HTTP/3 where are we now? State of the art in our servers.

Jean-Frederic Clere

COMMUNITY OVER CODE

Bratislava, Slovakia. June 3-5, 2004

What I will cover

- HTTP/2
  - HTTP/2 and ALPN
- HTTP/3
- Servers
  - Apache HTTPD
  - Tomcat
  - Traffic server
- Demos
- Questions?

Who I am

Jean-Frederic Clere
Red Hat
Years writing JAVA code and server software
Tomcat committer since 2001
Doing OpenSource since 1999
Cyclist/Runner etc
Lived 15 years in Spain (Barcelona)
Now in Neuchâtel (CH)

Why HTTP/2

- HTTP/1.1: June 1999 (RFC 2616)
  - 1999:
    - 1 page ~ 1kB HTML
  - 2019:
    - 1 page ~ 3MB HTML + IMAGES + JS + CSS etc
- Protocol:
  - Not adapted / inefficient / etc

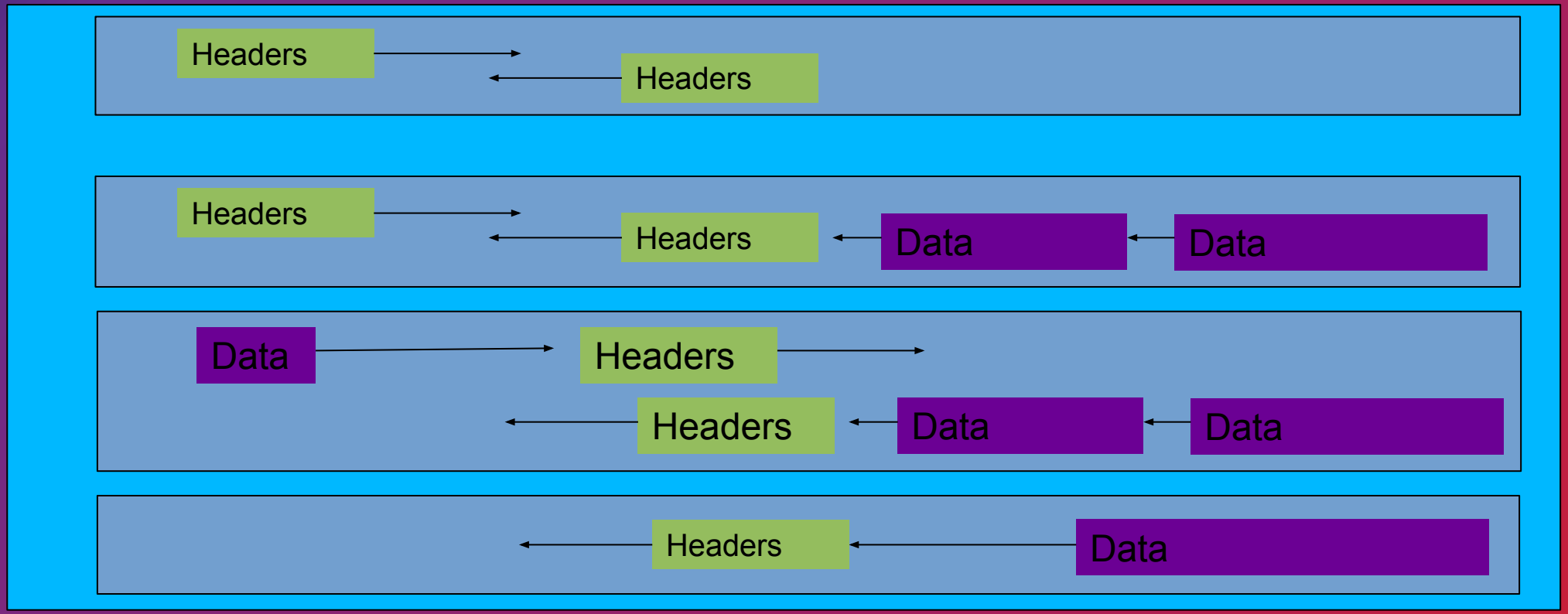
HTTP/2 general

- HTTP/2:
  - Binary
  - Frame
  - Multiplex
  - Based on SPDY
  - TLS everywhere:
    - Browser use https and strong ciphers
  - No forward proxy
  - h2c: Clear text only with reverse proxy (proxy to back-end server)

HTTP/2 general

- Two specifications:
  - Hypertext Transfer Protocol version 2 - RFC 7540
  - HPACK - Header Compression for HTTP/2 - RFC 7541
- By the Internet Engineering Task Force
- ALPN Application-Layer Protocol Negotiation - RFC 7301

HTTP/2 Multiplexed



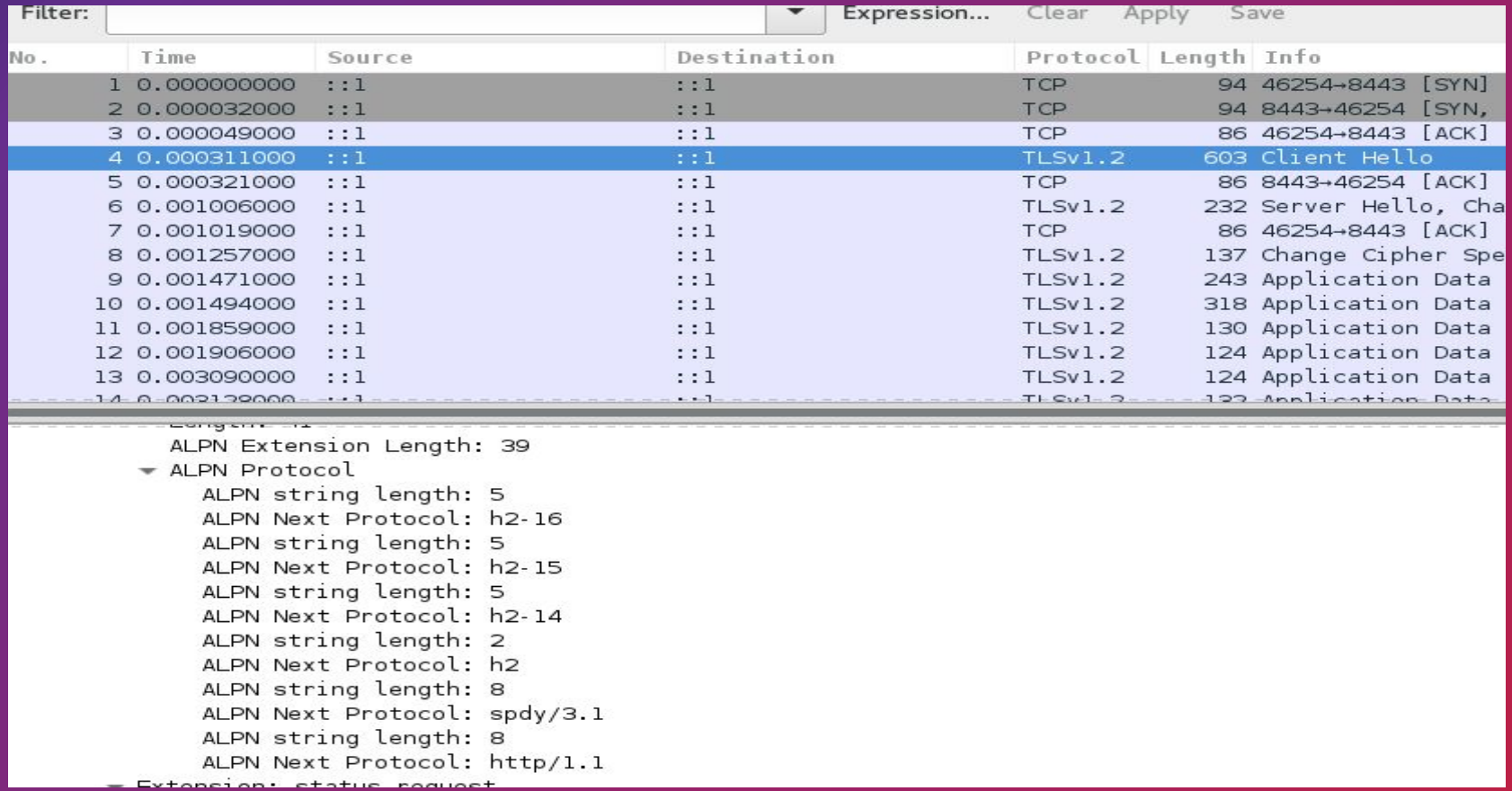
HTTP/2 : more

- HTTP headers compression
  - ~ 80 % save
- Request priority
  - Both sides
- Server Push
  - Prevent round trip to get element of a page
  - Faster / better rendering on browsers.

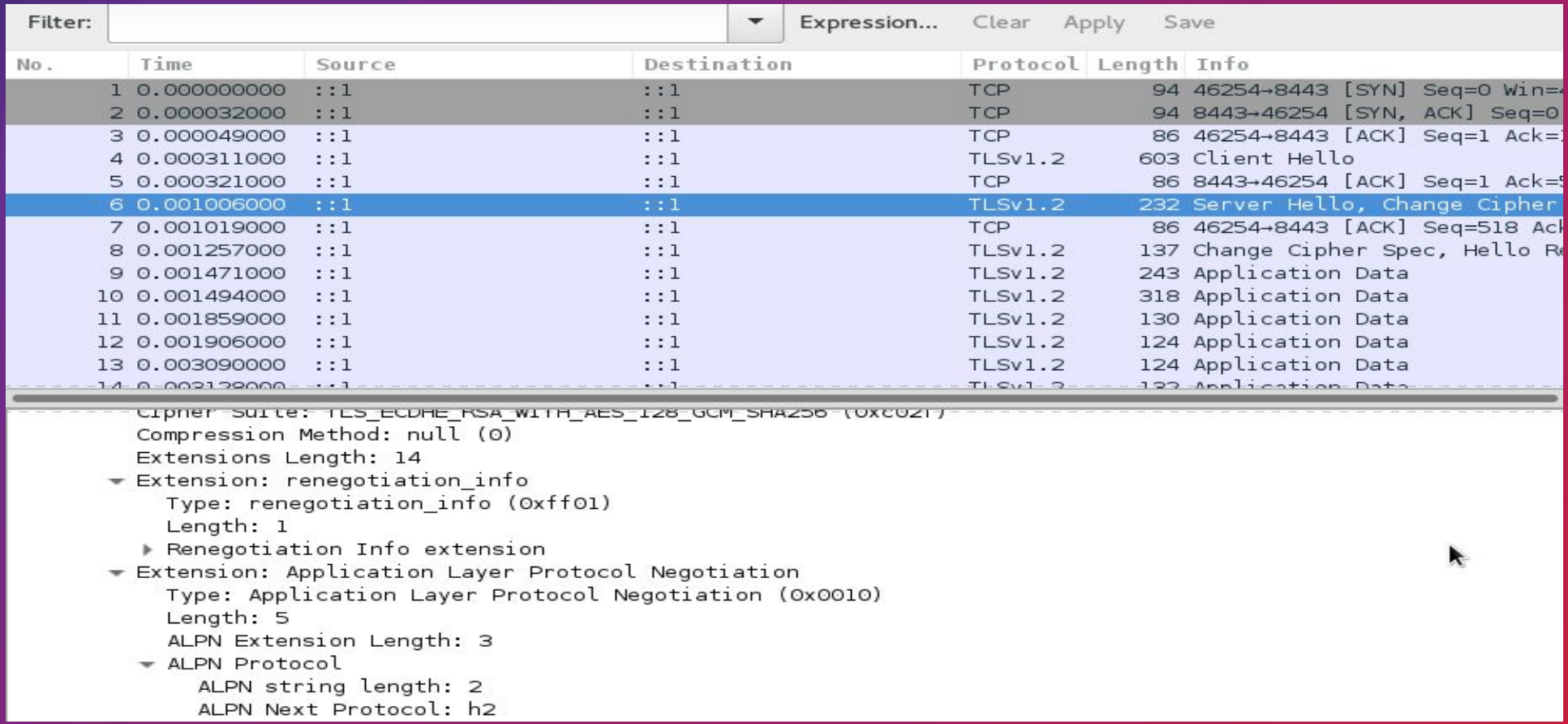
HTTP/2 With Browsers

- Browser with HTTP/2 and TLS
  - FireFox 34
  - Chrome 40 (with ALPN before was NPN)
  - IE 11
  - Opera and Safari 9
- → go for it now!

ALPN Client Hello (Firefox)



ALPN Server Hello (tomcat)



HTTP/2

- HTTP/2:
  - TCP/IP.
  - “safer” crypto is good but expensive.
  - No need to rewrite application to get the gains.

HTTP/2 : GO FOR IT

Then Why HTTP/3?

- TCP/IP:
  - Windows acks: 1 packet lost → all the channels blocked.
- UPD:
  - Channels are independent.
  - Need higher protocol level to insure integrity.
  - Packets might not be received in order.
- Security:
  - Need a patched version of OpenSSL (and use TLS-1.3)
  - UDP: cloud → no… but DNS → used everywhere!

HTTP/3 (RFC 9114 published June 2022)

- Use QUIC / TLS-1.3 / UDP
- To “transport” HTTP/1.1 like HTTP/2
- Initial connection TCP + Alt-Svc or HTTP/2
  - Response Alt-Svc: h3=":56666":
  - HTTP/2 ALTSVC frame
- problems:
  - UDP ports closed
  - UDP slower than TCP in Kernels
  - Needs extra CPU (?)
- Specifications:
  - RC 9114

Features: HTTP/2 vs HTTP/3

	HTTP/2	HTTP/3
Transport	TCP	UPD/QUIC
Streams	HTTP/2	QUIC
Clear text	yes (h2c: reverse proxy)	no
Independent streams	no	yes
Header compression	HPACK	QPACK
Server push	yes	yes
Early data	no	yes
0-RTT handshake	no (TLS-1.2)	Yes (TLS-1.3+)

HTTP/3 implementations

- quiche:
  - https://docs.quic.tech/quiche/
- Curl: https://curl.se/docs/http3.html
  - ngtcp2 (nghttp3/ngtcp2, patched openssl or GnuTLS)
  - quiche
  - msh3
  - In experimental at build time.
- Browser: chrome / firefox (active by default: Apr 2021).

HTTP/3 in our servers:

- Apache Tomcat: need time (wait for HTTP/3 streams?)
- Apache HTTPD: need time (probably like http/2)
- Traffic Server: in the 9.1.x experimental (need patched openssl)
  - See ATS docs / curl docs
  - 11-dev: boringSSL and quiche

TrafficServer / Configuration

- records.yaml
  - traffic_ctl config set proxy.config.http.server_ports "4443:quic" -c records.yaml
  - traffic_ctl config set proxy.config.udp.threads 1 -c records.yaml
  - traffic_ctl config set proxy.config.quic.initial_max_streams_bidi_in 100000
  - traffic_ctl config set proxy.config.quic.initial_max_streams_bidi_out 100000
- ssl_multicert.config:
  - **dest_ip=* ssl_cert_name=newcert.pem ssl_key_name=newkey.txt.pem**
- remap.config:
  - **map / http://127.0.0.1:8080**

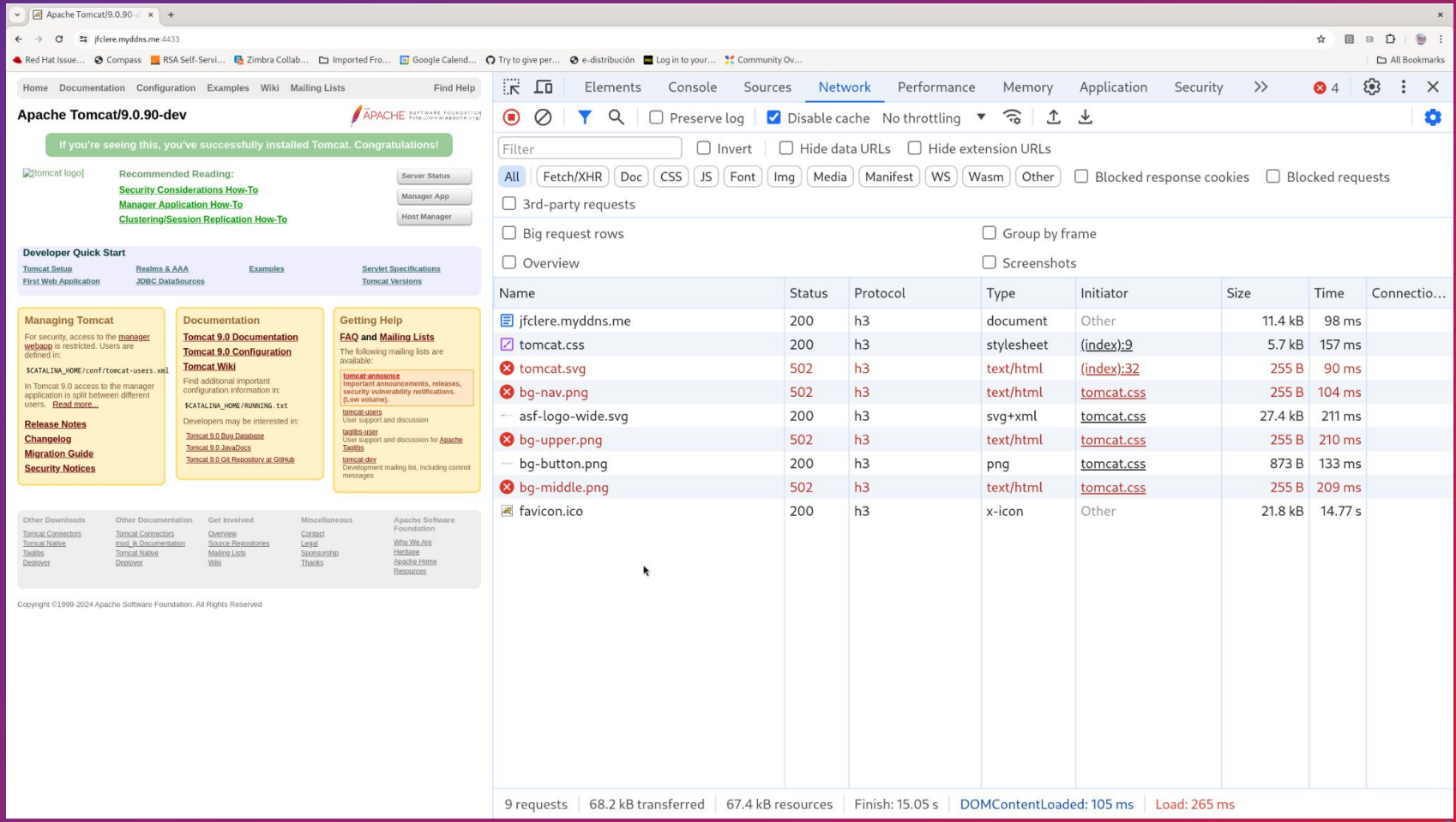
TrafficServer / H3 Demo

- Uses tomcat as backend
- Uses http/1.1 tomcat nio connector on 8080 as back-end.
- Uses Apache HTTPD https + mod_header to create the alt-svc

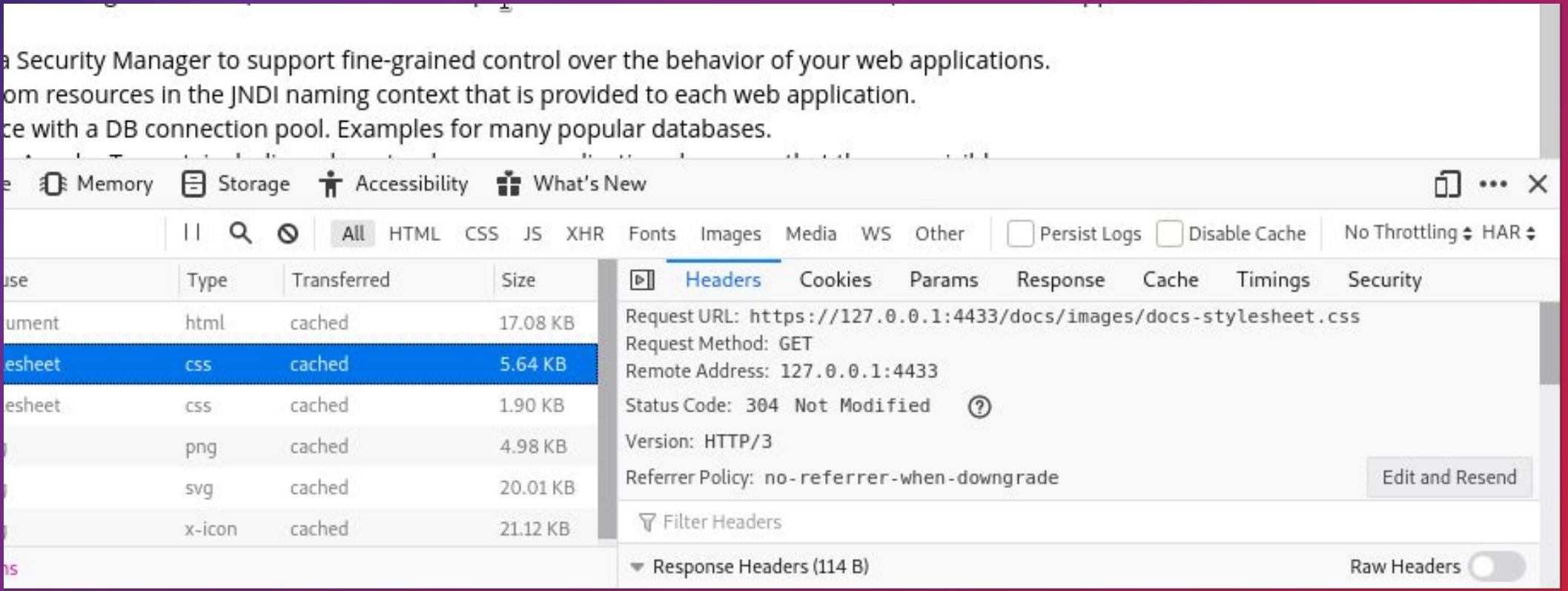
TrafficServer / Demo

- https://jfclere.myddns.me:4433/
- Response HTTP/1.1 (HTTP/2) header alt-svc
- alt-svc: h3=":4433"; ma=60; h3=":4433"; persist=1
- H3 (HTTP/3)
- ma=60 seconds = 1 minute.
- Next requests → HTTP/3

TrafficServer / Demo



TrafficServer / Demo



HTTP/3 more info:

- Playing with browsers:
  - Interop matrix
  - H3 activated by default since 2021 in Firefox/Chrome
- OpenSSL 3.3.x (3.2.x has a client QUIC API)

HTTP/3 openssl + nghttp3

- Basic client: (see also openssl one)
  - just testing.
  - using nghttp3 main. big callback and few functions
  - using openssl master to provide the QUIC layer.

    SSL *new_ssl = SSL_accept_stream(s, 0);

HTTP/3 openssl + nghttp3

- Basic server:
  - just testing.
  - using nghttp3 main. big callback and few functions
  - using openssl master to provide the QUIC layer.
    (and OpenSSL internal API for the moment)

HTTP/3 ready?

- Conclusion:
  - Not more a draft, last draft was H3-34.
  - UDP versus TCP.
  - Needs forked version of openssl… (0-RTT).
  - Or BoringSSL.
  - No need to rewrite application to get the gains.

HTTP/3 : wait

Questions?

COMMUNITY
OVER CODE

- jfclere@gmail.com
- users@tomcat.apache.org
- users@httpd.apache.org
- users@trafficserver.apache.org
- https://http2.github.io/ https://github.com/ngtcp2/nghttp3.git
- Client/Server: https://github.com/jfclere/openssl-h3-examples
- HTTP/3 see curl docs: http3-explained by Daniel
- More on HTP/3: https://github.com/jfclere/CoC23/tree/main/h3