HTTP, WebSocket, SPDY, HTTP/2.0

Evolution of Web Protocols

Simone Bordet
sbordet@intalio.com

Thomas Becker
tbecker@intalio.com
Intalio|Jetty
- Services, Training and Support for Jetty and CometD

Intalio|BPMS
- Business Process Management System

Intalio|Create
- The modern way to build business applications

http://intalio.com
Who are we

Simone Bordet
sbordet@intalio.com
@simonebordet

- Open Source Contributor
  Jetty, CometD, MX4J, Foxtrot, LiveTribe, JBoss, Larex
- Lead Architect at Intalio/Webtide
  SPDY and HTTP client lead
- CometD project leader
  Web messaging framework
- JVM tuning expert
- Runner, triathlete

Thomas Becker
tbecker@intalio.com
@tbecker00

- Open Source Contributor
  Jetty, Grails
- Senior Engineer at Intalio/Webtide
  Jetty's SPDY maintainer
- JVM tuning expert
- Mountain biker, climber
Agenda

- Some History of Web Protocols
- WebSocket - Bidirectional Web
- SPDY - A better Web
- HTTP 2.0 - the Future
- Q&A
Welcome to W3C Discussion Services

The W3C list archives include the following:

- Public lists.
- Member lists.
- Team archives.

admin@w3.org

The World Wide Web Consortium (W3C) develops interoperable technologies for Web content and applications via activity groups each working in a particular area. W3C also promotes use of technologies that promotes最佳的Web experience. W3C develops W3C Recommendations (Web standards) and maintains a current list of W3Cacking to the stated purpose and bylaws, W3C operates using a process open to all, where anyone can participate. W3C success depends on contributions from individuals and organizations, which in turn receive recognition for their contributions.

1 HTML (600 B)

1 HTML (31 KiB)
5 Images (13 KiB)
Total: ~50 KiB

Average Site has 80 resources!

1 HTML (31 KiB), 31 Images (97 KiB)
4 CSS (40 KiB), 2 JS (90 KiB)
Total: 40 resources, ~300 KiB

Simone Bordet
sbordet@italio.com
Growth of w3c.org

Growth of www.w3c.org

Year

Content size (KB)

Files per page

1996 2002 2012

Javascript
CSS
Images

HTML
Files

Simone Bordet
sbordet@intalio.com
Growth network capacity
## Calculated Page Load Time

### With HTTP/1.0

<table>
<thead>
<tr>
<th>Feature</th>
<th>1996</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>YEAR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td>HTTP/1.0</td>
<td>HTTP/1.0</td>
</tr>
<tr>
<td>max Connections</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Connections</td>
<td>1</td>
<td>38</td>
</tr>
<tr>
<td>Data per connection(KB)</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Request per connection</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>establish(ms)</td>
<td>120</td>
<td>950</td>
</tr>
<tr>
<td>requests(ms)</td>
<td>120</td>
<td>950</td>
</tr>
<tr>
<td>data(ms)</td>
<td>556</td>
<td>495</td>
</tr>
<tr>
<td>Slow start (ms)</td>
<td>278</td>
<td>247</td>
</tr>
<tr>
<td>Load Time (ms)</td>
<td>1073</td>
<td>2642</td>
</tr>
</tbody>
</table>

Content Growth > Network Growth
## Improved HTTP

### Calculated Load Time

With features

<table>
<thead>
<tr>
<th>YEAR</th>
<th>1996</th>
<th>2012</th>
<th>2012</th>
<th>2012</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol</td>
<td>HTTP/1.0</td>
<td>HTTP/1.1</td>
<td>HTTP/1.1</td>
<td>HTTP/1.1</td>
</tr>
<tr>
<td>Feature</td>
<td></td>
<td></td>
<td>Pipeline</td>
<td></td>
</tr>
<tr>
<td>max Connections</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Connections</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Data per connection</td>
<td>2</td>
<td>127</td>
<td>127</td>
<td>42</td>
</tr>
<tr>
<td>Request per connection</td>
<td>1</td>
<td>19</td>
<td>19</td>
<td>6</td>
</tr>
<tr>
<td>establish(ms)</td>
<td>120</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>requests(ms)</td>
<td>120</td>
<td>950</td>
<td>50</td>
<td>317</td>
</tr>
<tr>
<td>data(ms)</td>
<td>556</td>
<td>495</td>
<td>495</td>
<td>495</td>
</tr>
<tr>
<td>Slow start (ms)</td>
<td>278</td>
<td>16</td>
<td>16</td>
<td>47</td>
</tr>
<tr>
<td>Load Time (ms)</td>
<td>1073</td>
<td>1510</td>
<td>610</td>
<td>908</td>
</tr>
</tbody>
</table>

38/2 = 19 Round Trips

Semantic issues

Turned off by default

Connections have a cost

Simone Bordet

sbordet@intalio.com
Multiple Connection Issues

- **Server resources**
  - Buffers and kernel data
  - Threads or selectors

- **Load balancers must be sticky**
  - Avoid distributed session

- **Difficult concurrency issues**
  - Multiple simultaneous accesses
  - Maybe out of order via different paths

- **Starts an arms race**
  - If 6 are good, surely 12 are better?
  - Already happening with domain sharding!
New Protocols Needed

Content Growth continues > Network Growth
- HTTP has taken the easy wins
- HTTP does not support the rich semantics needed
- We've started to “bend the rules”

The natives are getting restless
- New & Experimental protocols have been deployed!
  - Websocket
    - New Semantics
  - SPDY
    - Better Efficiency
WebSocket
Effort Initiated by Browser Vendors to

- Bidirectional low latency communications
- Replace Comet HTTP “hacks”
- Implement rich web applications within the browser
  - *JavaFX? Silverlight? Flash?*

**Standardized**

- IETF - RFC6455 wire protocol
- W3C - HTML5 Javascript API
- JCP - JSR356 Java API (in progress)
interface WebSocket : EventTarget {
    readonly attribute DOMString url;
    attribute EventHandler onopen;
    attribute EventHandler onerror;
    attribute EventHandler onclose;
    attribute EventHandler onmessage;

    void send(DOMString data);
    void send(Blob data);

    void close(...);
}
WebSocket Upgrade

- Runs on port 80 (or 443 for wss)
- Uses HTTP/1.1 Upgrade mechanism

**REQUEST**

```
GET / HTTP/1.1
Host: localhost:8080
Origin: http://localhost:8080
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Key: SbdIETLKHQ1TNBLeZFZS0g==
Sec-WebSocket-Version: 13
```

**RESPONSE**

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: websocket
Sec-WebSocket-Accept: y4yXRUolfnFfo3Jc5HFqRHNgx2A=
```
WebSocket Framing

WebSocket frames of two types
- Control frames (Close, Ping, Pong)
- Data frames (UTF8 Text & Binary)

Protocol is very compact
- Smallest data frame has only 2 bytes overhead
HTTP Performance

- Jetty-7.6, CometD-2.4, Chat simulation
- HTTP
WebSocket Performance

- Jetty-7.6, CometD-2.4, Chat Simulation
- WebSocket
WebSocket Deployment

**Browsers**
- Firefox 11 (android 15)
- Internet Explorer 10 PP5
- Chrome 16 (android 18)
- Safari 6 (iOS 6)
- Opera 12.1 (mini N/A)
- Blackberry 7.0
- Jetty Client 7, 8, 9

**Java Servers**
- Jetty 7, Jetty 8, Jetty 9
- Glassfish
- Resin 4
- Tomcat 7
  - *not final yet*

59.03% by StatCounter GlobalStats
WebSocket Conclusions

- WebSocket is Here NOW!
  - In production
  - Big benefits
  - Use Jetty!

- WebSocket Limitations
  - Does not provide HTTP semantic
    - New applications need to be written
  - Intermediaries may need to be WebSocket aware
    - There are some problems
  - No connection limit
    - Multiplexing coming
  - Very Low level
    - Use CometD!
SPDY
- **SPDY is a live experiment to improve HTTP**
  - Addresses HTTP 1.1 limits
  - Designed to be faster and better than HTTP

- **Already widely deployed!**
  - Google, Twitter, Webtide, etc.
  - Chrome & Firefox

- **The SPDY protocol replaces HTTP on the wire**
  - But it's transparent for applications
SPDY Protocol Initiation

- **SPDY uses TLS (aka SSL) connection on port 443**
  - TLS extended with Next Protocol Negotiation

- **New framing protocol over TLS**
  - Intermediaries don't know it is not HTTP wrapped in TLS

- **Transports existing HTTP Semantics (GET, POST, HEAD etc)**
  - Client and Server applications don't know it is not HTTP
SPDY defines a new framing layer

- Binary, slightly more expensive than WebSocket's
- Faster than HTTP to parse
- Built to carry HTTP, but can carry other protocols
SPDY Multiplexing

- Multiplexing is built-in
Multiplexing

- Allows to make better use of TCP connections
  - Reduces TCP slow start
- Uses less resources on the server
- Responses may be sent out of order
  - Without waiting for slow responses

Key point: reducing round trip waits

- Caused by limited connections and lack of multiplexing
SPDY is built for HTTP
- But allows WebSocket to be carried too

HTTP header compression
- Typical HTTP requests contains ~1 KiB of headers
  - Most of them are constant
- Saves ~25% on first request, 80% and more on subsequent requests

Headers for typical requests to webtide.com:
- 568 bytes → 389 bytes → 26 bytes
- 100% → 68% → 4.5%
SPDY Push

- Server pushes secondary resources that are associated to a primary resource
- Works in collaboration with browser's cache for a better user experience

Jetty is one of the first SPDY servers that provides automated push functionalities

- Totally transparent for applications
- Based on the “Referrer” header
  - To associate primary and secondary resources
- Using “If-Modified-Since” header to avoid unnecessary pushes
How much faster than HTTP can SPDY be?
- Early to say, but initial figures up to 25%
- SPDY optimizations limited by HTTP “hacks”
  - Domain sharding

Can my JEE web application leverage SPDY?
- Yes! And without changes!
Jetty 7 & Jetty 8
- SPDY added by mocking HTTP wire protocol
- Multiplexing requires mock protocol per channel

Jetty 9
- Re architected to separate wire protocol from semantics
- HTTP semantics shared between HTTP, SPDY
- Supports 1:n Protocol:Semantics
- Same separation of Websocket protocol from semantics
- HTTP, SPDY, WebSocket, TLS 1st class citizens

PLEASE USE SPDY NOW!
- Your use case can make Jetty and the SPDY better
SPDY Clients

- **Desktop**
  - Firefox 13
  - Chrome 4
  - Opera 12.1
  - Jetty Client 7,8, 9

- **Mobile**
  - Android Browser 4.1
  - Chrome for Android 18.0
  - Firefox Android 15.0

45.68% by StatCounter GlobalStats
HTTP 2.0 work has started
- IETF HTTPbis working group rechartered

Several Proposals received:
- Google - SPDY as the basis
- Microsoft – websocket framing with HTTP semantics

SPDY Adopted as the starting point
- Unlikely to require TLS or NPN
- Will define precise interactions with proxies
- PLEASE let us end up with one framing layer
Conclusions

- The Web is evolving
  - Web Protocols must keep the pace with new loads
  - Applications must keep pace with new semantics

- Try WebSocket today
  - Production ready

- Try SPDY today
  - Check your application is ready for the future
  - Check the future is ready for your application
**References**

- **WebSocket Protocol**

- **JSR 340 (Servlet 3.1)**

- **SPDY**
  - [http://www.chromium.org/spdy/](http://www.chromium.org/spdy/)

- **Jetty**
  - [http://eclipse.org/jetty](http://eclipse.org/jetty)
Questions & Answers