Apache Tomcat NEXT

Progress Report

Jean-Frederic Clere, Manager, Red Hat

AGENDA

- Who I am
- New features from specifications
- Tomcat specific new features
- Tomcat features removed
- Internal changes
- Why Apache Tomcat 8.5?
- HTTP/2 and ALPN
- SNI
- OpenSSLImplementation
- Migration from 8.0 to 8.5
- Get involved
- 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)

Tomcat

Tomcat versions

Tomcat	Java EE	Minimum Java SE	Servlet	JSP	EL	WebSocket	JASPIC	1st Stable Release	EOL
5.x	4	1.4	2.4	2.0	N/A	N/A	N/A	08 2004	09 2012
6.x	5	5	2.5	2.1	2.1	N/A	N/A	02 2007	12 2016
7.x	6	6	3.0	2.2	2.2	1.1	N/A	01 2011	TBD
8.0.x	7	7	3.1	2.3	3.0	1.1	N/A	02 2014	xx 2016?
8.5.x	7	7	3.1	2.3	3.0	1.1	1.1	06 2016	TBD
9.x	8	8	4.0	2.4?	3.1?	2.0?	1.1?	2017	TBD

xx: was 09 in June ;-) 8.0.38 released 11 October

New features from specifications

JavaEE 8

- Key elements
  - HTTP/2
  - Simplification
  - Better integration for managed beans
  - Better infrastructure for the cloud

Specifications

Servlet 4.0

- HTTP/2
- Usability improvements
  - HttpFilter, default methods
- Clarifications
- Enhancement requests

Specifications

HTTP/2

- HTTP/2 requires some TLS features
  - Server Name Indication (SNI)
  - Application Layer Protocol Negotiation (ALPN)
- Full support
  - 8.5.3 considered stable. (since June 2016)
- h2c available (for proxies)
- h2 requires APR/native/OpenSSL due to ALPN requirements
- Server push available

Specifications

Servlet 4.0 HTTP/2

- Java EE 8 must run on Java 8
- Java EE 8 requires Servlet 4.0
- Servlet 4.0 requires HTTP/2
- HTTP/2 requires ALPN
- Java 8 does not support ALPN
- ALPN support will be available in Java 9
- ALPN support will likely be backported to Java 8 at some point...

Specifications

Other

- WebSocket 1.2 (keep 1.1?)
  - Standard extension for compression/multiplexing?
- JSP 2.4 (keep 2.3?)
  - Imports to clarify (EL 3.0 related)
- EL 3.1 (keep 3.0?)
  - Only minor improvements/clarifications needed
- JASPIC 1.1 (New!)
  - Java Authentication Service Provider Interface for Containers. Used to support Oauth (login)

Tomcat New Features

TLS support improvements (1)

- Major rewrite of TLS support
- Tomcat 8 supports
  - one TLS virtual host per connector
  - one certificate per virtual host
- Tomcat 9 supports
  - multiple virtual hosts per connector (SNI)
  - multiple certificates per virtual host
- TLS configuration has changed to support this

Tomcat New Features

TLS support improvements (2)

- SNI and multiple certificates supported by all connectors
  - APR/native support via the OpenSSL API
  - JSSE support via parsing the initial handshake
- ALPN supported by APR/native or OpenSSLImplementation
  - JSSE support is currently TBD
- Common (where possible) configuration for all connectors
  - Some JSSE / OpenSSL differences remain.
  - OpenSSL engine option of NIO and NIO2 connectors
    - Allows OpenSSL performance with NIO/NIO2 APIs
    - Use automatically when tc-native is installed.

Tomcat Removed Features

Old blocking O/I connectors...

- BIO HTTP and BIO AJP connectors
  - Websocket and Servlet 3.1 require non-blocking IO
  - Emulation of non-blocking is bad:
    - Complex
    - Not scalable
    - Risky: stuff that might break.
    - Decision remove them.
- Still 3 connectors:
  - NIO default connector
  - NIO2 introduced in Tomcat 8.0
  - APR/Native still available. (requires tomcat-native libraries)

Tomcat Removed Features

Comet

- Proprietary interface for asynchronous I/O
- Users are moving (have moved) to WebSocket
- Adds complexity to all the connectors
- Therefore decided to remove it

Internal Changes

Connectors

- Removed
  - BIO
  - Comet
- Reduce duplication
  - HTTP upgrade from 12 classes to 3
  - HTTP/1.1 cleanup = removed ~ 50% (~2500 loc)
  - AJP 1.3 cleanup = remove ~ 30%
- No connector specific HTTP/2 code
- Implementation specific per connector → Endpoint
- Implementation specific per connection → SocketWrapper

Internal Changes

Websocket

- Refactored I/O implementation
  - Direct to Tomcat's I/O layer
  - Not via Servlet 3.1 non-blocking API
- Simpler
- Faster
- Extension support likely to require further refactoring?

Internal Changes

Other

- Remove use of system properties for configuration
  - Move to per Context / Host / Server / Connector
  - keep the system property as a default
- Made RFC 6265 CookieProcessor the default
  - Note UTF-8 extension

Why Tomcat 8.5?

EE8 late...

- Tomcat 9 stable release is tied to the release of Java EE 8
- Java EE 8 has been repeatedly delayed
  - Currently delayed until at least H1 2017
- Don't want users to have to wait another year+ to get access our new features:
  - HTTP/2
  - OpenSSL encryption for JSSE
  - TLS virtual hosting
  - JASPIC
- Hence, Tomcat 8.5...

What is Tomcat 8.5?

Tomcat 9.0.0.M4…

- Started from Apache Tomcat 9.0.0M4
- Reverted all Servlet 4.0 API changes
- Reworked code that required Java 8
- Tomcat specific Push Server API
- Configuration compatible with 8.0.x
- “big” removal:
  - Comet (migrate to WebSocket)
  - BIO (Connector… probably not noticed)

Tomcat 8.5 timing

Possible roadmap

- ~6 months of 8.0.x and 8.5.x
  - Extended if needed.
- ~ one month between releases
- ~ after no more 8.0.x releases
- First 8.5 release 24 March 2016
- Current release: 8.5.6 stable
- Expect last 8.0.x soon: no date yet!

Why HTTP/2

- HTTP/1.1: June 1999 (RFC 2616)
  - 1999:

    1 page ~ 1kB HTML

    2015:

    1 page ~ 3MB HTML + IMAGES + JS + CSS etc

  Protocol:
  - Not adapted / inefficient / etc

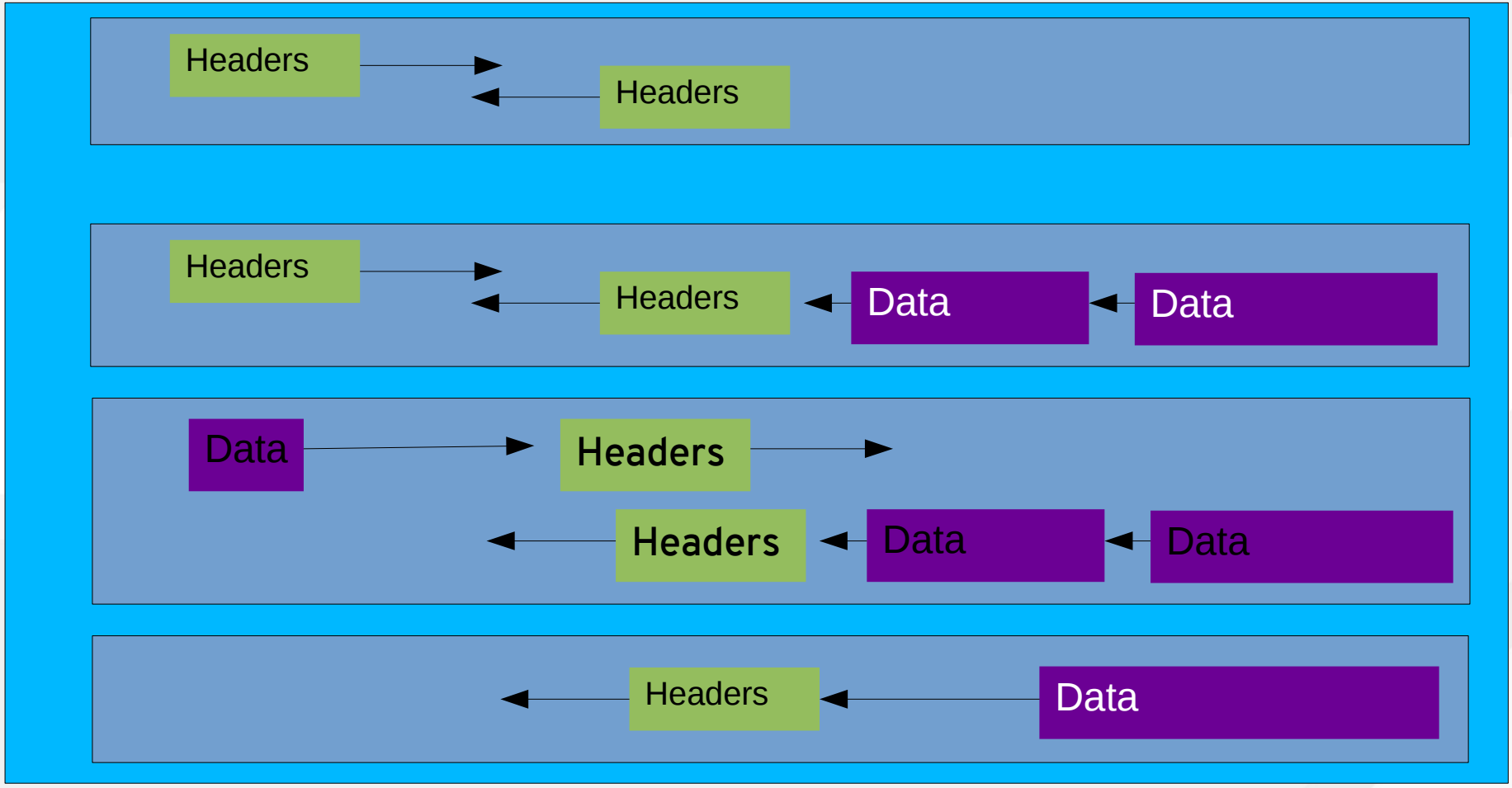
HTTP/2 general

- HTTP/2:
  - Binary
  - Frame
  - Multiplex
  - Based on SPDY
  - TLS everywhere:
    - Browers 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 Multiplexed



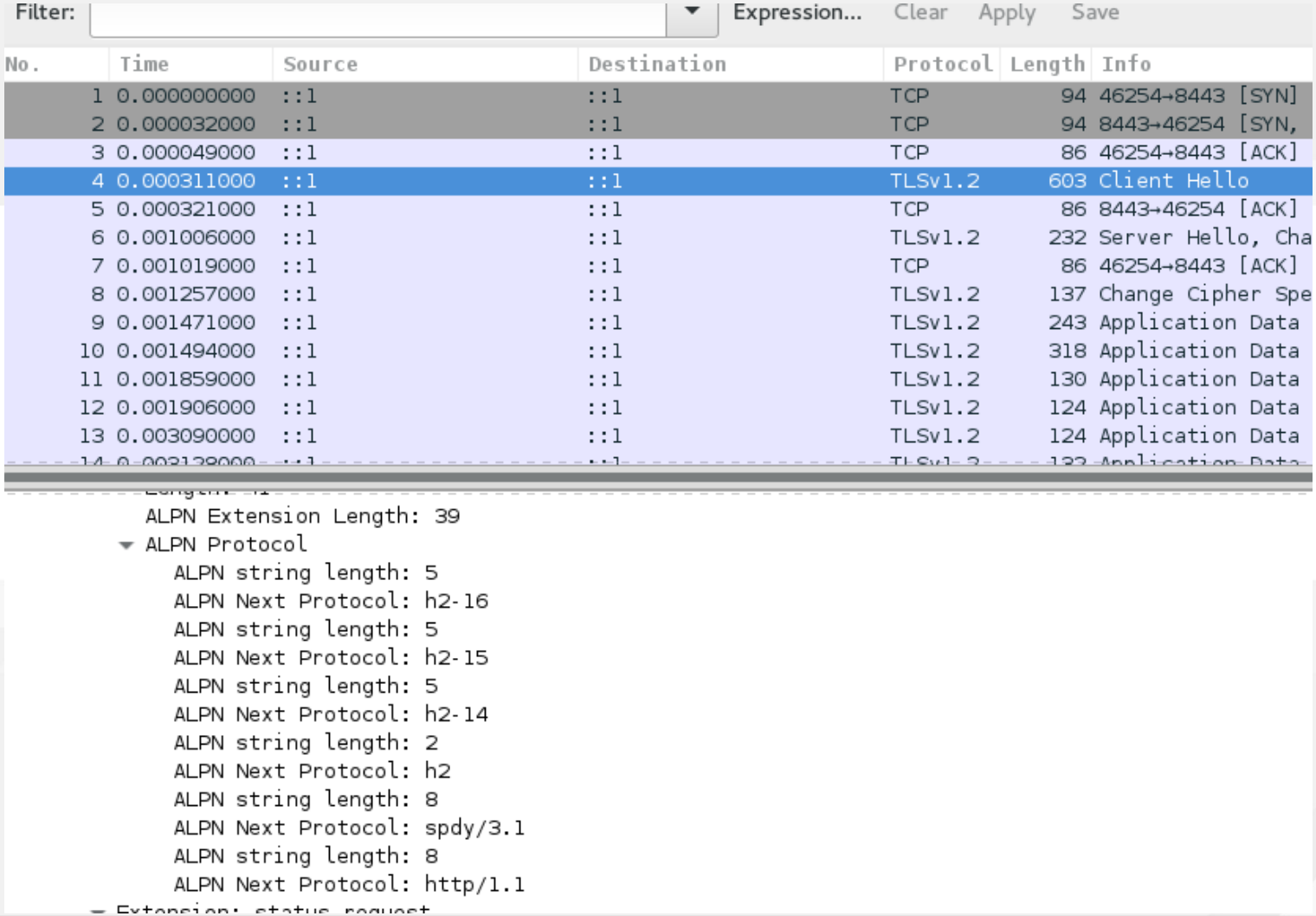
HTTP/2 : more

- HTTP headers compression
  - ~ 80 % saved
- Request priority
  - Both sides
- Server Push
  - Prevents round trips to get page elements.
  - Faster / better rendering on browsers.

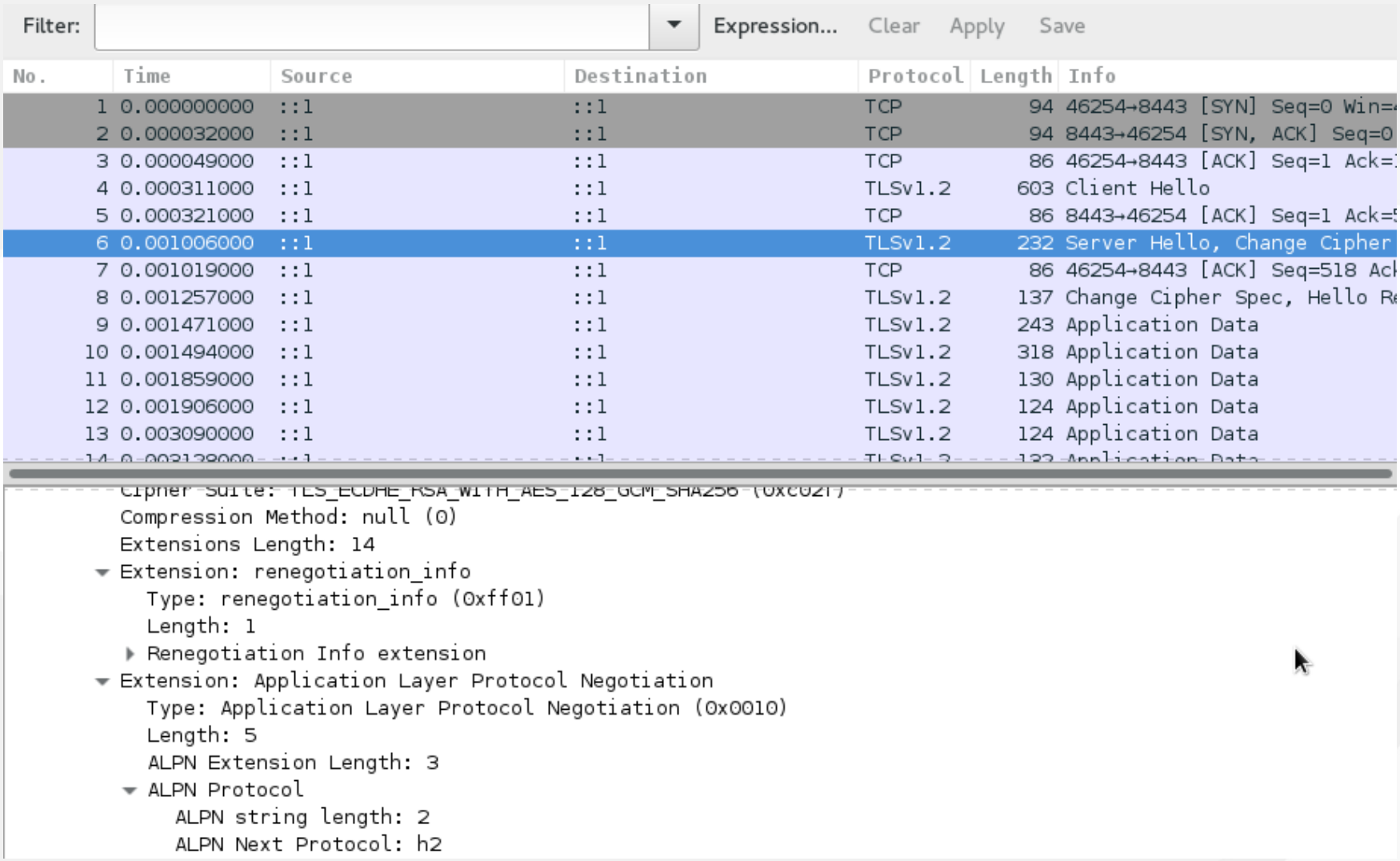
HTTP/2 When Browsers

- Browser with HTTP/2 and TLS
  - FireFox 34
  - Chrome 40 (with ALPN before was NPN)
  - IE 11
  - Opera and Safari 9
- Stats from docs.trafficserver and ci.trafficserver:
  - More than 50% is over HTTP/2 (data from April)
- → go for it now!

ALPN Client Hello (Firefox)



ALPN Server Hello (tomcat)



TC connector server.xml

```
<Connector
    port="8002"
   scheme="https"
   SSLEnabled="true"
   ciphers="TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256"
   SSLCertificateFile="/home/jfclere/CERTS/newcert.pem"
   SSLCertificateKeyFile="/home/jfclere/CERTS/newkey.txt.pem"
   protocol="org.apache.coyote.http11.Http11AprProtocol">
     <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
<Connector/>
<Connector port="8003" protocol="HTTP/1.1"
        SSLEnabled="true" scheme="https" secure="true"
        keystoreFile="conf/.keystore" keystorePass="changeit"
        socket.directBuffer="true" socket.directSslBuffer="true">
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
</Connector>
<Connector
   port="8004"
   protocol="org.apache.coyote.http11.Http11AprProtocol">
   <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
</Connector>
```

Tomcat / configuration

In bin/setenv.sh:

LD_LIBRARY_PATH=/home/jfclere/tomcat-native/native/.libs

export LD_LIBRARY_PATH

And the libtcnative-1.so linked with openssl-1.0.2c, checking with ldd:

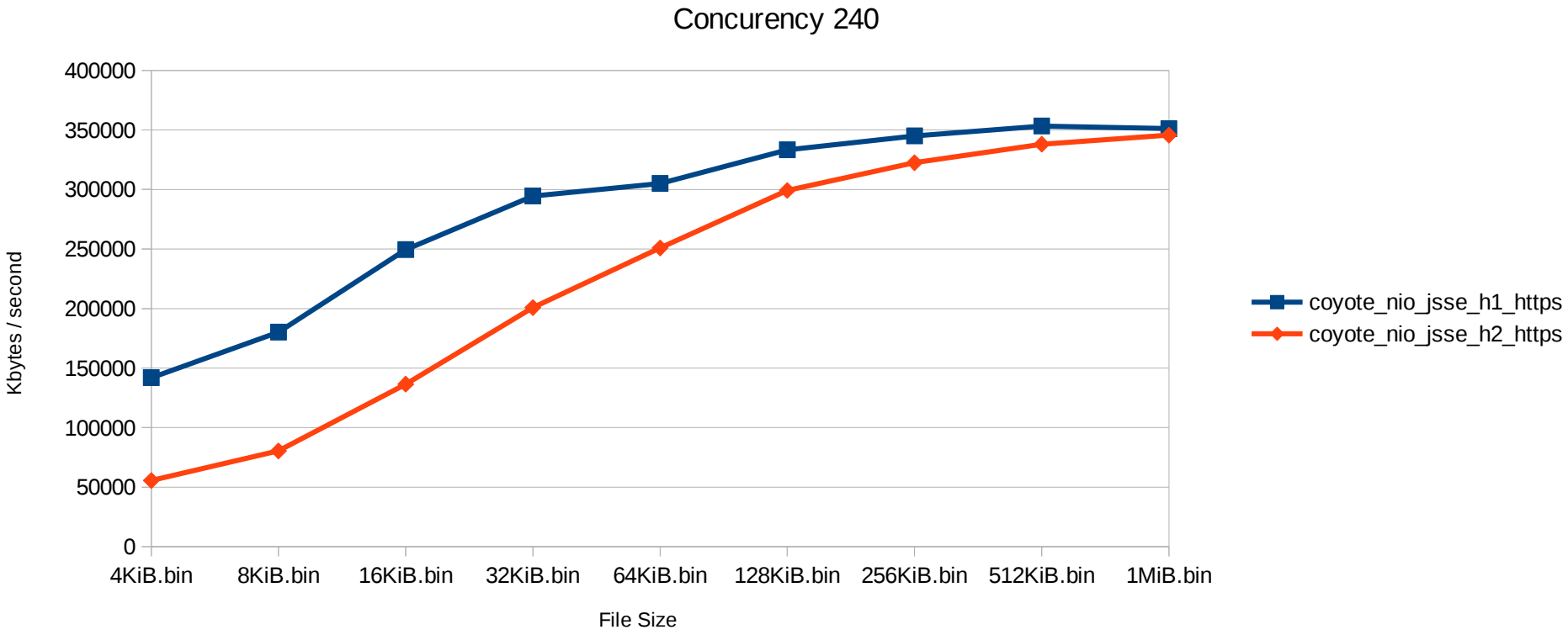
libssl.so.1.0.0 => /home/jfclere/OPENSSL-1.0.2c/lib/libssl.so.1.0.0 (0x00007f6ab147b000)

libcrypto.so.1.0.0 => /home/jfclere/OPENSSL-1.0.2c/lib/libcrypto.so.1.0.0 (0x00007f6ab1028000)

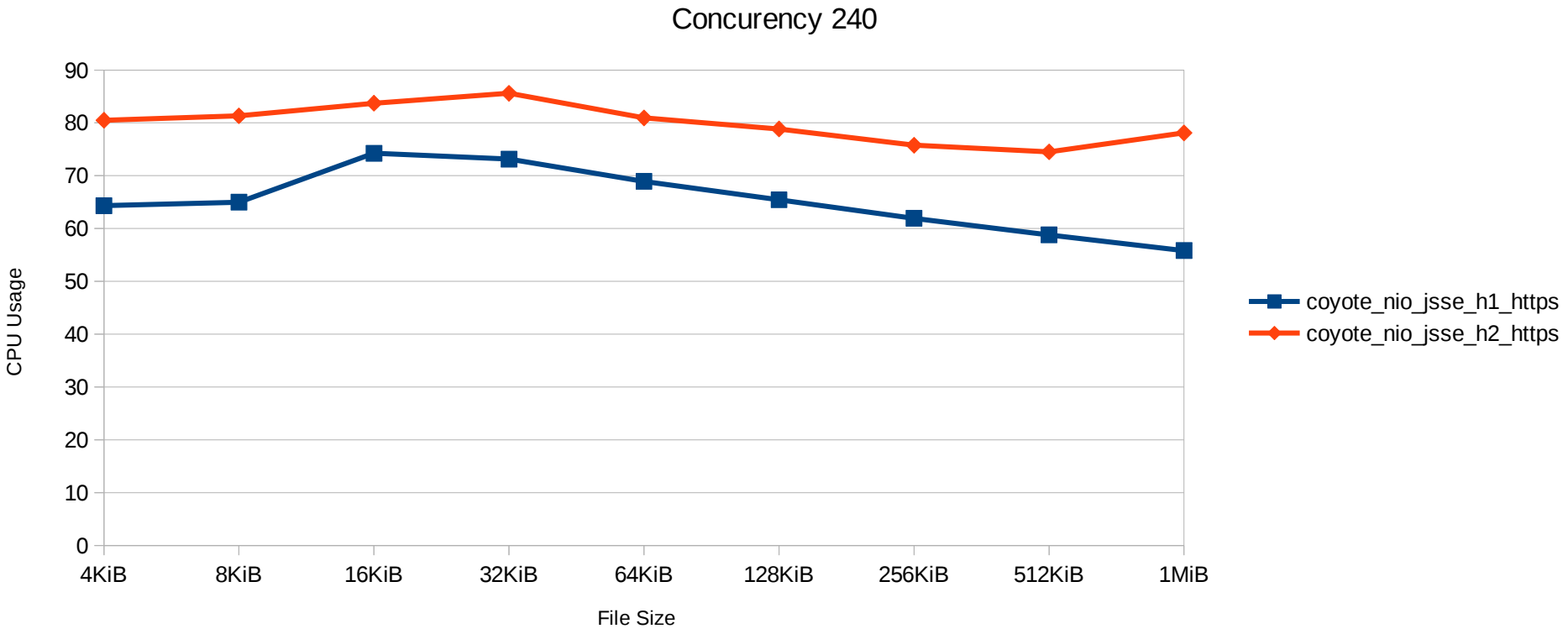
libapr-1.so.0 => /home/jfclere/APR-1.4.x/lib/libapr-1.so.0 (0x00007f6ab0dfa000)

Usually the openssl of recent distribution (fedora 23) will work.

Tomcat / Performances



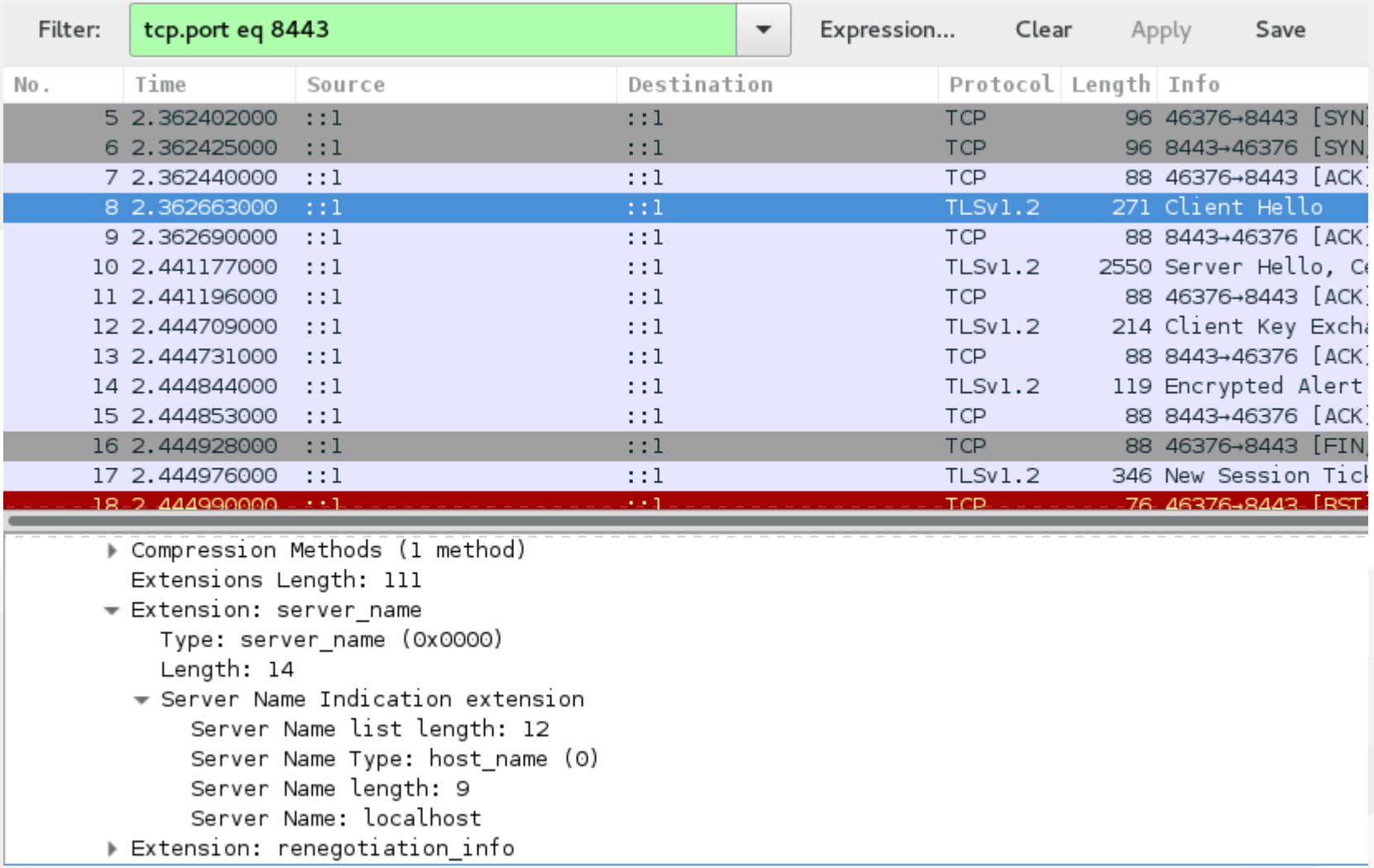
Tomcat / Performances



Tomcat / Demo

- No server push (may be change it: SimpleImagePush)
- Multiplexing
- headers compression
- Page html page:
  - That requires a lot (~1000) of (~4Kbytes) images to render.

SNI Client Hello (Firefox)



TC connector server.xml

```
<Connector protocol="org.apache.coyote.http11.Http11AprProtocol"
      SSLEnabled="true" maxThreads="150" secure="true" scheme="https"
      defaultSSLHostConfigName="local1.com" port="8443" >
      <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
 <SSLHostConfig honorCipherOrder="false" hostName="local1.com" >
  <Certificate certificateKeyFile="conf/local1.key"
         certificateFile="conf/local1.crt"
         type="RSA" />
 </SSLHostConfig>
 <SSLHostConfig honorCipherOrder="false" hostName="local2.com">
  <Certificate certificateKeyFile="conf/local2.key"
           certificateFile="conf/local2.crt"
           type="RSA" />
 </SSLHostConfig>
</Connector>
```

Tomcat / Demo

- 2 pairs of key/certificate
  - local1.com
  - local2.com
- /etc/hosts
  - 127.0.0.1 localhost local1.com local2.com
- SNI allows to select the right key/certificate

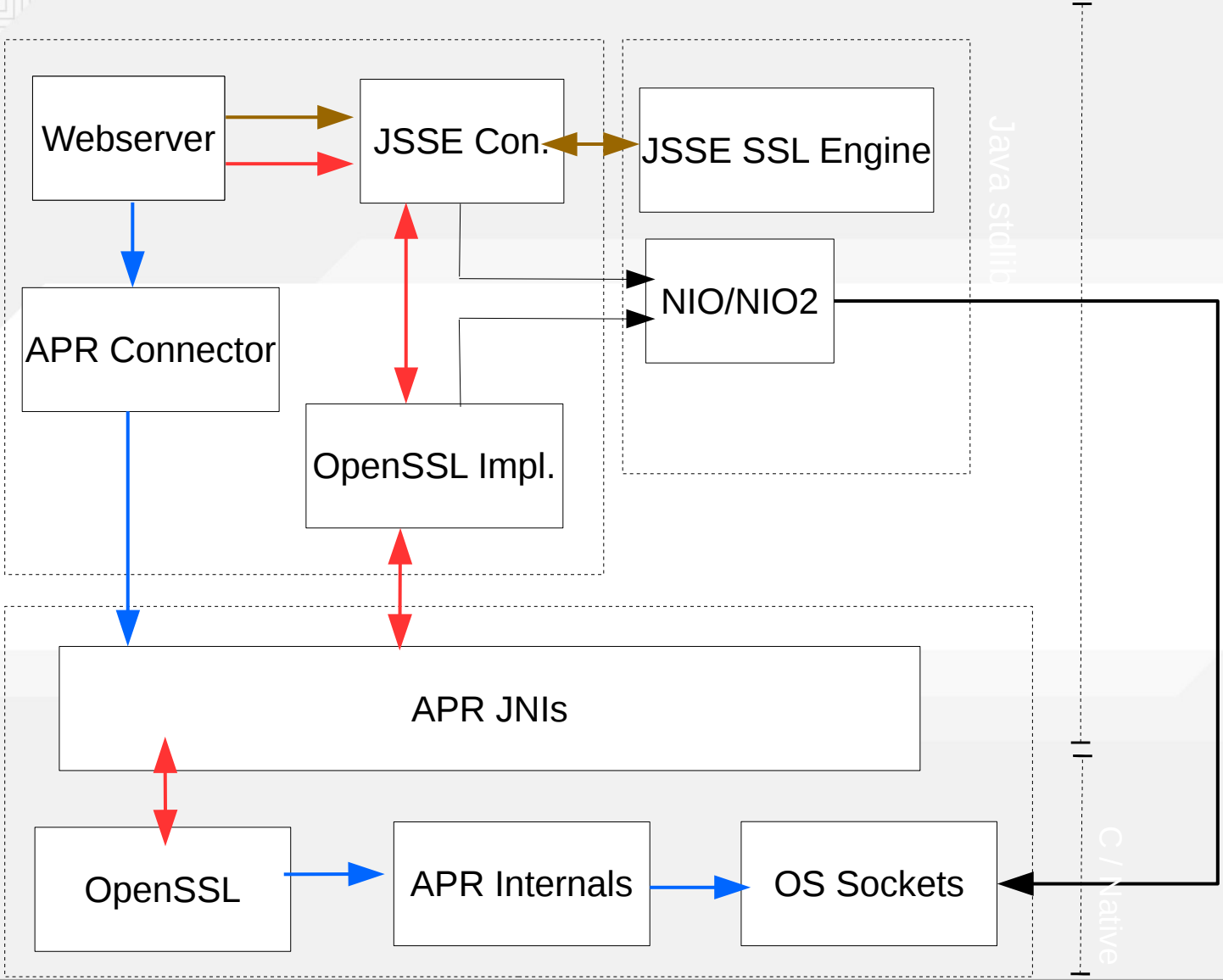
Why a new SSLImplementation

- JSSE:
  - Very slow
  - Missing features: like ALPN (JEP 244: TLS Application-Layer Protocol Negotiation)
  - Hardware acceleration used to be very partial (like AES in early java8)
- Native connector:
  - Fast but a lot of native code
  - Use OpenSSL for SSL/TLS.
- New OpenSSL implemetation:
  - Fast.
  - Uses only a OpenSSL for native code (no native socket, poller etc).
  - Works with NIO and NIO2.
  - Uses OpenSSL for SSL/TLS. (warp, unwarp, handshake etc).

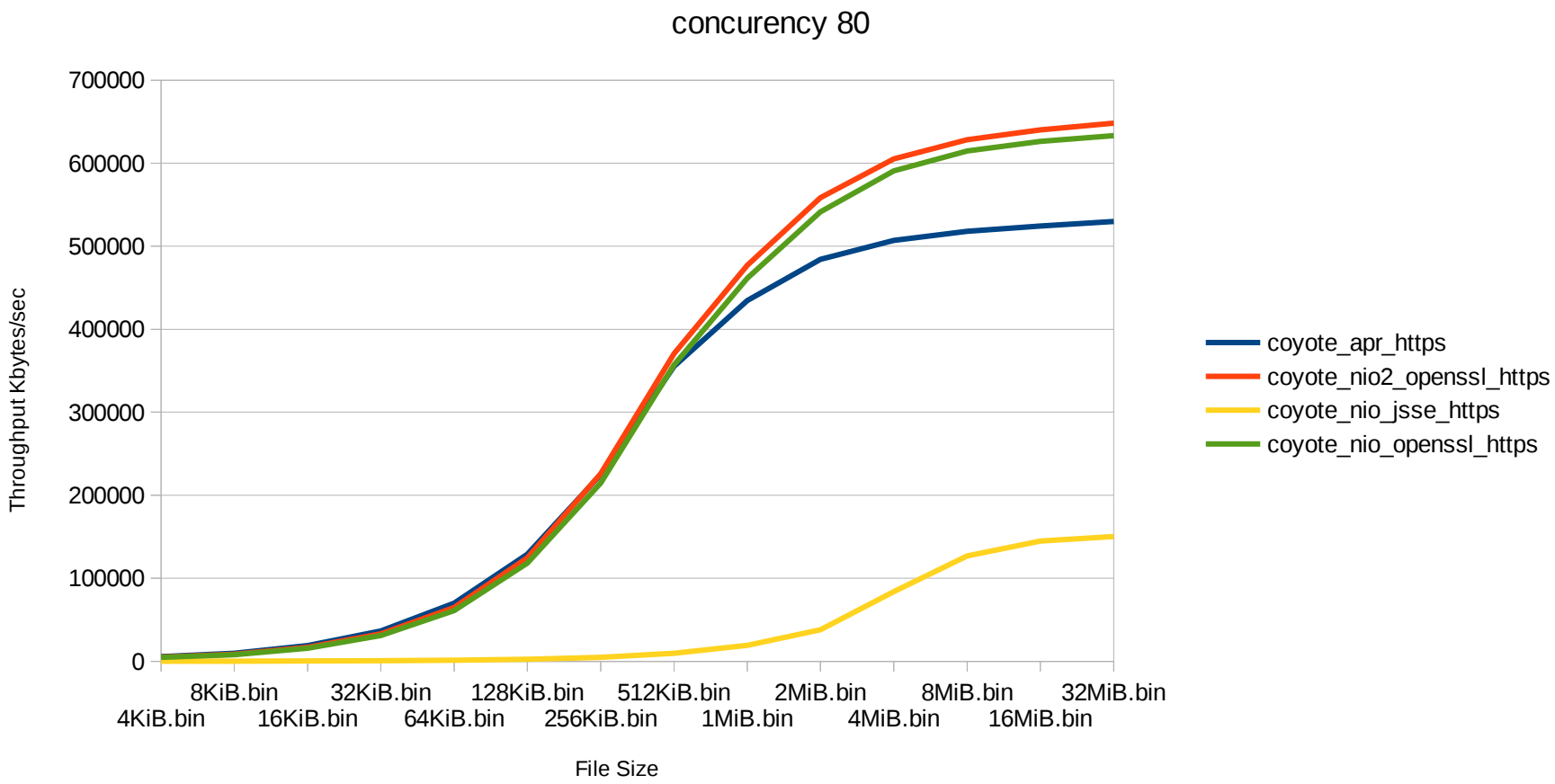
OpenSSLImplementation

- Code orginates from netty-tcnative a forked Tomcat Native
- Prototype (2015):
  - Done with the BeFriNe University
  - Tested and ported to tc_trunk last summer
- SSL Configuration compatible with the JSSE configuration style (*)
- Uses keystores (*)
- Uses OpenSSL BIO to wrap/unwarp, handshake
- Uses java NIO or NIO2 Sockets for the reads and writes
- Automatically enabled when TC native is installed/enabled (*)

How TLS is done in Tomcat



Connector Throughput (c80)



TC connector server.xml

OLD NATIVE CONNECTOR WAY:

```
<Connector
   port="8002"
   scheme="https"
   SSLEnabled="true"
   SSLCertificateFile="/home/jfclere/CERTS/newcert.pem"
   SSLCertificateKeyFile="/home/jfclere/CERTS/newkey.txt.pem"
   protocol="org.apache.coyote.http11.Http11AprProtocol">
    <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
<Connector/>
```

NEW OPENSSLImplementation WAY: (AprLifecycleListener" with SSLEngine="on" + tcnative libs)

```
<Connector port="8003" protocol="HTTP/1.1"
        SSLEnabled="true" scheme="https" secure="true"
        keystoreFile="conf/.keystore" keystorePass="changeit"
        socket.directBuffer="true" socket.directSslBuffer="true">
        <UpgradeProtocol className="org.apache.coyote.http2.Http2Protocol" />
</Connector>
```

Migration from Apache Tomcat 8.0.x

- Aiming to make it a seamless process for most users
  - Some users will have some work to do
- Configuration files can be re-used
  - Will need migration to use new TLS features
- Some removed features will not be replaced
  - Comet (Stick with final 8.0, revert 7.0 or migrate WebSocket)
- Work arounds may be added for some removed features
  - BIO
- Removed deprecated code may be restored
  - Manager, Context, RealmBase

GET INVOLVED

Help is welcomed ;-)

- SVN:
  - http://svn.apache.org/repos/asf/tomcat/tc8.5.x/trunk/
  - http://svn.apache.org/repos/asf/tomcat/trunk/
- MAIL LISTS:
  - dev@tomcat.apache.org Dev list.
  - users@tomcat.apache.org Users list.
- WIKI:
  - http://wiki.apache.org/tomcat/FrontPage

APACHECON
Europe

16-18.11.16

SEVILLE, SPAIN

THANK YOU

jfclere@gmail.com

THE LINUX FOUNDATION