HTTP/2, HTTP/3 the State of the Art in Our Servers

Jean-Frederic Clere @jfclere
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

- 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 - RFC7540
  - HPACK - Header Compression for HTTP/2 - RFC7541
- By the Internet Engineering Task Force
- ALPN Application-Layer Protocol Negotiation - RFC 7301
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)

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0000000000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>94</td>
<td>46254-8443 [SYN]</td>
</tr>
<tr>
<td>2</td>
<td>0.0000320000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>94</td>
<td>8443-46254 [SYN, ACK]</td>
</tr>
<tr>
<td>3</td>
<td>0.0000490000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>86</td>
<td>46254-8443 [ACK]</td>
</tr>
<tr>
<td>4</td>
<td>0.0000311000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>663</td>
<td>Client Hello</td>
</tr>
<tr>
<td>5</td>
<td>0.0000321000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>86</td>
<td>8443-46254 [ACK]</td>
</tr>
<tr>
<td>6</td>
<td>0.0010060000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>232</td>
<td>Server Hello, Change Cipher Spec</td>
</tr>
<tr>
<td>7</td>
<td>0.0010190000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>86</td>
<td>46254-8443 [ACK]</td>
</tr>
<tr>
<td>8</td>
<td>0.0012570000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>137</td>
<td>Change Cipher Spec</td>
</tr>
<tr>
<td>9</td>
<td>0.0014710000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>243</td>
<td>Application Data</td>
</tr>
<tr>
<td>10</td>
<td>0.0014940000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>318</td>
<td>Application Data</td>
</tr>
<tr>
<td>11</td>
<td>0.0018590000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>130</td>
<td>Application Data</td>
</tr>
<tr>
<td>12</td>
<td>0.0019060000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>124</td>
<td>Application Data</td>
</tr>
<tr>
<td>13</td>
<td>0.0030900000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>124</td>
<td>Application Data</td>
</tr>
<tr>
<td>14</td>
<td>0.0031290000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>123</td>
<td>Application Data</td>
</tr>
</tbody>
</table>

**ALPN Extension Length:** 39

**ALPN Protocol**
- ALPN string length: 5
- ALPN Next Protocol: h2-16
- ALPN string length: 5
- ALPN Next Protocol: h2-15
- ALPN string length: 5
- ALPN Next Protocol: h2-14
- ALPN string length: 2
- ALPN Next Protocol: h2
- ALPN string length: 8
- ALPN Next Protocol: spdy/3.1
- ALPN string length: 8
- ALPN Next Protocol: http/1.1

- Extensions: status request
## ALPN Server Hello (tomcat)

### Packet Details

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.00000000000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>94</td>
<td>46254-8443 [SYN] Seq=0 Win=52000</td>
</tr>
<tr>
<td>2</td>
<td>0.0000032000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>94</td>
<td>8443-46254 [SYN, ACK] Seq=0</td>
</tr>
<tr>
<td>3</td>
<td>0.0000049000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>86</td>
<td>46254-8443 [ACK] Seq=1 Ack=1</td>
</tr>
<tr>
<td>4</td>
<td>0.0000311000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>603</td>
<td>Client Hello</td>
</tr>
<tr>
<td>5</td>
<td>0.0000521000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>86</td>
<td>46254-8443 [ACK] Seq=1 Ack=1</td>
</tr>
<tr>
<td>6</td>
<td>0.00010000000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>232</td>
<td>Server Hello, Change Cipher Spec</td>
</tr>
<tr>
<td>7</td>
<td>0.0001019000</td>
<td>::1</td>
<td>::1</td>
<td>TCP</td>
<td>86</td>
<td>46254-8443 [ACK] Seq=518 Ack=1</td>
</tr>
<tr>
<td>8</td>
<td>0.0001257000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>137</td>
<td>Change Cipher Spec, Hello Request</td>
</tr>
<tr>
<td>9</td>
<td>0.0001471000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>243</td>
<td>Application Data</td>
</tr>
<tr>
<td>10</td>
<td>0.0001494000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>318</td>
<td>Application Data</td>
</tr>
<tr>
<td>11</td>
<td>0.0001598000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>130</td>
<td>Application Data</td>
</tr>
<tr>
<td>12</td>
<td>0.0001606000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>124</td>
<td>Application Data</td>
</tr>
<tr>
<td>13</td>
<td>0.0003001000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>124</td>
<td>Application Data</td>
</tr>
<tr>
<td>14</td>
<td>0.0003128000</td>
<td>::1</td>
<td>::1</td>
<td>TLSv1.2</td>
<td>123</td>
<td>Application Data</td>
</tr>
</tbody>
</table>

**Cipher Suite:** TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0x0021)

**Compression Method:** null (0)

**Extensions Length:** 14

- **Extension:** renegotiation_info
  - Type: renegotiation_info (0xff01)
  - Length: 1

  **Renegotiation Info extension**

- **Extension:** Application Layer Protocol Negotiation
  - Type: Application Layer Protocol Negotiation (0x0010)
  - Length: 5

  **ALPN Extension Length:** 3

  **ALPN Protocol**
  - ALPN string length: 2
  - ALPN Next Protocol: h2
Requirements

- OpenSSL for our 3 servers
  - At least 1.0.2c
- Tomcat (at least 9.x)
  - Tomcat-native or modern JVM
- Httpd (since 2.4.17)
  - HTTP/2 C Library (libnghttp2)
- TrafficServer (since ATS v5.3.2).
  - Nothing except openssl.
Status

- Tomcat (trunk/8.5)
  - Full support / released as stable.
  - Needs servlet 4.0 (JSR 369) for server PUSH API (TC 11.0.x)
  - Can't be full JAVA until JDK9 (ALPN support)
- Httpd (available since 2.4.17)
  - Full support (since 2.4.20)
- TrafficServer (since 5.3.0) (flow control 6.1)
  - Priorities (6.2.0) and Server PUSH (7.0.0)
<Connector
   port="8002"
   MaxThreads="150"
   SSLEnabled="true">
   <SSLHostConfig>
      <Certificate
         certificateFile="/home/jfclere/H3/certs/pubcert.pem"
         certificateKeyFile="/home/jfclere/H3/certs/privkey.pem"/>
   </SSLHostConfig>
</Connector>
Tomcat / Performances

Concurency 240

- coyote_nio_isse_h1_https
- coyote_nio_isse_h2_https
Tomcat / Performances

![Concurrency 240 Diagram]

- **CPU Usage**
- **File Size**: 4KiB, 8KiB, 16KiB, 32KiB, 64KiB, 128KiB, 256KiB, 512KiB, 1MiB
- Lines represent:
  - `coyote_nio_jsse_h1_https`
  - `coyote_nio_jsse_h2_https`
No server push (anyway the browsers stop supporting it :-()

Multiplexing

headers compression

HTML page:
- That requires a lot (~100) of (~4Kbytes) images to render.
- use a servlet
TrafficServer / Configuration

- records.yaml
  - traffic_ctl config set proxy.config.http.server_ports "8888:ssl" -c records.yaml

- 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

- ip_allow.config:
  - src_ip=192.168.1.38 action=ip_allow method=ALL
TrafficServer / Demo

- Like tomcat one
- Uses http/1.1 tomcat nio connector on 8080 as back-end.
HTTPd / Configuration

- httpd.conf:

  LoadModule h2_module modules/mod_h2.so
  Listen 8006
  <VirtualHost *:8006>
    Protocols h2 http/1.1
    ProtocolsHonorOrder on
    SSLEngine on
    SSLCertificateFile "/home/jfclere/CERTS/newcert.pem"
    SSLCertificateKeyFile "/home/jfclere/CERTS/newkey.pem"
    SSLCACertificateFile "/etc/pki/CA/cacert.pem"
  </VirtualHost>
HTTPd / Configuration proxy

- httpd.conf:

  LoadModule http2_module modules/mod_http2.so
  LoadModule proxy_http2_module modules/mod_proxy_http2.so
  Listen 8006
  <VirtualHost *:8006>
    Protocols h2 http/1.1
    ProtocolsHonorOrder on
    SSLEngine on
    ...
    ProxyPass "/" "h2c://localhost:8003/"
  </VirtualHost>
HTTPd / Demo

- Like the tomcat one:
  - htdocs/http2.html
  - htdocs/images/ the images.
HTTP/2 move to it?

Conclusion:
- Using HTTP/2 without PUSH is already good.
- “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:
  - RFC 9114
<table>
<thead>
<tr>
<th>Feature</th>
<th>HTTP/2</th>
<th>HTTP/3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transport</td>
<td>TCP</td>
<td>UPD/QUIC</td>
</tr>
<tr>
<td>Streams</td>
<td>HTTP/2</td>
<td>QUIC</td>
</tr>
<tr>
<td>Clear text</td>
<td>yes (h2c: reverse proxy)</td>
<td>no</td>
</tr>
<tr>
<td>Independent streams</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Header compression</td>
<td>HPACK</td>
<td>QPACK</td>
</tr>
<tr>
<td>Server push</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Early data</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>0-RTT handshake</td>
<td>no (TLS-1.2)</td>
<td>Yes (TLS-1.3+)</td>
</tr>
</tbody>
</table>
HTTP/3 implementations

- quiche:
  - [https://docs.quic.tech/quiche/](https://docs.quic.tech/quiche/)
- Curl: [https://curl.se/docs/http3.html](https://curl.se/docs/http3.html)
  - ngtcp2 ([nghttp3/ngtcp2](https://nghttp3.ngtcp2.io), patched openssl or GnuTLS)
  - quiche
  - msh3
  - In experimental at build time.
- Browser: [chrome](https://www.google.com/chrome) / firefox (active by default: Apr 2021).
HTTP/3 in our servers:

- Apache Tomcat: Problem UDP socket API incomplete (java 15)
- Apache HTTPD: need time probably like http/2
- Traffic Server: in the 9.1.x experimental (need patched openssl)
  - See ATS docs / curl docs
  - 10-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, h3-29=":4433"; ma=60

- H3-29 (HTTP/3 draft 29)
  
  ma=60 seconds = 1 minute.

- Next requests → HTTP/3
Apache Tomcat 9
Version 9.0.27-dev, Oct 29 2019

Documentation Index

Introduction
This is the top-level entry point of the documentation bundle for the Apache Tomcat Servlet/JSP container. Apache Tomcat version 9.0 implements the Servlet 4.0 and JavaServer Pages 2.3 specifications from the Java Community Process, and includes many additional features that make it a useful platform for developing and deploying web applications and web services.

Select one of the links from the navigation menu (to the left) to drill down to the more detailed documentation that is available. Each available manual is described in more detail below.

Apache Tomcat User Guide

The following documents will assist you in downloading and installing Apache Tomcat, and using many of the Apache Tomcat features.

1. Introduction - A brief, high level, overview of Apache Tomcat.
2. Setup - How to install and run Apache Tomcat on a variety of platforms.
3. First web application - An introduction to the concepts of a web application as defined in the Servlet Specification. Covers basic organization of your web application source tree, the structure of a web application archive, and an introduction to the web application deployment descriptor ([/WEB-INF/web.xml]).
4. Deployer - Operating the Apache Tomcat Deployer to deploy, precompile, and validate web applications.
5. Manager - Operating the Manager web app to deploy, undeploy, and redeploy applications while Apache Tomcat is running.
6. Host Manager - Operating the Host Manager web app to add and remove virtual hosts while Apache Tomcat is running.
7. Realms and Access Control - Description of how to configure realms (databases of users, passwords, and their associated roles) for use in web applications that utilize container Managed Security.
8. Security Manager - Configuring and using a Java Security Manager to support fine-grained control over the behavior of your web applications.
9. JNDI Resources - Configuring standard and custom resources in the JNDI naming context that is provided to each web application.
10. JDBC DataSource - Configuring a JNDI DataSource with a DB connection pool. Examples for many popular databases.
TrafficServer / Demo

A Security Manager to support fine-grained control over the behavior of your web applications. It provides a Security Manager to support fine-grained control over the behavior of your web applications.

From resources in the JNDI naming context that is provided to each web application. This resource is associated with a DB connection pool. Examples for many popular databases.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Transferred</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>sheet</td>
<td>css</td>
<td>cached</td>
<td>5.64 KB</td>
</tr>
<tr>
<td>sheet</td>
<td>css</td>
<td>cached</td>
<td>1.90 KB</td>
</tr>
<tr>
<td></td>
<td>png</td>
<td>cached</td>
<td>4.98 KB</td>
</tr>
<tr>
<td></td>
<td>svg</td>
<td>cached</td>
<td>20.01 KB</td>
</tr>
<tr>
<td></td>
<td>x-icon</td>
<td>cached</td>
<td>21.12 KB</td>
</tr>
</tbody>
</table>

Request URL: https://127.0.0.1:4433/docs/images/docs-stylesheet.css
Request Method: GET
Remote Address: 127.0.0.1:4433
Status Code: 304 Not Modified
Version: HTTP/3
Referrer Policy: no-referrer-when-downgrade

Edit and Resend

Response Headers (114 B)
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:
  - 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 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?

Thank you!

- jfclere@gmail.com
- users@tomcat.apache.org
- users@httpd.apache.org
- users@trafficserver.apache.org
- Client tries: https://github.com/jfclere/openssl-h3-examples
- HTTP/3 see curl docs: http3-explained by Daniel