SHARKFEST'14

WIRESHARK DEVELOPER AND USER CONFERENCE
JUNE 16-20 2014 › DOMINICAN UNIVERSITY

# B6: GET /started/with/ HTTP Analysis

Robert Bullen
Application Performance Engineer
Blue Cross Blue Shield of Minnesota
robert_d_bullen@bluecrossmn.com

# The BCBSMN Experience

- Who is Blue Cross Blue Shield of Minnesota?
  - We are the first “Blue” health plan in the nation & the largest in Minnesota.
  - We have 2.6 million members across all 50 states and 3,500 employees.
  - Our administrative costs are less than 10 cents on the dollar, among lowest in the country.
- What do I do there?
  - I perform deep dive packet analysis for a few reasons:
    - To comprehend application functionality for modeling in our APM solution.
    - Troubleshooting.
    - Troubleshooting.
    - Troubleshooting.
  - I co-architect, implement, and administer our Shared Visibility Fabric (SVF).
  - I implement and administer our packet capture appliances.
  - I code in “down” time.

# HTTP Is…

- Simple
  - It is stateless.
  - It is a ping/pong request/response protocol (ignore pipelining).
  - It uses human-readable requests, responses, headers, and sometimes payloads.

# HTTP Is…

- Distributed/Multitiered
  - Services can be load balanced.
  - Connections can be forward and/or reverse proxied.
  - Static content can be separated and cached in a different tier from dynamic content.
    - Content can be localized through a CDN.
  - Resources can be redirected (e.g. URL shrinkers rely on this).
  - Applications might be composites that pull from multiple sites.

# HTTP Is…

- Flavored
  - HTTP 1.0
  - HTTP 1.1 (this is the important one)
  - WebSockets (sorta)
  - SPDY/HTTP 2.0

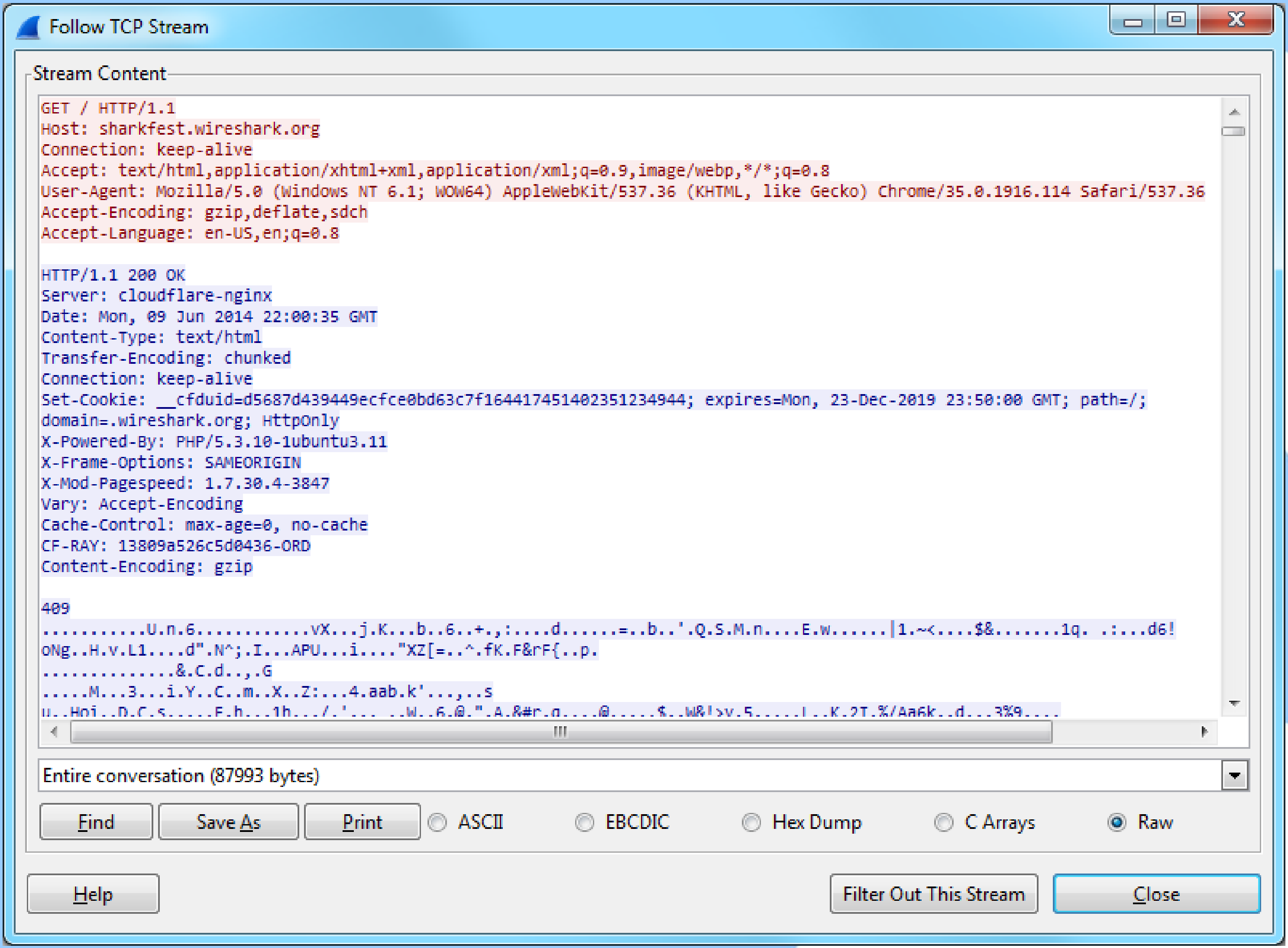
# HTTP Is…

- Ubiquitous
  - Web and application servers serving HTML.
  - Middle tier application servers publishing SOAP services.
  - Back-end SOA buses accepting SOAP/XML calls as a façade to legacy services.
  - Internet RESTful APIs to database-like resources.
  - Clients and servers are readily available as standalone programs or as libraries in most programming/scripting languages.

# HTTP Is…

- Complex
  - Applications can utilize cookies or HTML hidden fields for statefulness
  - Applications can add caching for performance
  - Applications can add concurrency for throughput
  - Applications can choose to encode content:
    - Compressed (Content-Encoding)
    - Chunked (Transfer-Encoding)
  - More and more often encryption using SSL/TLS is in place at every tier (a.k.a. HTTPS)
    - Analysis gets trickier but is still possible.
    - Remember all those distributed/multitier hops? You'll need keys for each of those tiers you with to analyze.

# HTTP Is…



```
GET / HTTP/1.1
Host: sharkfest.wireshark.org
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.114 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8

HTTP/1.1 200 OK
Server: cloudflare-nginx
Date: Mon, 09 Jun 2014 22:00:35 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: __cfduid=d5687d439449ecfce0bd63c7f16441745140235l234944; expires=Mon, 23-Dec-2019 23:50:00 GMT; path=/;
domain=.wireshark.org; HttpOnly
X-Powered-By: PHP/5.3.10-1ubuntu3.11
X-Frame-Options: SAMEORIGIN
X-Mod-Pagespeed: 1.7.30.4-3847
Vary: Accept-Encoding
Cache-Control: max-age=0, no-cache
CF-RAY: 13809a526c5d0436-ORD
Content-Encoding: gzip

409
..........U.n.6...........vX...j.K...b..6..+.,:....d......=..b..'.Q.S.M.n....E.w......1.~<....$&.......1q. .:...d6!
oNg..H.v.L1....d".N^;.I...APU...i...."XZ[=..^.fK.F&rF{..p.
.............&.C.d..,.G
.....M...3...i.Y..C..m..X..Z:...4.aab.k'...,..s
u..Hoi..D.C.s.....F.h...1h.../.'... ..W..6.@.".A.&#r.q....@.....$..W&!>v.5.....|..K.2T.%/Aa6k..d...3%9....
```

Entire conversation (87993 bytes)

# URIs

```
 http://username:password@example.com:8042/over/there/index.dtb?type=animal&name=narwhal#nose
 \__/   \_______________/ \_________/ \__/            \___/ \_/ \______________________/ \__/
  |            |               |       |                |    |            |                |
  |        userinfo         hostname  port              |    |          query          fragment
  |     \________________________________/\_____________|____|/ \__/        \__/
  |                     |                          |    |    |    |          |
  |                     |                          |    |    |    |          |
scheme              authority                    path   |    |    interpretable as keys
        \_______________________________________________|____|/       \____/     \_____/
                             |                          |    |          |           |
                     hierarchical part                  |    |    interpretable as values
                                                        |    |
                                   interpretable as filename |
                                                             |
                                                             |
                                               interpretable as extension
```

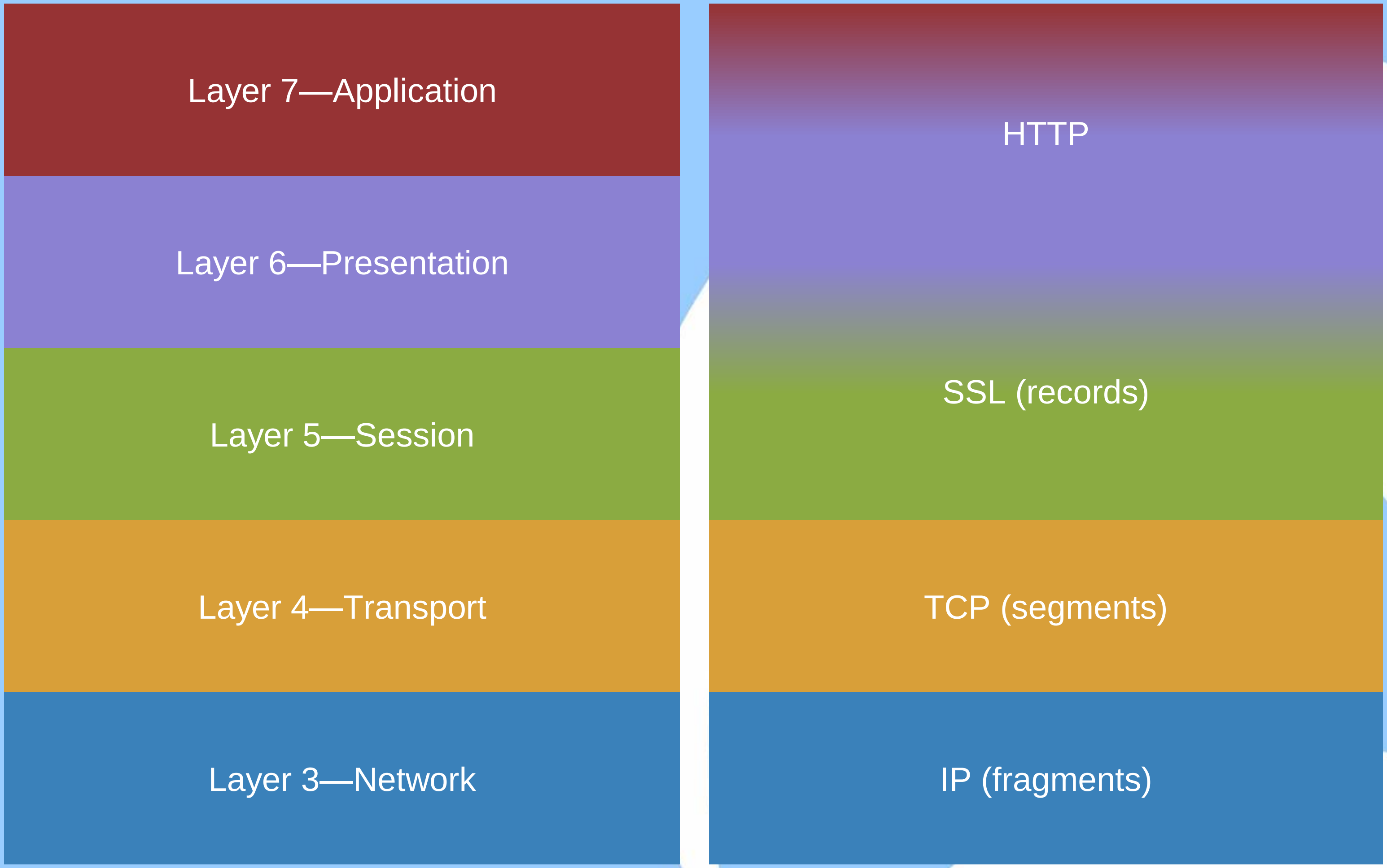
# HTTP Request Methods

- Three most common:
  - GET
    - Requests a representation of the specified resource. Requests using GET should only retrieve data and should have no other effect.
    - I equate this to a deterministic, non-modifying function (idempotent).
  - POST
    - Requests that the server accept the entity enclosed in the request as a new subordinate of the web resource identified by the URI. The data POSTed might be, as examples, an annotation for existing resources; a message for a bulletin board, newsgroup, mailing list, or comment thread; a block of data that is the result of submitting a web form to a data-handling process; or an item to add to a database.
    - I like to think of this as a nondeterministic, modifying procedure invocation.
  - CONNECT
    - Instructs an intermediate proxy to create a tunnel to the remote host.
- Others:
  - HEAD, PUT, DELETE, TRACE, OPTIONS, PATCH

# HTTP Response Status Codes

- 1xx—Informational
  - 100 Continue—The request header is valid and the client may proceed with sending the request payload.
- 2xx—Successful
  - 200 OK—Need I say more?
  - 202 Accepted—The request has been queued; check back later.
- 3xx—Redirection
  - 302 Found—The requested resource has been temporarily moved and the browser should issue a request to the URL supplied in the Location response header.
  - 304 Not Modified—The requested resource has not been modified and the browser should read from its local cache instead.
- 4xx—Client Error
  - 401 Unauthorized—Anonymous clients are not authorized to view the requested content and must provide authentication information in the WWW-Authenticate request header.
  - 404 Not Found—The requested resource does not exist on the server.
- 5xx - Server Error
  - 500 Internal Server Error—Oftentimes this is the result of an uncaught exception (i.e. an unexpected and unhandled condition or a system error such as out of memory).

# HTTP Is Layer 7



Layer 7—Application

Layer 6—Presentation

Layer 5—Session

Layer 4—Transport

Layer 3—Network

HTTP

SSL (records)

TCP (segments)

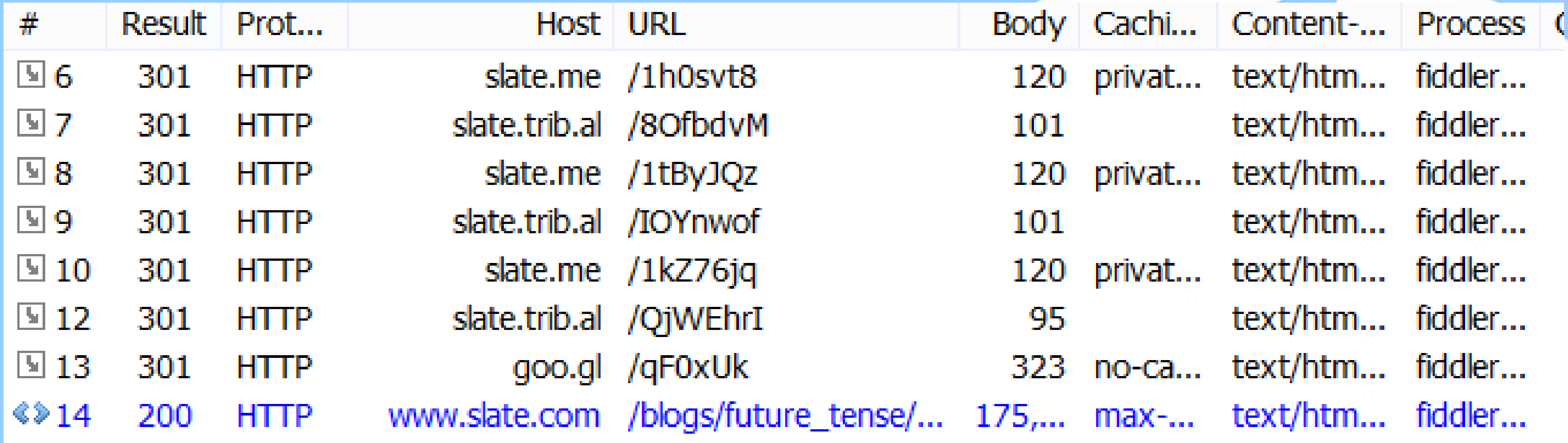
IP (fragments)

# SSL Decryption

- You must be in possession of the private key.
  - Wireshark supports PEM or PKCS#12 format. I wrote a paper covering terminology, key file formats, and extracting private keys from those file formats, which you can download at http://goo.gl/w2r7kt.
  - The negotiated cryptography algorithm must not be Diffie-Hellman.
- You must configure Wireshark with server:port to private keys mappings.
- The client key exchange must be present in the capture.
  - The client key exchange occurs during the SSL handshake.
  - Rarely you may see a client and server renegotiate in the middle of an established connection.
  - SSL has a performance optimization called session caching where a client and server can reuse previously agreed upon session keys from different conversations.

# URL Redirection

http://www.hanselman.com/blog/ThisURLShortenerSituationIsOfficiallyOutOfControl.aspx

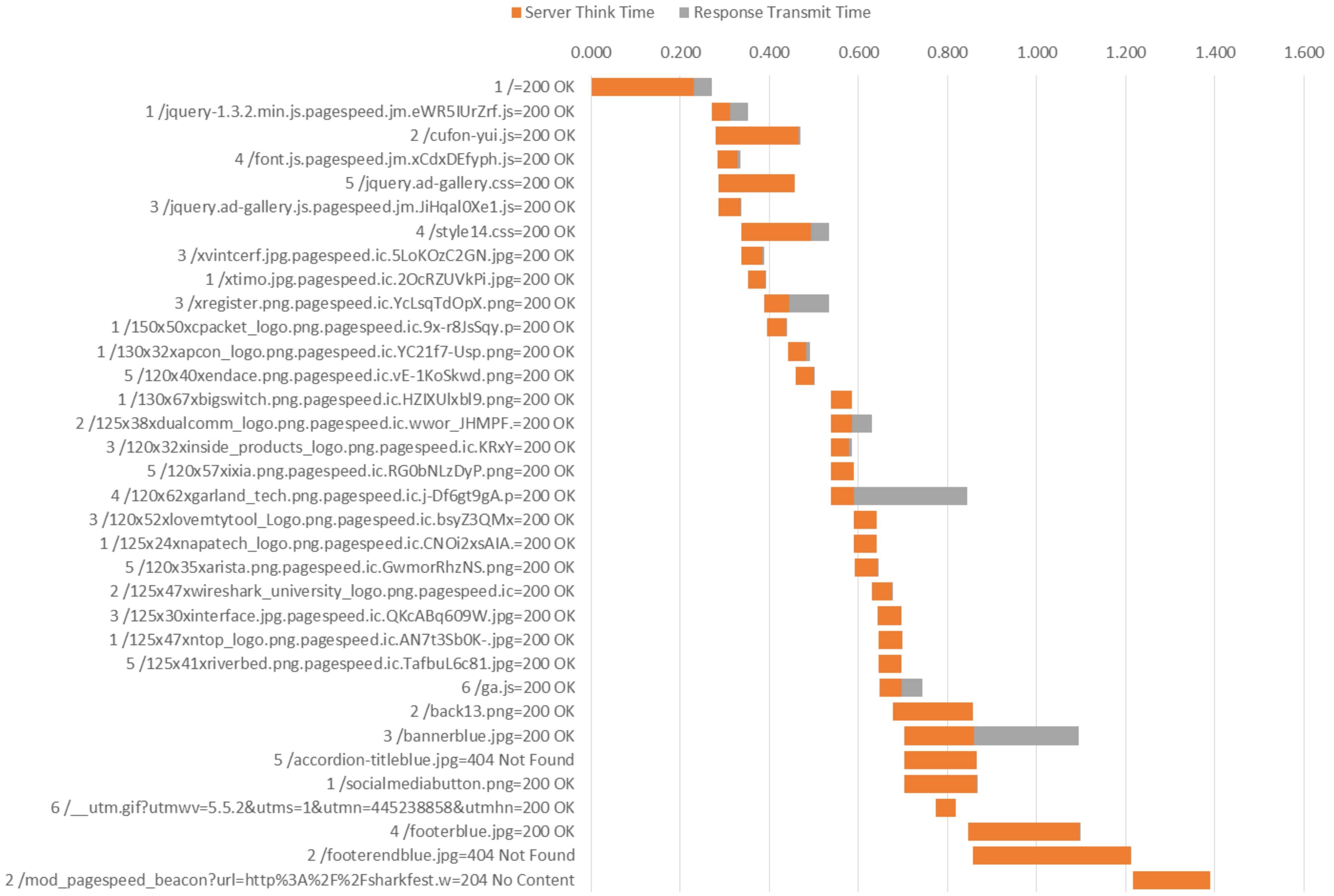


| # | Result | Prot... | Host | URL | Body | Cachi... | Content-... | Process | ( |
|---|---|---|---|---|---|---|---|---|---|
| 6 | 301 | HTTP | slate.me | /1h0svt8 | 120 | privat... | text/htm... | fiddler... | |
| 7 | 301 | HTTP | slate.trib.al | /8OfbdvM | 101 | | text/htm... | fiddler... | |
| 8 | 301 | HTTP | slate.me | /1tByJQz | 120 | privat... | text/htm... | fiddler... | |
| 9 | 301 | HTTP | slate.trib.al | /IOYnwof | 101 | | text/htm... | fiddler... | |
| 10 | 301 | HTTP | slate.me | /1kZ76jq | 120 | privat... | text/htm... | fiddler... | |
| 12 | 301 | HTTP | slate.trib.al | /QjWEhrI | 95 | | text/htm... | fiddler... | |
| 13 | 301 | HTTP | goo.gl | /qF0xUk | 323 | no-ca... | text/htm... | fiddler... | |
| 14 | 200 | HTTP | www.slate.com | /blogs/future_tense/... | 175,... | max-... | text/htm... | fiddler... | |

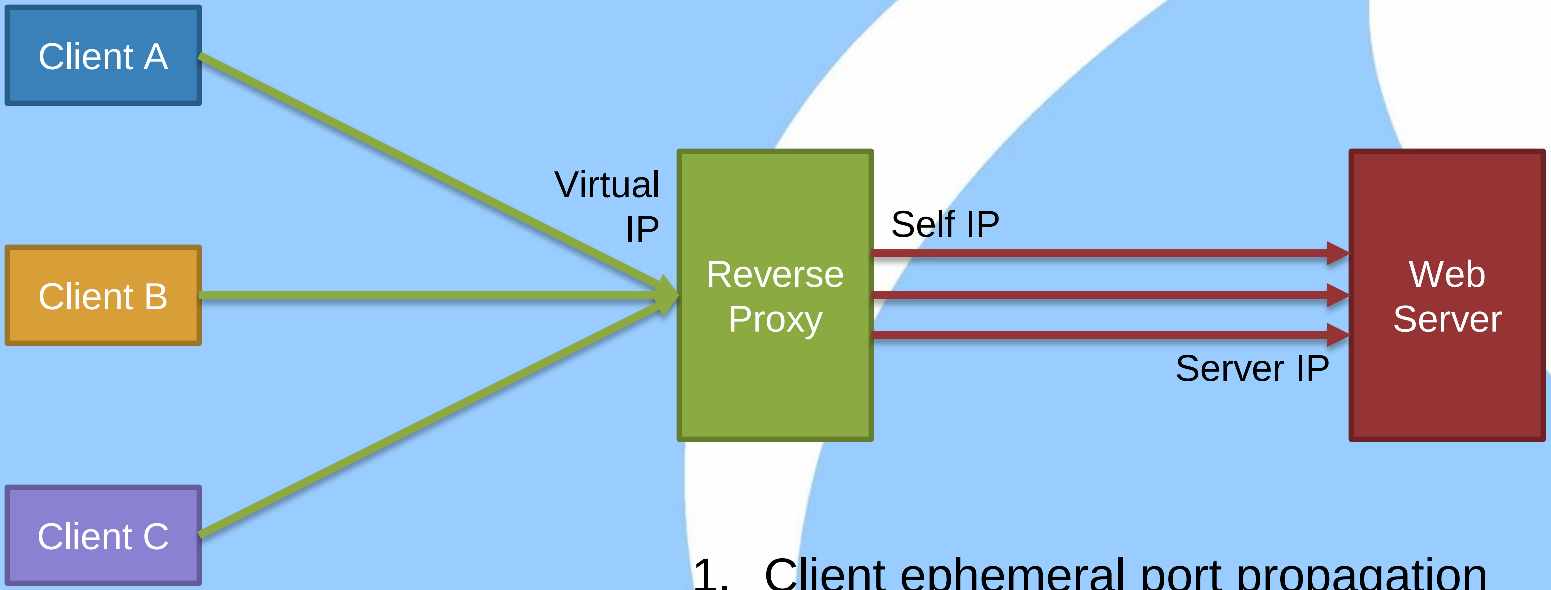
# The Waterfall Diagram

- The Waterfall diagram is the best way to start analyzing single client web page performance.
- All the major browsers now come with debugging tools baked right in (“F12” tools) that present a waterfall diagram of (among many other things).
- Third party tools are also available:
  - HTTP Watch—”HTTP Sniffer” (http://httpwatch.com)
  - Fiddler—”Web application debugging proxy” (http://www.telerik.com/fiddler)
- The information gathered by “F12” tools can be saved to an HTTP archive (HAR) file.
- A Python script called pcap2har (https://github.com/andrewf/pcap2har) can be used to convert PCAPs to HAR files.

# sharkfest.wireshark.org Home Page Load Waterfall

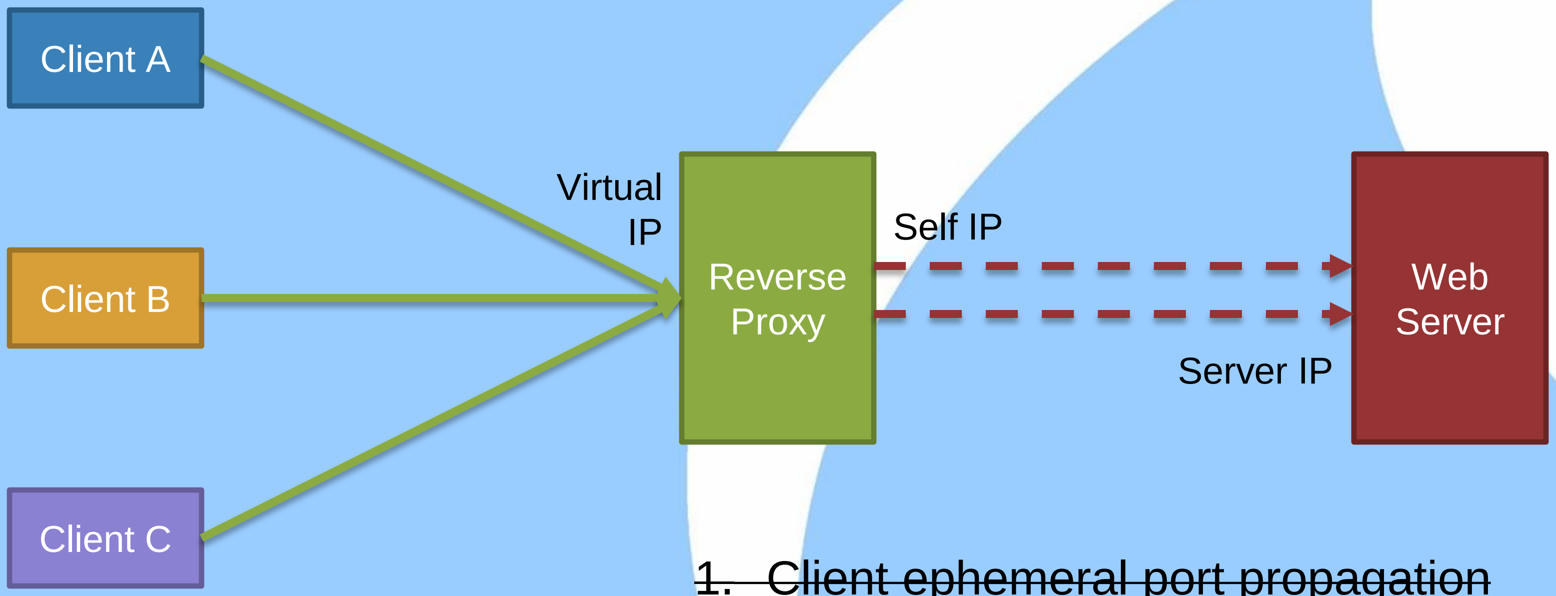


# Reverse Proxies & Client Identification



1. Client ephemeral port propagation
2. X-Forwarded-For header
3. Proxy-added cookie header (encoded)
4. Payload matching

# Reverse Proxies & Client Identification



1. ~~Client ephemeral port propagation~~
2. X-Forwarded-For header
3. Proxy-added cookie header (encoded)
4. Payload matching

# XFF, BIGIP

- X-Forwarded-For: 192.168.1.1
- BIGipServerLive_pool=375537930.544.0000
  - Decoded: IP Address: 10.65.98.22 Port: 34

# Top Performance Bottlenecks

- HTML Content
  - Improper caching of static objects
  - Requiring authentication for *every* object on a page
- Client/Server Configuration
  - Low concurrency
  - Poor TCP connection reuse
  - Poor SSL session caching
- Busy server
  - High think time
  - High response transmission time (mid-stream delays)
- Intermediate Devices
  - HTTP proxies or WAFs introducing latency
  - Load balancer challenges
    - Unsynchronized object tags on pool servers
    - Client port collisions

# Resources

- HTTP Introduction—http://www.httpwatch.com/httpgallery/
- SSL Analysis—http://sharkfest.wireshark.org/sharkfest.09/AU2_Blok_SSL_Troubleshooting_with_Wireshark_and_Tshark.pps