Nuts and Bolts of WebSocket

Arun Gupta, @arungupta
Red Hat

Agenda

- Introduction
- WebSocket using JSR 356
- Securing WebSocket
- Load Balance WebSocket
- REST and SSE
- Debugging

“Limitations” of HTTP

- Client-driven
- Half-duplex
- Verbose
- New TCP connection

“Hello World” HTTP Request/Response

```
POST /websocket-vs-rest-payload/webresources/rest HTTP/1.1\r\n
Host: localhost:8080\r\n
Connection: keep-alive\r\n
Content-Length: 11\r\n
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/32.0.1700.107 Safari/537.36\r\n
Origin: chrome-extension://hgmloofddffdnphfgcellkdfbfbjeloo\r\n
Content-Type: text/plain \r\n
Accept: */*\r\n
Accept-Encoding: gzip,deflate,sdch\r\n
Accept-Language: en-US,en;q=0.8\r\n
\r\n
```

```
HTTP/1.1 200 OK\r\n
Connection: keep-alive\r\n
X-Powered-By: Undertow 1\r\n
Server: Wildfly 8 \r\n
Content-Type: text/plain\r\n
Content-Length: 11 \r\n
Date: Fri, 21 Feb 2014 21:27:53 GMT \r\n
\r\n
```

663 bytes

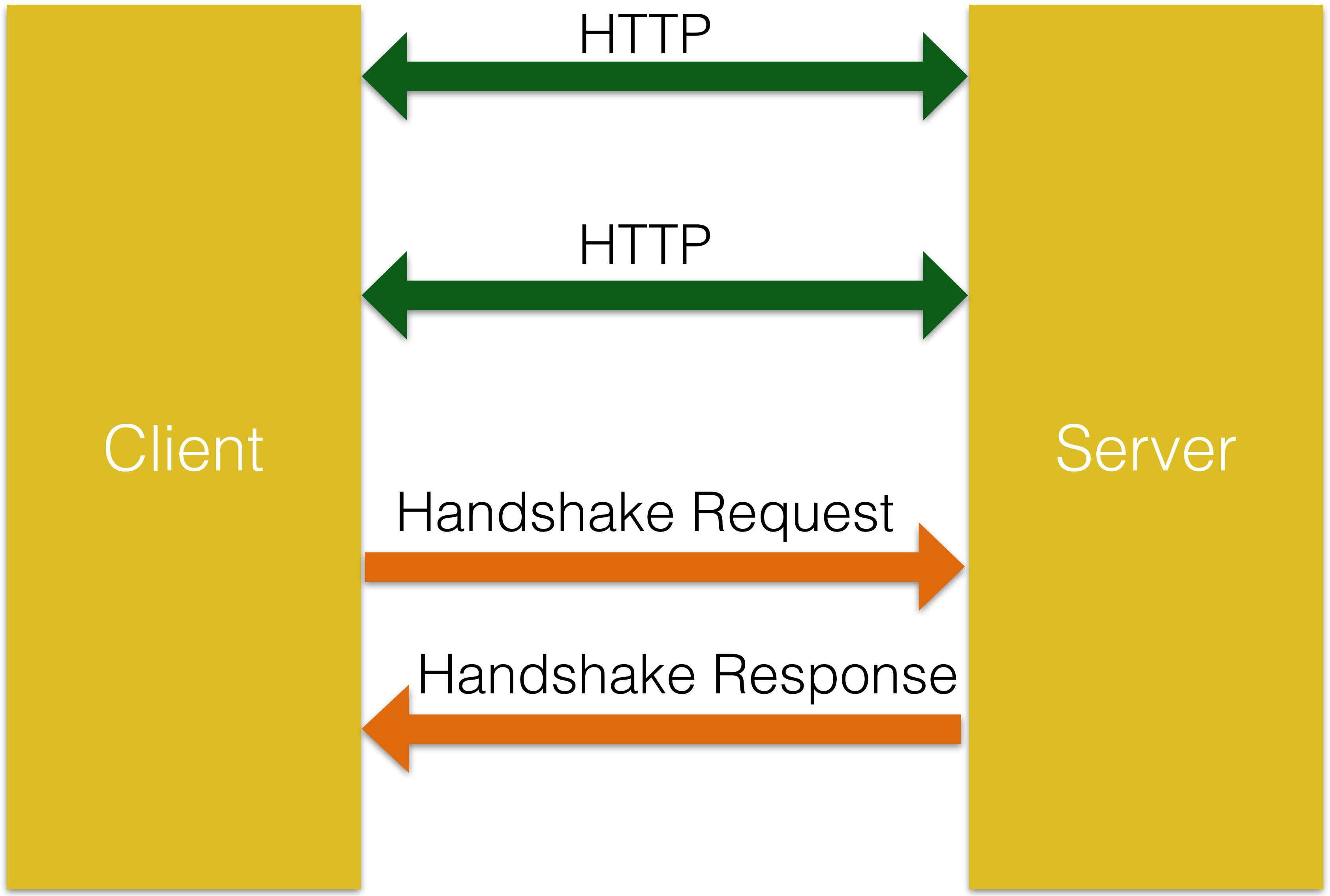
How WebSocket solves it ?

- Bi-directional (client-driven)
- Full-duplex (half-duplex)
- Lean protocol (verbose)
- Single TCP connection (new TCP)

What is WebSocket ?

- Bi-directional, full-duplex, communication channel over a single TCP connection
- Originally proposed as part of HTML5
- IETF-defined **Protocol**: RFC 6455
- W3C-defined **JavaScript API**

How does it work ?



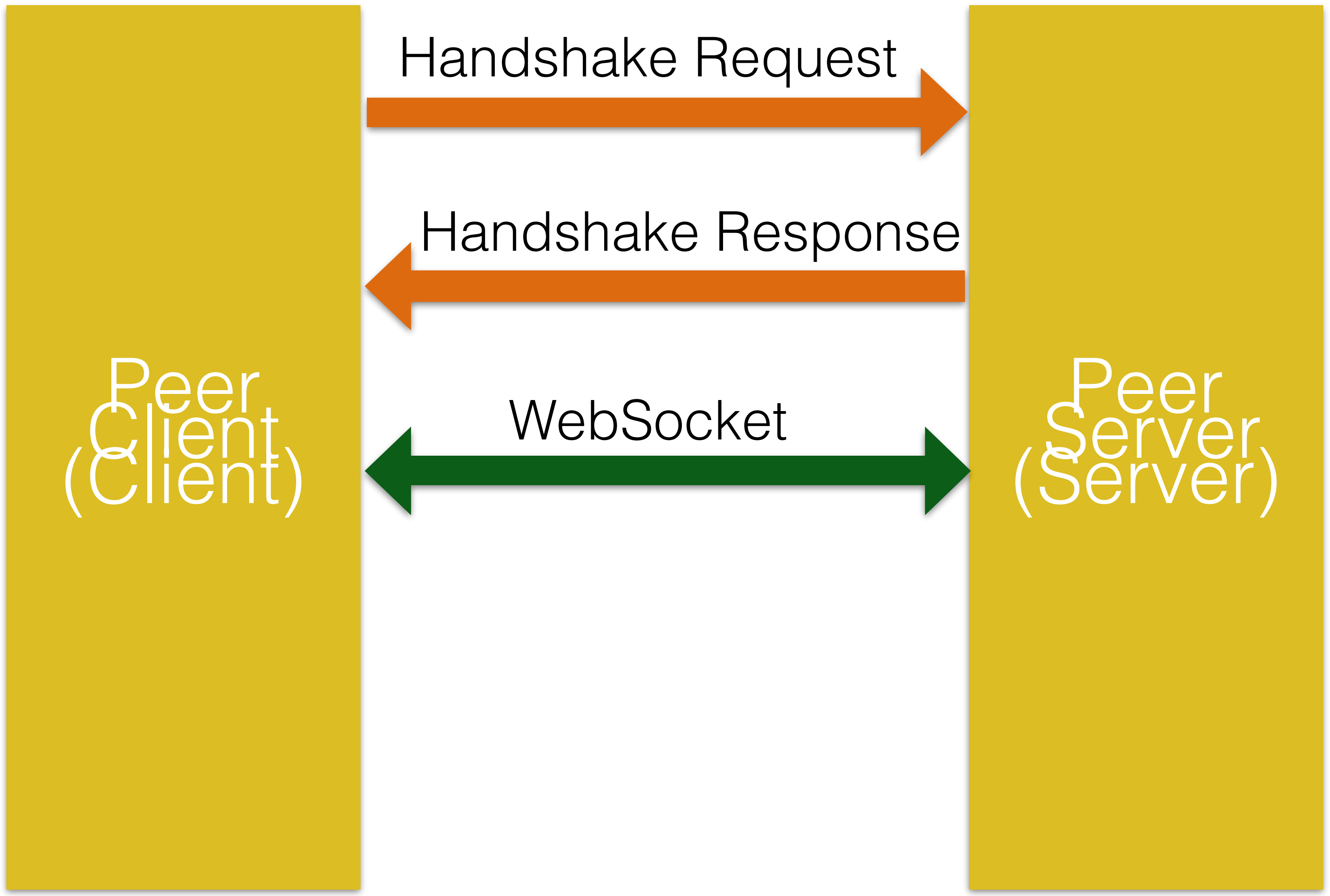
Handshake Request

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

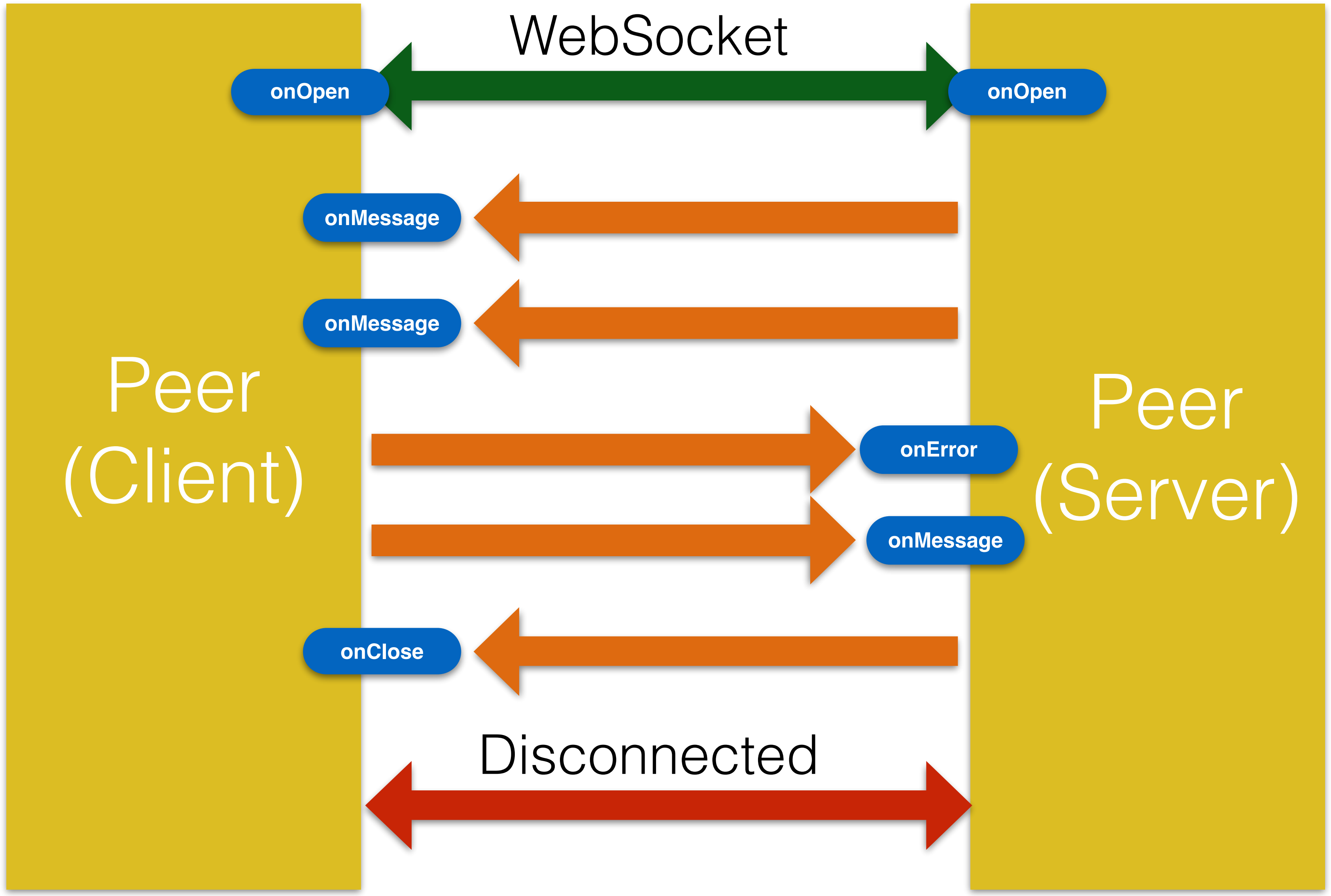
Handshake Response

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

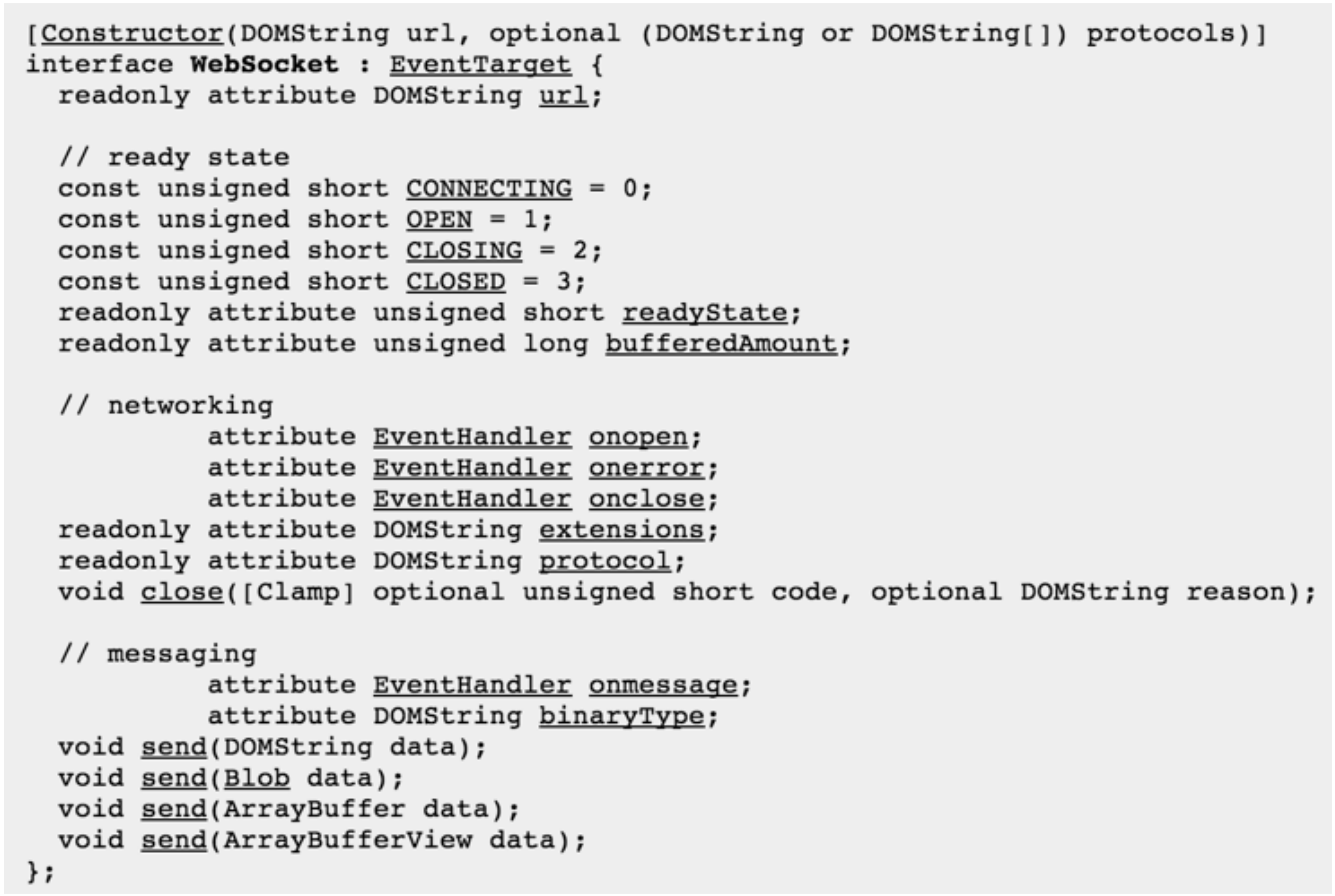
How does it work ?



How does it work ?



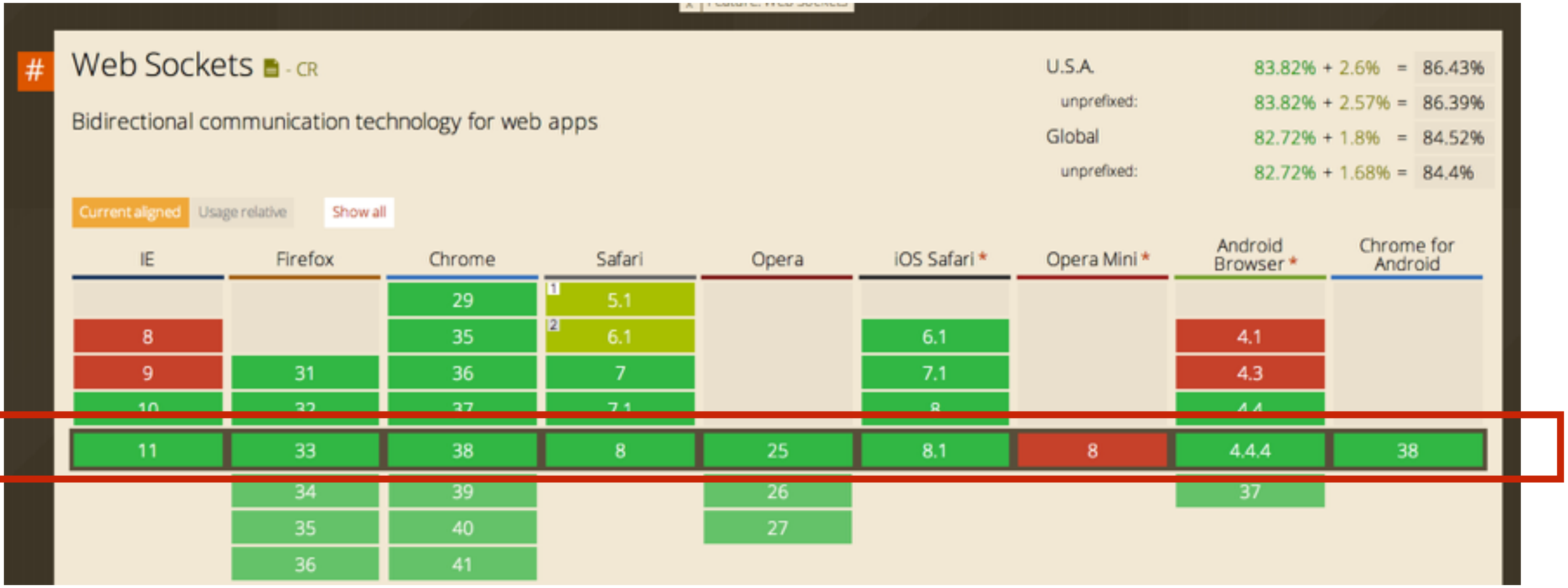
WebSocket JavaScript API



```
[Constructor(DOMString url, optional (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
  readonly attribute DOMString url;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;

  // networking
           attribute EventHandler onopen;
           attribute EventHandler onerror;
           attribute EventHandler onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] optional unsigned short code, optional DOMString reason);

  // messaging
           attribute EventHandler onmessage;
           attribute DOMString binaryType;
  void send(DOMString data);
  void send(Blob data);
  void send(ArrayBuffer data);
  void send(ArrayBufferView data);
};
```

www.w3.org/TR/websockets

Support in Browsers



Web Sockets - CR

Bidirectional communication technology for web apps

U.S.A.	83.82%	+ 2.6%	=	86.43%
unprefixed:	83.82%	+ 2.57%	=	86.39%
Global	82.72%	+ 1.8%	=	84.52%
unprefixed:	82.72%	+ 1.68%	=	84.4%

Current aligned · Usage relative · Show all

IE	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
		29	[1] 5.1					
8		35	[2] 6.1		6.1		4.1	
9	31	36	7		7.1		4.3	
10	32	37	7.1		8		4.4	
11	33	38	8	25	8.1	8	4.4.4	38
	34	39		26			37	
	35	40		27				
	36	41						

caniuse.com/websockets

Java API for WebSocket

- API for WebSocket server and client endpoint
  - Annotated: **`@ServerEndpoint, @ClientEndpoint`**
  - Programmatic: **`Endpoint`**
    - WebSocket opening handshake negotiation
- Lifecycle Callback methods
- Integration with Java EE technologies

Annotated Endpoint

```
import javax.websocket.*;

@ServerEndpoint("/hello")
public class HelloBean {
  @OnMessage
  public String sayHello(String name) {
    return "Hello " + name;
}
```

WebSocket Annotations

- Class-level annotations
  - **`@ServerEndpoint`**: Turns a POJO in a server endpoint
  - **`@ClientEndpoint`**: Turns a POJO in a client endpoint

WebSocket Annotations

- Method-level annotations
  - **`@OnMessage`**: Intercepts WebSocket messages
  - **`@OnOpen`**: Intercepts WebSocket open events
  - **`@OnClose`**: Intercepts WebSocket close events
  - **`@OnError`**: Intercepts WebSocket error events

WebSocket Annotations

- Parameter-level annotations
    - `@PathParam`: Matches path segment of a URI-template

@ServerEndpoint attributes

- `value`: Relative URI or URI template e.g. ‘/hello’ or ‘/chat/{subscriber-level}’
- `decoders`: list of message decoder classnames
- `encoders`: list of message encoder classnames
- `subprotocols`: list of the names of the supported subprotocols

Chat Server

```
@ServerEndpoint("/chat")
public class ChatBean {
  static Set<Session> peers = Collections.synchronizedSet("...");

  @OnOpen
  public void onOpen(Session peer) {
    peers.add(peer);
  }

  @OnClose
  public void onClose(Session peer) {
    peers.remove(peer);
  }

  @OnMessage
  public void message(String message) {
    for (Session peer : peers) {
      peer.getBasicRemote().sendObject(message);
    }
  }
}
```

Chat Server Simplified

```
@ServerEndpoint("/chat")
public class ChatBean {
  @OnMessage
  public void message(String message, Session endpoint) {
    for (Session peer : endpoint.getOpenSessions()) {
      peer.getBasicRemote().sendObject(message);
    }
  }
}
```

http://blog.arungupta.me/2014/10/websocket-chat-wildfly-openshift-techtip51/

http://mywildfly-milestogo.rhcloud.com/chat/

Custom Payloads

```
@ServerEndpoint(
  value="/hello",
  decoders={MyMessageDecoder.class},
  encoders={MyMessageEncoder.class}
)
public class MyEndpoint {
  . . .
}
```

Custom Payloads: Text decoder

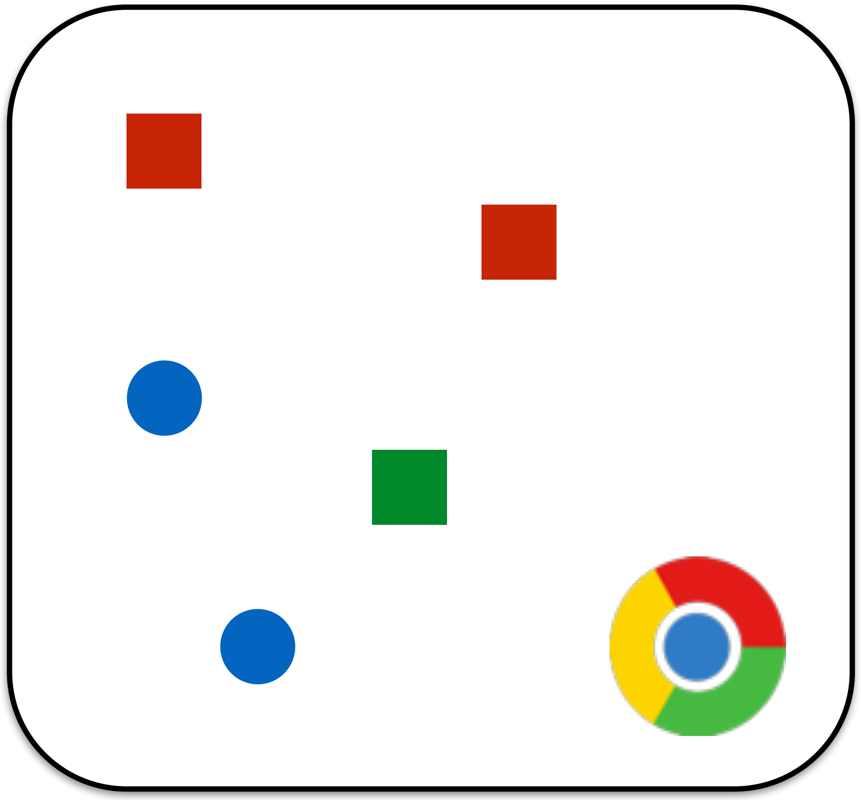
```
public class MyMessageDecoder implements Decoder.Text<MyMessage> {
  public MyMessage decode(String s) {
    JsonObject jsonObject = Json.createReader("...").readObject();
    return new MyMessage(jsonObject);
  }

  public boolean willDecode(String string) {
    . . .
    return true;
  }

  . . .
}
```

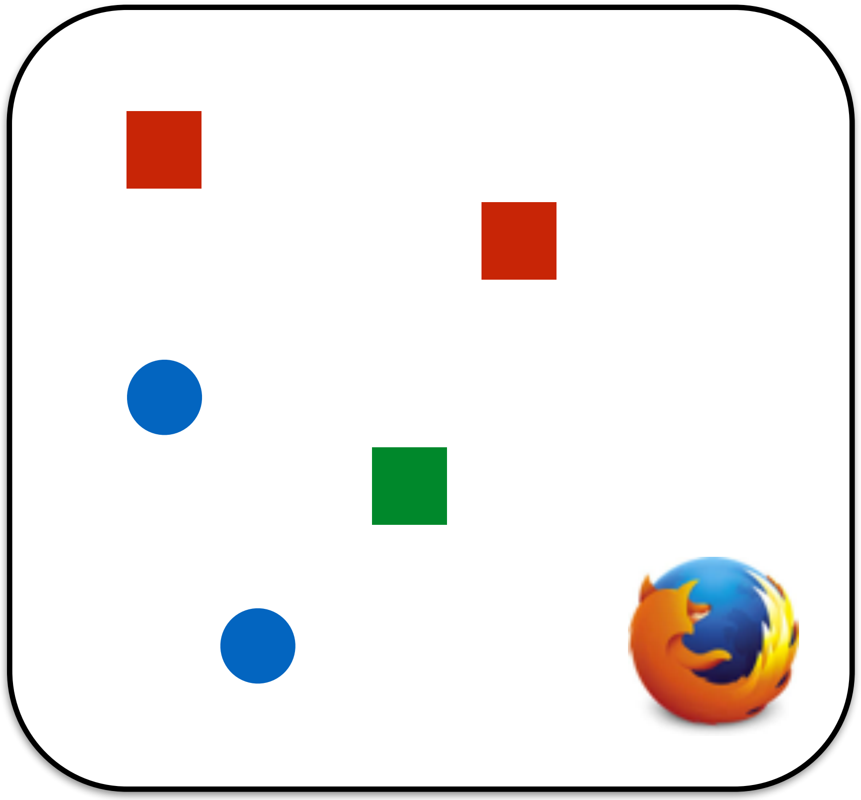
Custom Payloads: Text encoder

```
public class MyMessageDecoder implements Encoder.Text<MyMessage> {
  public String encode(MyMessage myMessage) {
    return myMessage.jsonObject.toString();
  }

  . . .
}
```

Custom Payloads: Binary decoder

```
public class MyMessageDecoder implements Decoder.Binary<MyMessage>
  public MyMessage decode(byte[] s) {
    . . .
    return myMessage;
  }

  public boolean willDecode(byte[] string) {
    . . .
    return true;
  }

  . . .
}
```





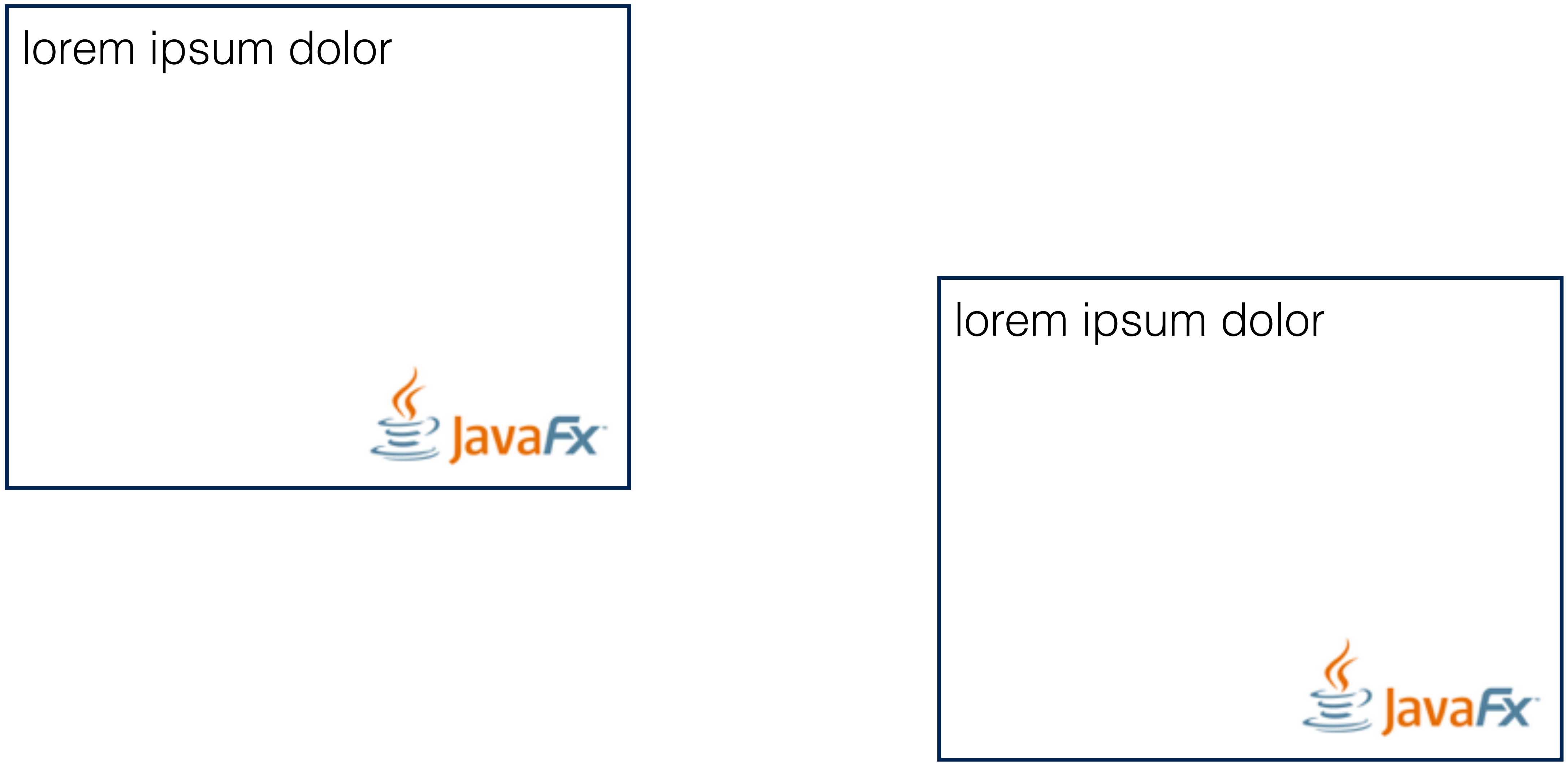
https://github.com/javaee-samples/javaee7-samples/tree/master/websocket/whiteboard

http://mywildfly-milestogo.rhcloud.com/whiteboard/

Client Endpoint

```
@ClientEndpoint
public class HelloClient {
  @OnMessage public void message(
    String message,
    Session session) {
    //. . .
  }
}
```

```
WebSocketContainer c = ContainerProvider.getWebSocketContainer();
c.connectToServer(HelloClient.class, "hello");
```



https://github.com/javaee-samples/javaee7-samples/tree/master/websocket/google-docs

Securing WebSockets

- Origin-based security model
- Sec-xxx keys can not be set using XMLHttpRequest
    - Sec-WebSocket-Key, Sec-WebSocket-Version
- User-based security using Servlet security mechanism
    - Endpoint mapped by **ws://** is protected using security model defined using the corresponding http:// URI
    - Authorization defined using `<security-constraint>`
- Transport Confidentiality using **wss://**
    - Access allowed over encrypted connection only

User-based Security

http://blog.arungupta.me/2014/10/securing-websockets-username-password-servlet-security-techtip49/

TLS-based Security

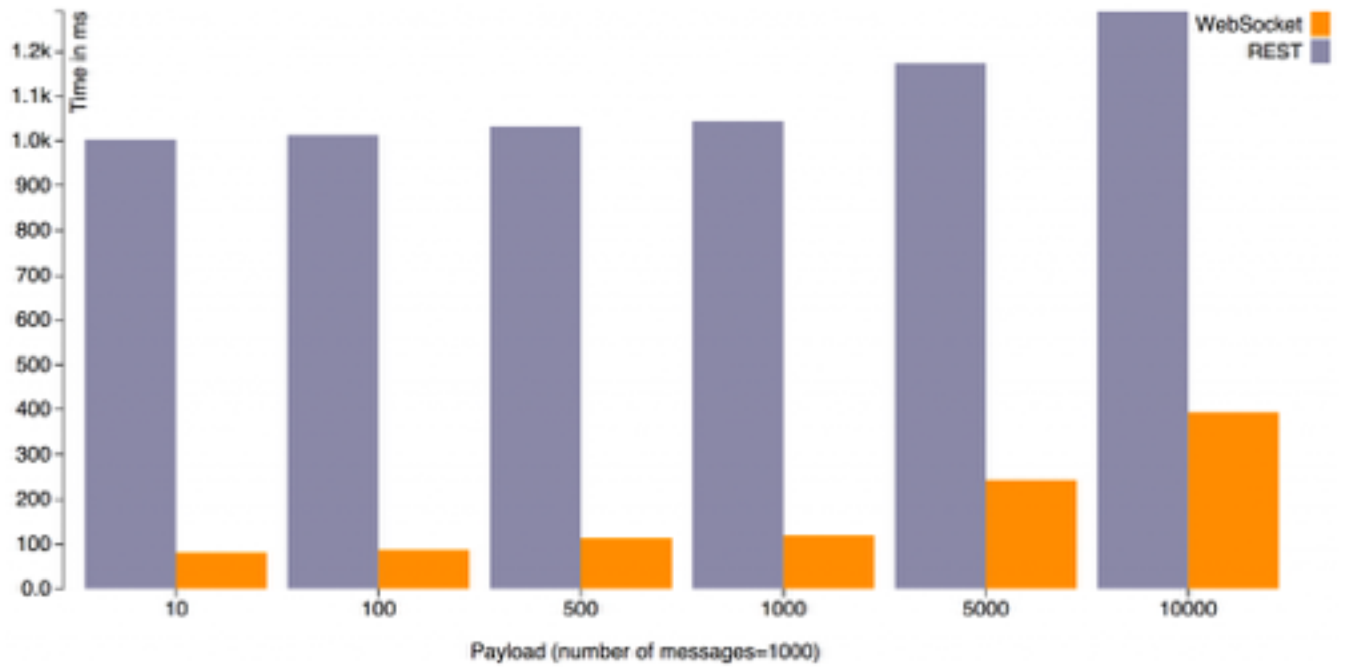
http://blog.arungupta.me/2014/10/securing-websocket-wss-https-tls-techtip50/

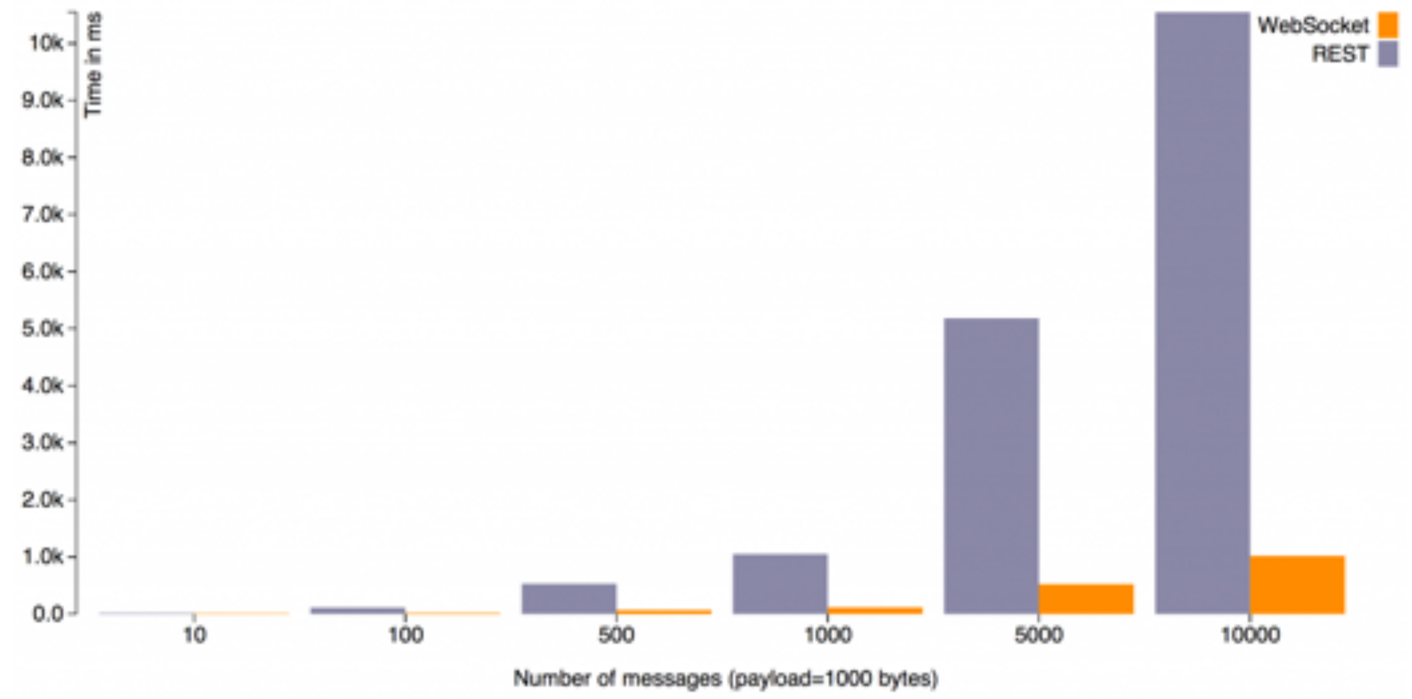
Load Balance WebSocket

- Reverse proxy
- Only vertical scaling
- No session replication

http://blog.arungupta.me/2014/08/load-balance-websockets-apache-httpd-techtip48/

Compare with REST





http://blog.arungupta.me/2014/02/rest-vs-websocket-comparison-benchmarks/

Server-Sent Events

- Part of HTML5 Specification
- Server-push notifications
- Cross-browser JavaScript API: **`EventSource`**
- Message callbacks
- MIME type: **`text/eventstream`**

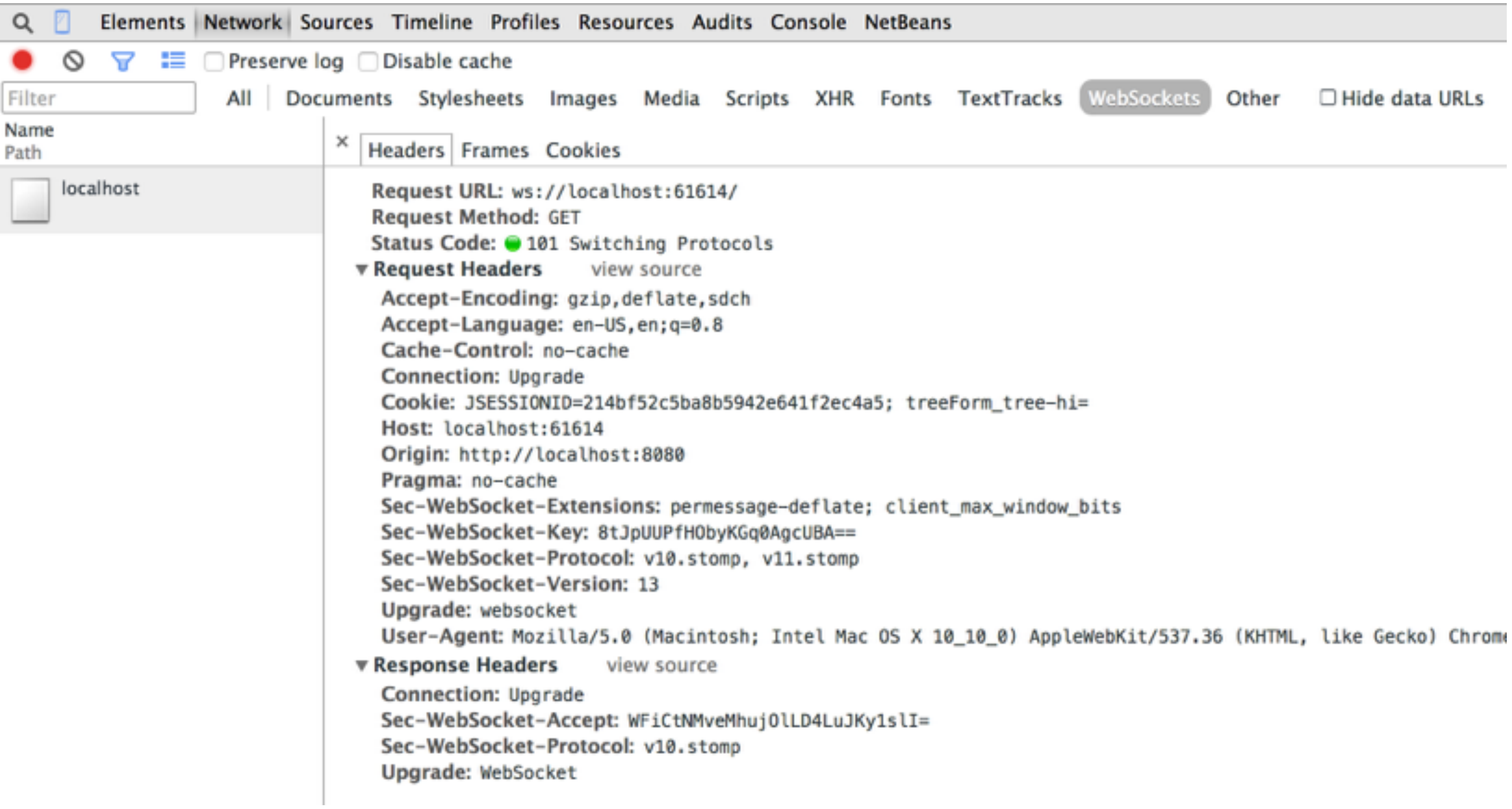
WebSockets and SSE ?

WebSocket	Server-Sent Event
Over a custom protocol	Over simple HTTP
Full-duplex, bi-directional	Server-push only, client-server OOB
Native support in most browsers	Can be poly-filled to backport
Not straight forward protocol	Simpler protocol

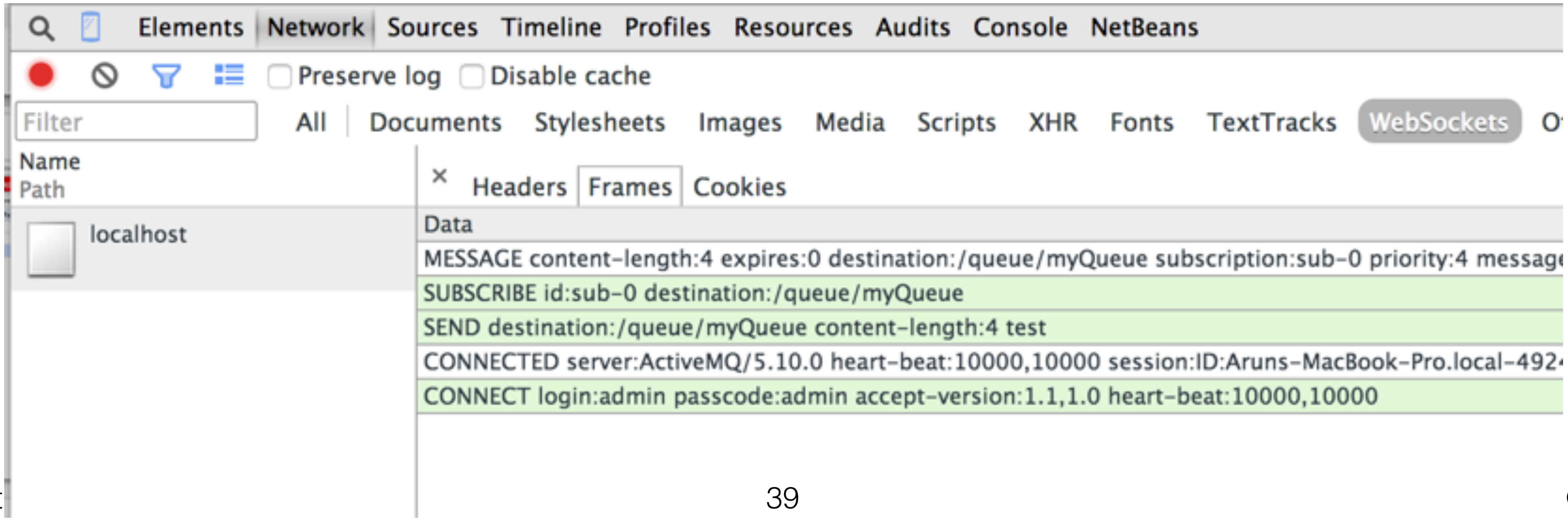
WebSockets and SSE ?

WebSocket	Server-Sent Event
Application-specific reconnection	Built-in support for reconnection and event id
Require server and/or proxy configurations	No server or proxy change required
Text and Binary	Text only
Pre-defined message handlers	Pre-defined and arbitrary

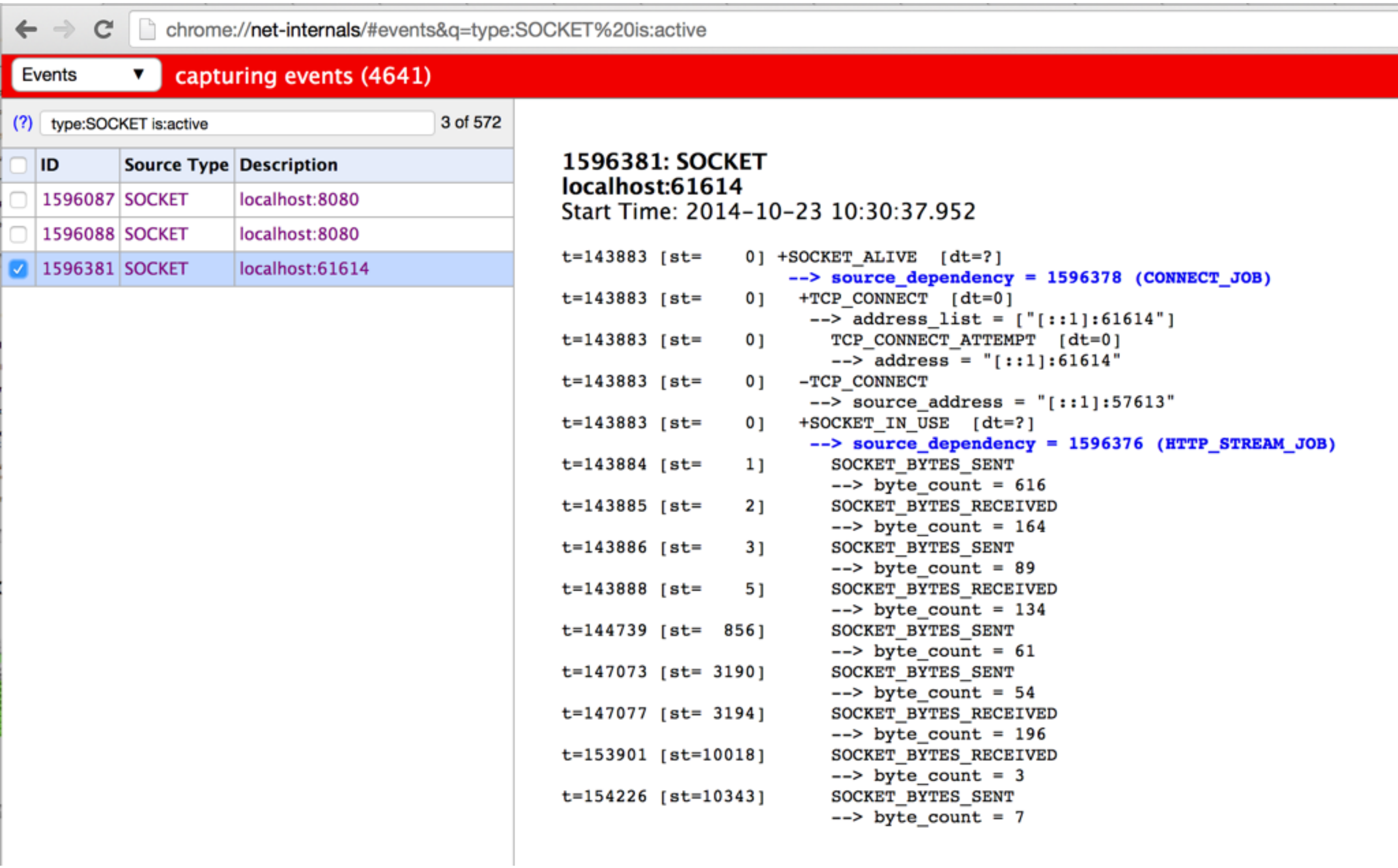
Debugging WebSockets


Elements Network Sources Timeline Profiles Resources Audits Console NetBeans
Preserve log
Disable cache
Filter
All Documents Stylesheets Images Media Scripts XHR Fonts TextTracks WebSockets Other
Hide data URLs
Name
Path
localhost
Headers Frames Cookies

```
Request URL: ws://localhost:61614/
Request Method: GET
Status Code: ● 101 Switching Protocols
▼ Request Headers      view source
   Accept-Encoding: gzip,deflate,sdch
   Accept-Language: en-US,en;q=0.8
   Cache-Control: no-cache
   Connection: Upgrade
   Cookie: JSESSIONID=214bf52c5ba8b5942e641f2ec4a5; treeForm_tree-hi=
   Host: localhost:61614
   Origin: http://localhost:8080
   Pragma: no-cache
   Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
   Sec-WebSocket-Key: 8tJpUUPfHObyKGq0AgcUBA==
   Sec-WebSocket-Protocol: v10.stomp, v11.stomp
   Sec-WebSocket-Version: 13
   Upgrade: websocket
   User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_0) AppleWebKit/537.36 (KHTML, like Gecko) Chrom
▼ Response Headers      view source
   Connection: Upgrade
   Sec-WebSocket-Accept: WFiCtNMveMhujOlLD4LuJKy1slI=
   Sec-WebSocket-Protocol: v10.stomp
   Upgrade: WebSocket
```


Elements Network Sources Timeline Profiles Resources Audits Console NetBeans
Preserve log
Disable cache
Filter
All Documents Stylesheets Images Media Scripts XHR Fonts TextTracks WebSockets O
Name
Path
localhost
Headers Frames Cookies

```
Data
MESSAGE content-length:4 expires:0 destination:/queue/myQueue subscription:sub-0 priority:4 message
SUBSCRIBE id:sub-0 destination:/queue/myQueue
SEND destination:/queue/myQueue content-length:4 test
CONNECTED server:ActiveMQ/5.10.0 heart-beat:10000,10000 session:ID:Aruns-MacBook-Pro.local-492
CONNECT login:admin passcode:admin accept-version:1.1,1.0 heart-beat:10000,10000
```



chrome://net-internals/#events&q=type:SOCKET%20is:active

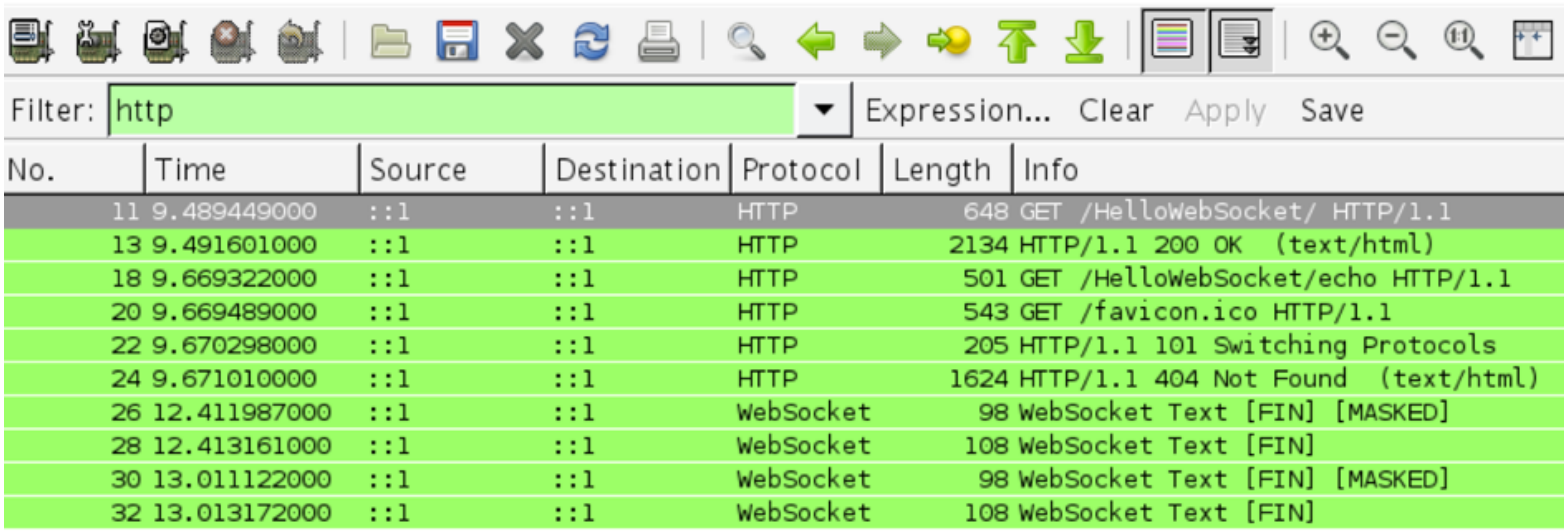
Events ▼ capturing events (4641)

(?) type:SOCKET is:active 3 of 572

	ID	Source Type	Description
☐	1596087	SOCKET	localhost:8080
☐	1596088	SOCKET	localhost:8080
✓	1596381	SOCKET	localhost:61614

1596381: SOCKET
localhost:61614
Start Time: 2014-10-23 10:30:37.952

```
t=143883 [st=    0] +SOCKET_ALIVE  [dt=?]
                     --> source_dependency = 1596378 (CONNECT_JOB)
t=143883 [st=    0]   +TCP_CONNECT  [dt=0]
                       --> address_list = ["[::1]:61614"]
t=143883 [st=    0]      TCP_CONNECT_ATTEMPT  [dt=0]
                         --> address = "[::1]:61614"
t=143883 [st=    0]   -TCP_CONNECT
                       --> source_address = "[::1]:57613"
t=143883 [st=    0]   +SOCKET_IN_USE  [dt=?]
                       --> source_dependency = 1596376 (HTTP_STREAM_JOB)
t=143884 [st=    1]      SOCKET_BYTES_SENT
                         --> byte_count = 616
t=143885 [st=    2]      SOCKET_BYTES_RECEIVED
                         --> byte_count = 164
t=143886 [st=    3]      SOCKET_BYTES_SENT
                         --> byte_count = 89
t=143888 [st=    5]      SOCKET_BYTES_RECEIVED
                         --> byte_count = 134
t=144739 [st=  856]      SOCKET_BYTES_SENT
                         --> byte_count = 61
t=147073 [st= 3190]      SOCKET_BYTES_SENT
                         --> byte_count = 54
t=147077 [st= 3194]      SOCKET_BYTES_RECEIVED
                         --> byte_count = 196
t=153901 [st=10018]      SOCKET_BYTES_RECEIVED
                         --> byte_count = 3
t=154226 [st=10343]      SOCKET_BYTES_SENT
                         --> byte_count = 7
```



Filter: http

Expression... Clear Apply Save

No.	Time	Source	Destination	Protocol	Length	Info
11	9.489449000	::1	::1	HTTP	648	GET /HelloWebSocket/ HTTP/1.1
13	9.491601000	::1	::1	HTTP	2134	HTTP/1.1 200 OK (text/html)
18	9.669322000	::1	::1	HTTP	501	GET /HelloWebSocket/echo HTTP/1.1
20	9.669489000	::1	::1	HTTP	543	GET /favicon.ico HTTP/1.1
22	9.670298000	::1	::1	HTTP	205	HTTP/1.1 101 Switching Protocols
24	9.671010000	::1	::1	HTTP	1624	HTTP/1.1 404 Not Found (text/html)
26	12.411987000	::1	::1	WebSocket	98	WebSocket Text [FIN] [MASKED]
28	12.413161000	::1	::1	WebSocket	108	WebSocket Text [FIN]
30	13.011122000	::1	::1	WebSocket	98	WebSocket Text [FIN] [MASKED]
32	13.013172000	::1	::1	WebSocket	108	WebSocket Text [FIN]

Resources

- Material: github.com/arun-gupta/nuts-and-bolts-of-websocket