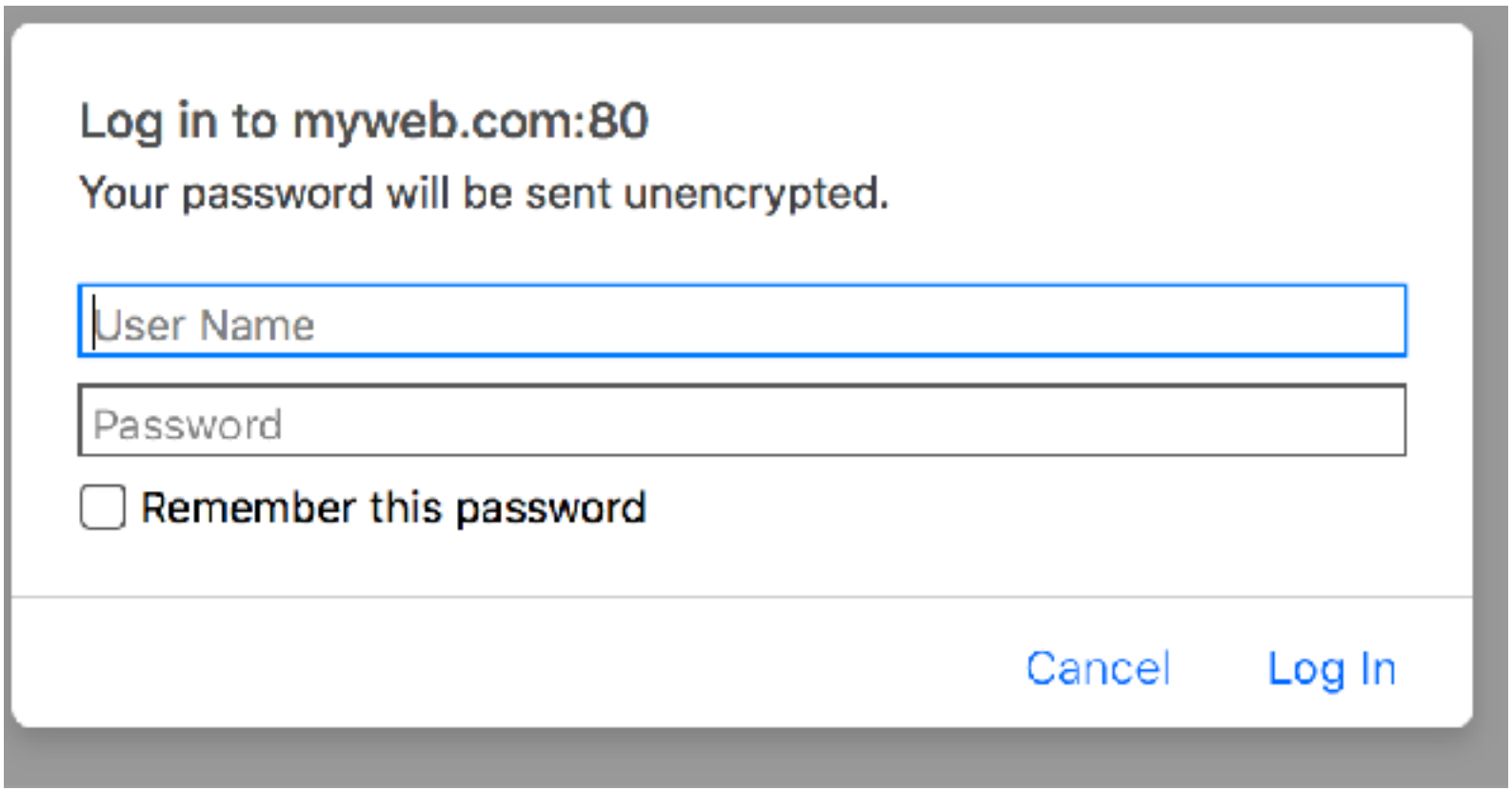
- Authorization: Basic
  - Uses Base64 encoding
- Apache configuration

```
<Directory /var/www/html/private>
 AuthType Basic
 AuthName "For Bites Workshop"
 AuthBasicProvider file
 AuthUserFile /etc/apache2/passwdfile
 Require user CSE
</Directory>
```

- Commands to create passwords
  - `htpasswd [-c] /etc/apache2/passwdfile`
- Restart apache

HTTP Authentication

- Access /private/abcd.html
  - Should see the response as below
  - Enter the username/pass



Encoding of username/password

- Uses Base64 encoding.
  - Letters:
    - 'A-Za-z0-9+/' # 64 letters (6 bits)
    - '=' a filler.
    - For input data, take 6 bits at a time and use the corresponding encoding.
  - Example: 'bites' is 0x6269746564, i.e.

```
01100010 01101001 01110100 01100101 01100100
```

  - First 6 bits: 011000 i.e. value 24 i.e. letter Y (0—>A)
  - Second 6 bites: 100110 i.e. value 40, letter 'o'
- Username and password are separated by ':' (Colon)
- Transmitted in clear text

HTTP Redirect

- Content type
  - accessing hello.html vs hello.txt
    - `Content-Type: text/html`
    - `Content-Type: text/plain`
- Location (i.e. URL Redirect)
- Apache config

  `-Redirect /CSE http://cse.ewit.local`
- Access <server>/CSE
  - 302 - `Location: http://cse.ewit.local`
- Access <server>/ISE
  - 301 - `Location: http://ise.ewit.local`
- Access `google.com`
  - user browser or (`wget -d` <URL>)

Dynamic Web

- Invoking CGI
  - Apache config

```
<Directory /var/www/html/cgi>
    Options ExecCGI
    SetHandler cgi-script
</Directory>
```

  - Enabling CGI as module
    - `sudo a2enmod cgi`

Working of cgi-bin

- Web server executes the program referred in URL
- Program is responsible for providing all the response header
  - Web server reads all the data sent by program
  - Validates that output is proper
- If program crashes (exits improperly)
  - HTTP headers could be corrupted/improper
- When web servers sees inconsistency,
  - Given `500 Internal Server Error`

Example of 500 error

- Sample CGI script : `cgi-good.sh`

```
#!/bin/bash
echo "Content-Type: text/html”;
echo “”;
echo "<h1>Hello World!</h1>”;
exit;
```

- Error noticed by web server
  - No empty line between HTTP headers and HTML content

Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx,5xx
- Hands-on 5: 206, Compression, Accept-Lang
- Hands-on 6: 301/302, 400, 401, Multi-site
- TCP/UDP : Streaming and reliability
- Hands-on 7: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 8: Slow urls, prefetch, sharding
- Summary

Hands-on 5:

- Use of compression
  - Use header Accept-Encoding: gzip to access the content. Use it with partial content access to download the same large file and verify the access
- Language preference usage
  - Set the language of your preference and access google.com and check the web page.

Hands-on 5:

- Incremental image display
  - Choose your image and see the continuous display. Install the program chunk-xfer.php
- Partial content access
  - Use the header Range: bytes:m1-m2 to access the partial content. Access a large file (e.g. 100KB) in chunks of 20KB, join the chunks and open the file in browser to verify that file is intact.

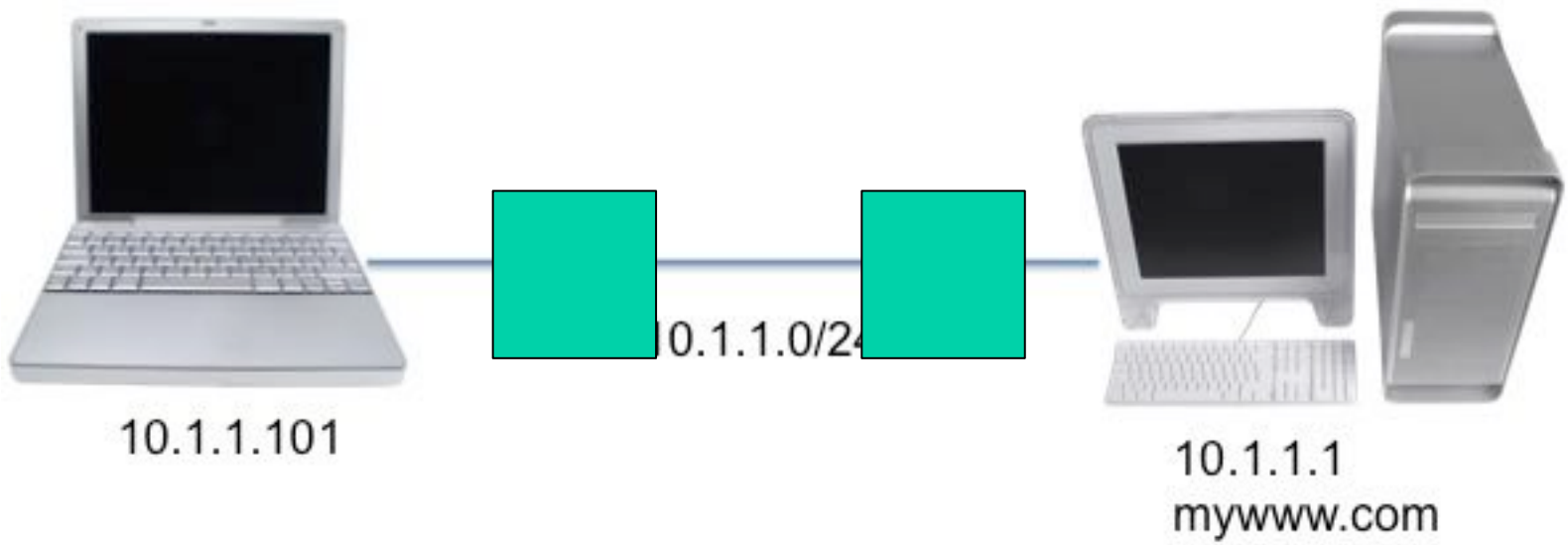
Hands-On 6

- Status code 301/302 (Header `Location:`)
  - `wget -d http://google.com`
- Status code `500`
  - <host:>`/cgi/bad-cgi.sh`
- Status code `401` (change Apache config)
  - Define username/password
  - Configure a directory with authentication
  - Restart apache
  - Access the web page from this URL.

Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Hands-on 5: 206, Compression, Accept-Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx
- Hands-on 6: 301/302, 400, 401, Multi-site
- TCP/UDP : Streaming and reliability
- Hands-on 7: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 8: Slow urls, prefetch, sharding Summary

TCP/UDP Setup Requirement



Transport Layer Protocol Characteristics

- Connection less
  - If Packets are not numbered, can't provide the order
  - may arrive out of order
  - No acknowledgement, Packets may be lost
  - No prior handshake
- Connection oriented
  - Setup, data xfer and teardown phase
  - Provides reliability, ordered delivery
  - Handles error control in a better way

Transport Layer Reliability Support

- Reliability
  - Needs error and flow control
    - Compels slower service
- Unreliable protocol
  - No extra overheads
- Reliability at data link layer
  - Provides error and flow control
  - Why do we need it at Transport layer?

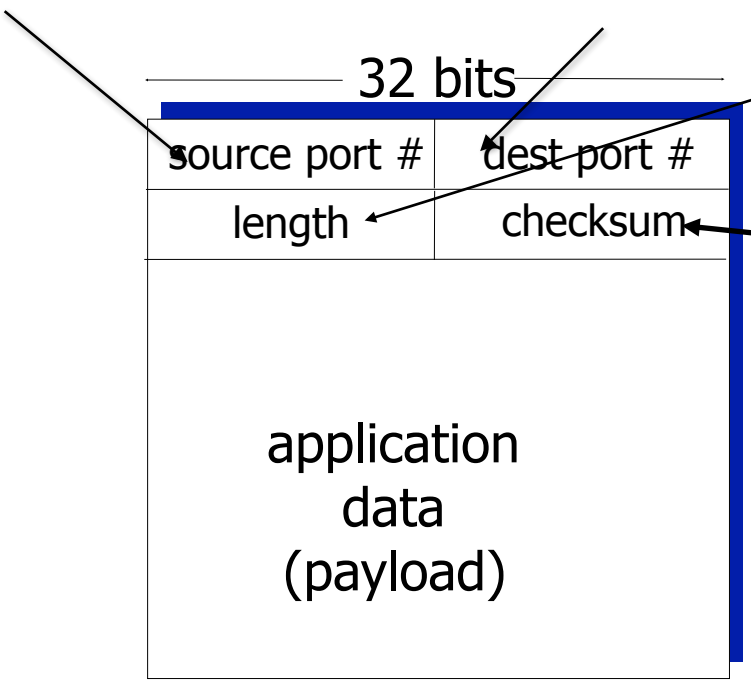
UDP Overview

- *no frills, bare bones* Internet transport protocol
- *best effort* service, UDP segments may be:
  - lost, delivered out-of-order to application
- connectionless
  - no handshaking between UDP sender, receiver
  - each UDP segment handled independently of others
- Used by: UDP used by:
  - streaming multimedia apps (loss tolerant, rate sensitive)

UDP Design Requirement

- How to design a simple transport layer ?
- Just provide transport on top of IP
  - Multiplexing and demultiplexing
  - Little bit of error checking
  - No handshake
- Rest all has to be managed by application
  - Application practically talks to IP
- DNS uses UDP
  - What happens when query/response is lost?

UDP: segment header



UDP segment format

length, in bytes of UDP segment, including header

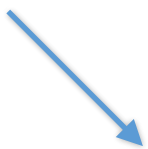
includes pseudo header

- no connection establishment (which can add delay)
- simple: no connection state at sender, receiver
- small header size
- no congestion control: UDP can blast away as fast as desired

Internet checksum: example RFC 1071

- Consider 3 words

```
    0110 0110 0110 0000   - 0x6660
    0101 0101 0101 0101   - 0x5555
    1000 1111 0000 1100   - 0x8F0C
   ----------------------------------------
   10100 1010 1100 0001   -0x14AC1
```



- Wrapping around the overflow bit makes it

```
    0100 1010 1100 0010   - 0x4AC2
```

- 1's complement will be

```
    1011 0101 0011 1101   - 0xB53D
```

When UDP is better than TCP

- Real time apps do not want congestion control
  - Some packet loss is okay
- Connection handshake not required
  - No overhead and quick response e.g. DNS
  - analogy: SMS/Whatsapp vs phone call, Alerts?
- No connection state maintenance
  - OS has less resources overhead for UDP
  - Can support more UDP clients than TCP
- Better utilisation efficiency
  - UPD overhead is 8 bytes vs 20 bytes of TCP

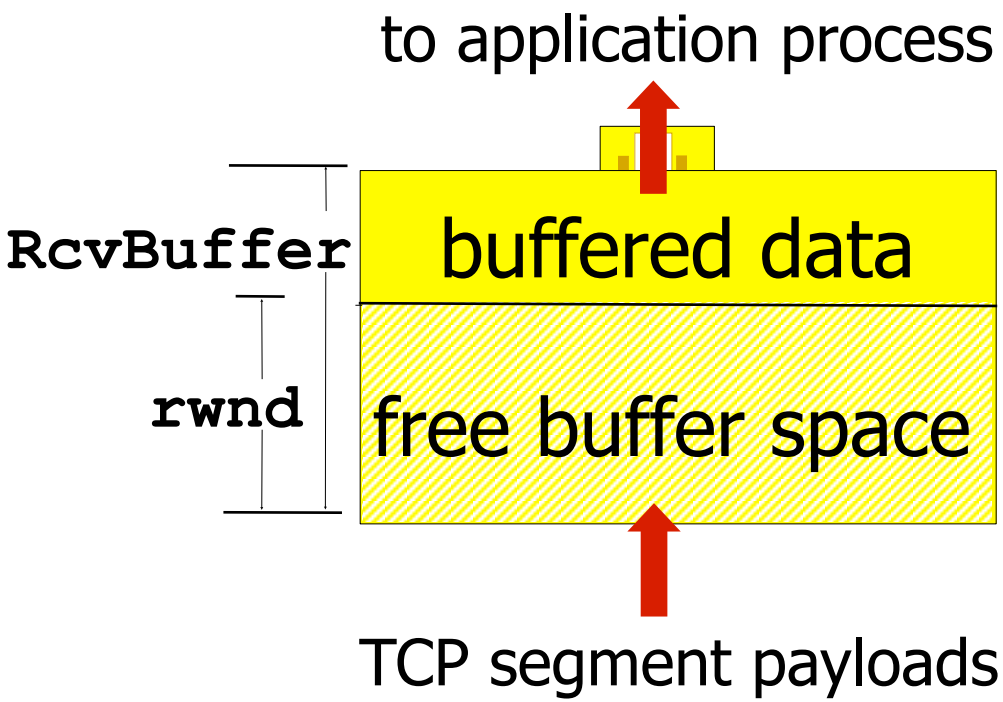
TCP Characteristics

- *point-to-point*: One sender, one receiver
- *reliable, in-order byte steam: No "message boundaries"*
- pipelined: *TCP congestion and flow control set window size*
- full duplex *data: Bi-directional data flow in same connection*
  - *MSS: maximum segment size, determined from link/ frame size*
- *connection-oriented*: Handshaking (exchange of control msgs) inits sender, receiver state before data exchange
- *flow controlled*: Sender will not overwhelm receiver

TCP Timeouts

- On timer expiration
  - Retransmits the segment that is not yet acked
  - Sets the `TimeoutInterval` double of previous value
  - Example: First time out `0.75s`
    - 2nd/3rd timeout will be `1.5s, 3.0s`
  - On receipt of Ack, it is computed from
- Provides a limited form of congestion control

TCP Flow Control



receiver-side buffering

src: Computer Networking : Kurose, Ross

Exploration Topics - Day 2

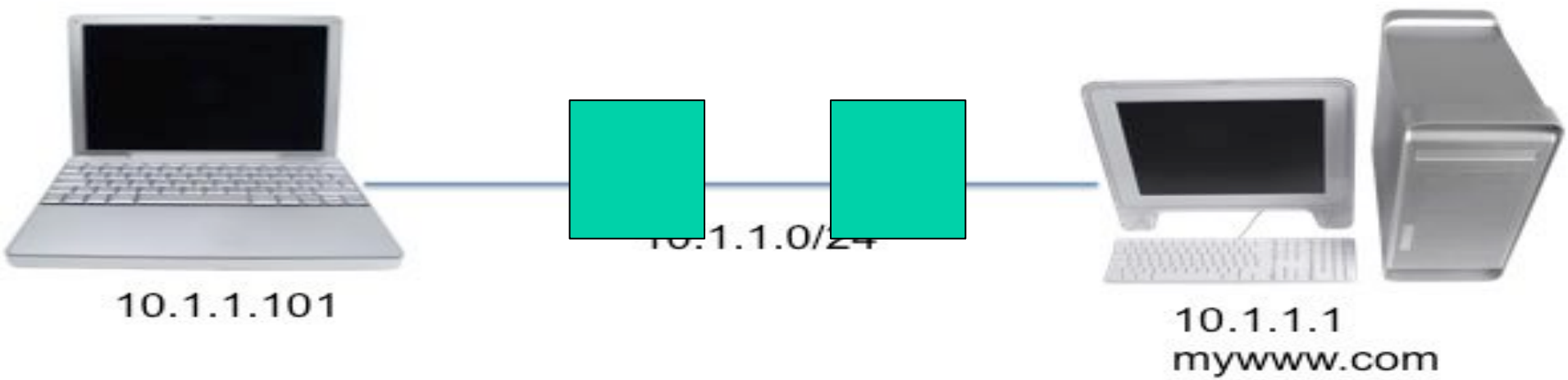
- Overview
- Partial Content Delivery, Compression, Lang
- Hands-on 5: 206, Compression, Accept-Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx
- Hands-on 6: 301/302, 400, 401, Multi-site
- TCP/UDP : Streaming and reliability
- Hands-on 7: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 7: Slow urls, prefetch, sharding
- Summary

Hands-On 7:a

- Between two machines communicate communicate UDP packets using nc (-u option).
- Send the data content as below and compute the checksum using the IP Address and port number used and verify it with checksum value in wireshark capture.
  - "ABCDE" and
  - "ABCDEUQUQUQ"
  - Are the two checksum same? Why or why not?
  - What other characters you can attach to "ABCDE" to ensure that checksum does not change

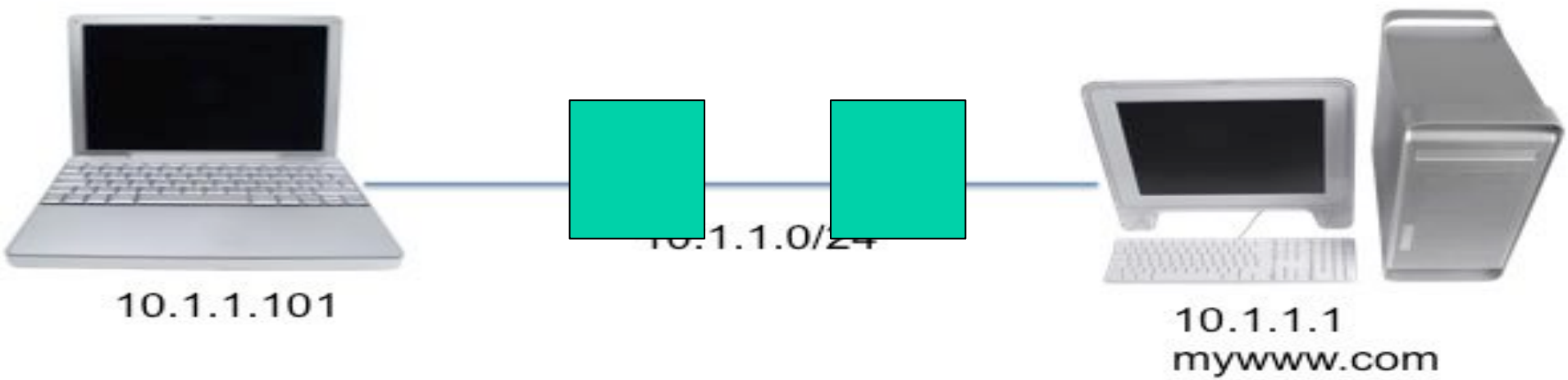
Hands-On 7b: UDP Packet Loss

- Data Transfer
  - Clients sends data (100 bytes) every 2s (10 times)
  - AA.. (1st pkt), BB..(2nd pkt), …, JJ..(10th pkt)
  - Break the link between switch at 7th sec and restore after 12 sec
- Server reads full 100 bytes.
- What would server receive:



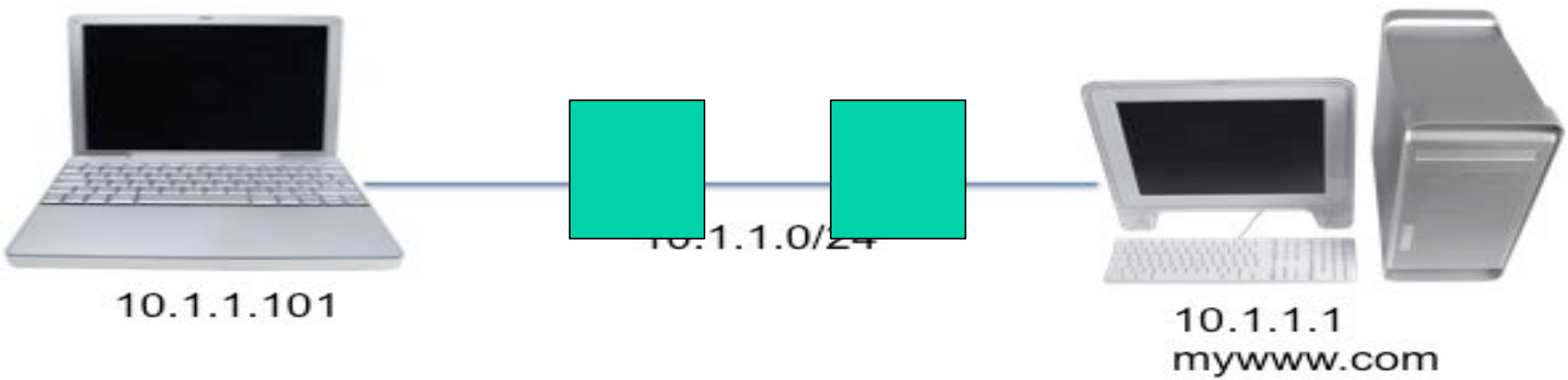
Hands-On 7c: UDP Messaging

- Data Transfer
  - Clients sends data (100 bytes) every 2s (10 times)
  - AA.. (1st pkt), BB..(2nd pkt), …, JJ..(10th pkt)
  - Break the link between switch at 7th sec and restore after 12 sec
- Server reads 50 bytes at a time
  - What would server receive?



TCP Streaming and Packet Loss

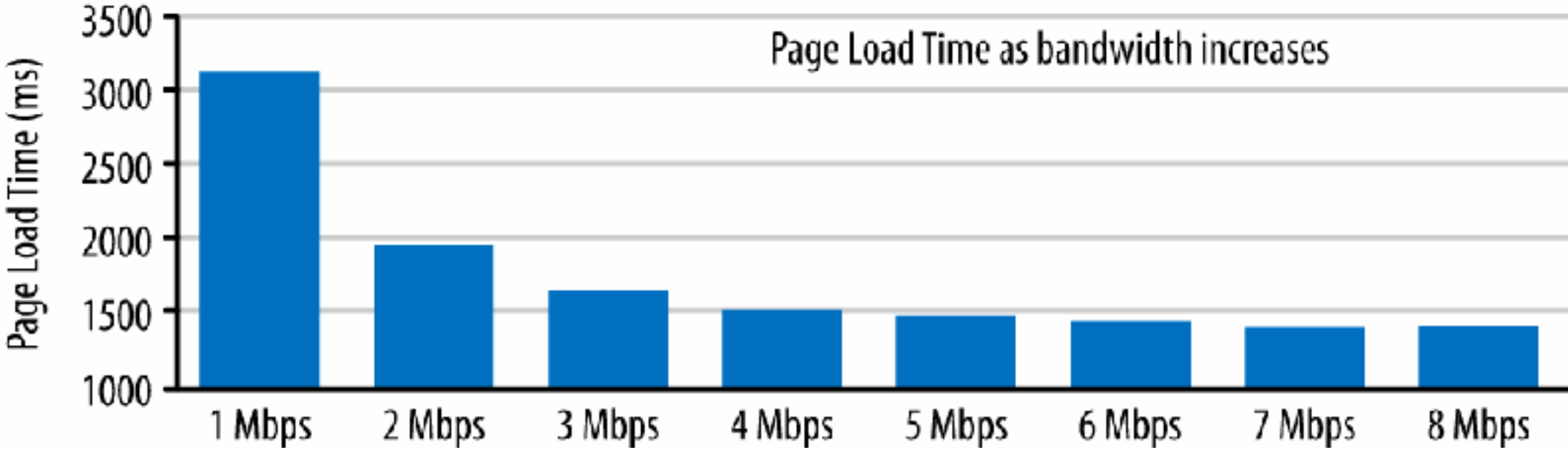
- Data Transfer
  - Clients sends data (100 bytes) every 2s (10 times)
    - AA.. (1st pkt), BB..(2nd pkt), …, JJ..(10th pkt)
  - Break the link between switch at 7th sec and restore after 12 sec
- Server reads 50 bytes at a time
- What would server receive:

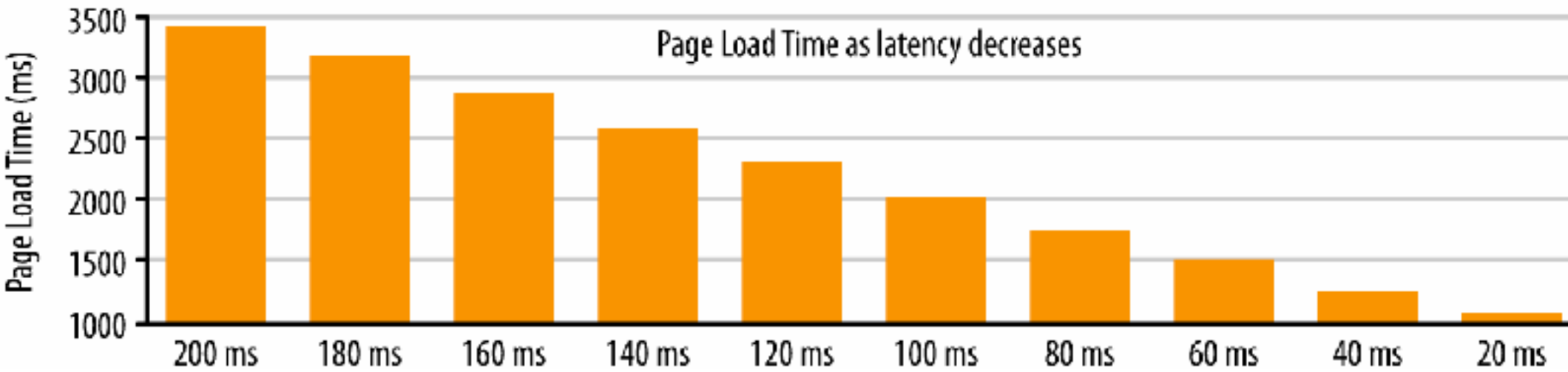


Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Hands-on 5: 206, Compression, Accept-Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx
- Hands-on 6: 301/302, 400, 401, Multi-site
- TCP/UDP : Streaming and reliability
- Hands-on 7: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 8: Slow urls, prefetch, sharding
- Summary

Page Load Time vs Latency





src: http://hpbn.co

Anatomy of Web AppIn

- src: httparchive.org/trends.php (Dec 2016)
- An avg web page
  - Domains: 20, Max Reqs on `1` domain: 50+
  - HTTPS requests: 37%, Compressed resp: 77%
  - Page with Redirects: 76%, with errors: 28%
  - Sites using CDN: 21%, cacheable resources: 49%
  - DOM elements: 900, Images: 55 requests, 1.6MB
  - TCP Conns: 33
  - Javascript: 23 requests, 400+KB

Webpage analysis

- http://webpagetest.org

Anatomy: http://ewit.edu

Load Time	First Byte	Start Render	Visually Complete
20.476s	0.733s	3.189s	20.290s

Document Complete			Fully Loaded			
Time	Requests	Bytes In	Time	Requests	Bytes In	Certificates
20.476s	170	9,100 KB	20.931s	171	9,102 KB	54 KB

src: www.webpagetest.org
Apr 10, 2017

Content Pre-fetch

- DNS Prefetch
- Content pre-fetch
- Rendering pre-fetched contents

Domain Sharding

- Domain and sub-domains
  – ?

Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Hands-on 5: 206, Compression, Accept-Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx
- Hands-on 6: 301/302, 400, 401, Multi-site
- TCP/UDP : Streaming and reliability
- Hands-on 8: Setup, Data Xfer, tear down
- Analyzing web perf, prefetch, domain sharding
- Hands-on 8: Slow urls, prefetch, sharding
- Summary

Exploration Topics - Day 2

- Overview
- Partial Content Delivery, Compression, Lang
- Hands-on 5: 206, Compression, Accept-Lang
- Auth, Redirect, Multi-sites, Status 3xx,4xx
- Hands-on 6: 301/302, 400, 401, Multi-site
- Analyzing web perf, prefetch, domain sharding
- Hands-on 7: Slow urls, prefetch, sharding
- TCP/UDP : Streaming and reliability
- Hands-on 8: Setup, Data Xfer, tear down
- Summary

Thank You