Introduction to WebSockets

About

Gunnar Hillert

- Company: SpringSource, a division of VMware
- Projects:
  - Spring Integration (http://www.springintegration.org)
  - Spring AMQP
  - Cloud Foundry (Maven Plugin)
- Twitter: @ghillert
- LinkedIn: http://www.linkedin.com/in/hillert
- Blog
  - http://blog.springsource.org/author/ghillert/
  - http://blog.hillert.com

Objectives

- Survey the lay of the land
- Less focus on syntax and mechanics
- Broad, pragmatic perspective
- Special emphasis on Java

Where to find the slides + samples?

- **Slides**: https://slideshare.net/hillert/ajug2012websocket
- **Samples**: https://github.com/cbeams/bitcoin-rt

WebSocket 101

The Problem

- Some web apps need two-way communication / rapid updates
- AJAX and Comet techniques can amount to an “abuse of HTTP”

The Problem

- Too many connections
- Too much overhead
- Too great a burden on the client

The Usual Suspects

- Trading
- Chat
- Gaming
- Collaboration
- Visualization

The Goal

"The goal of this technology is to provide a mechanism for browser-based applications that need two-way communication with servers that does not rely on opening multiple HTTP connections"

- RFC 6455, The WebSocket Protocol

The Approach

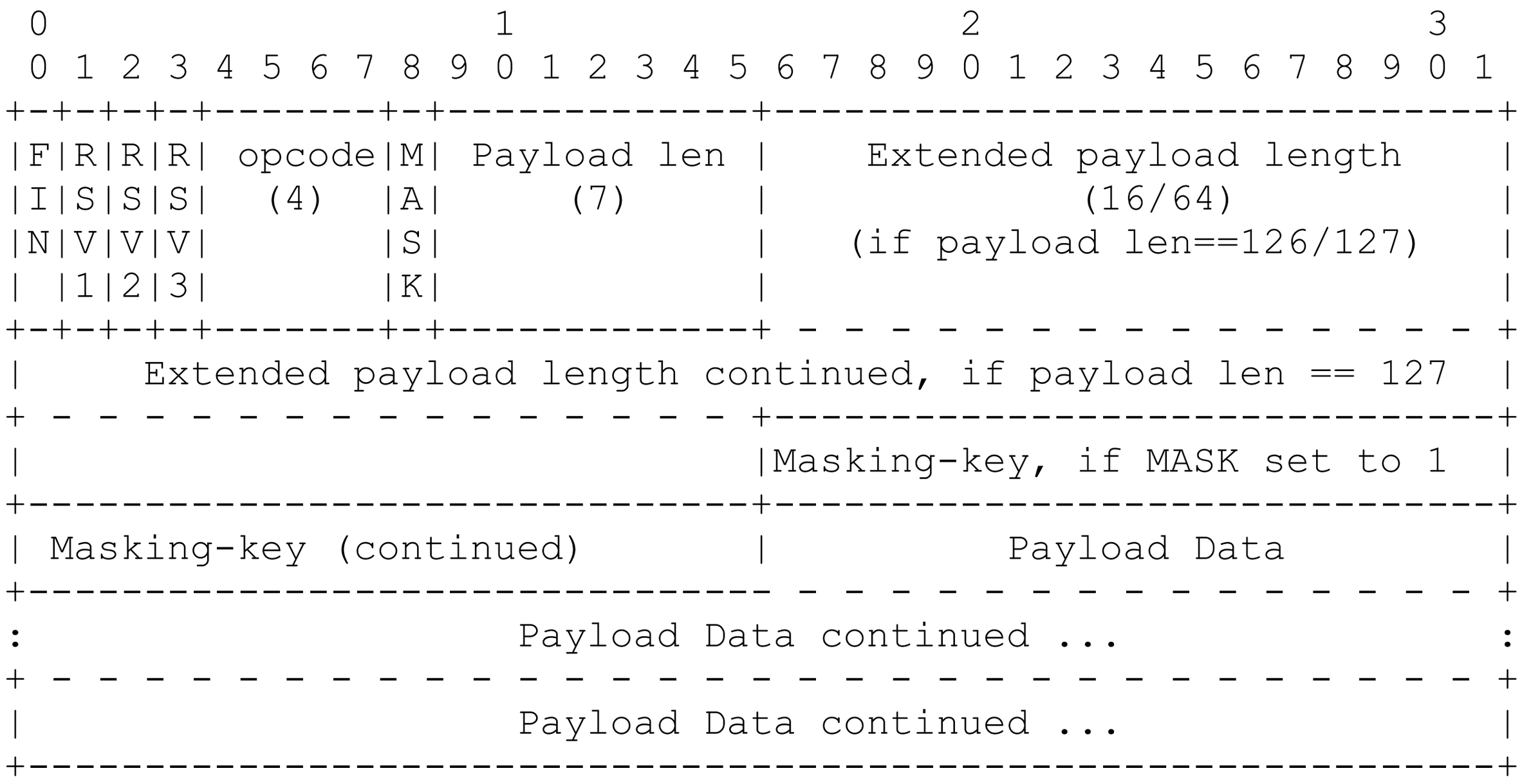
- Two-way messaging over a single connection
- Layer on TCP
- Not HTTP, but uses HTTP to bootstrap
- Extremely low-overhead

The WebSocket HTTP Handshake

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
```

```
HTTP/1.1 101 Switching
Protocols
Upgrade: websocket
Connection: Upgrade
```

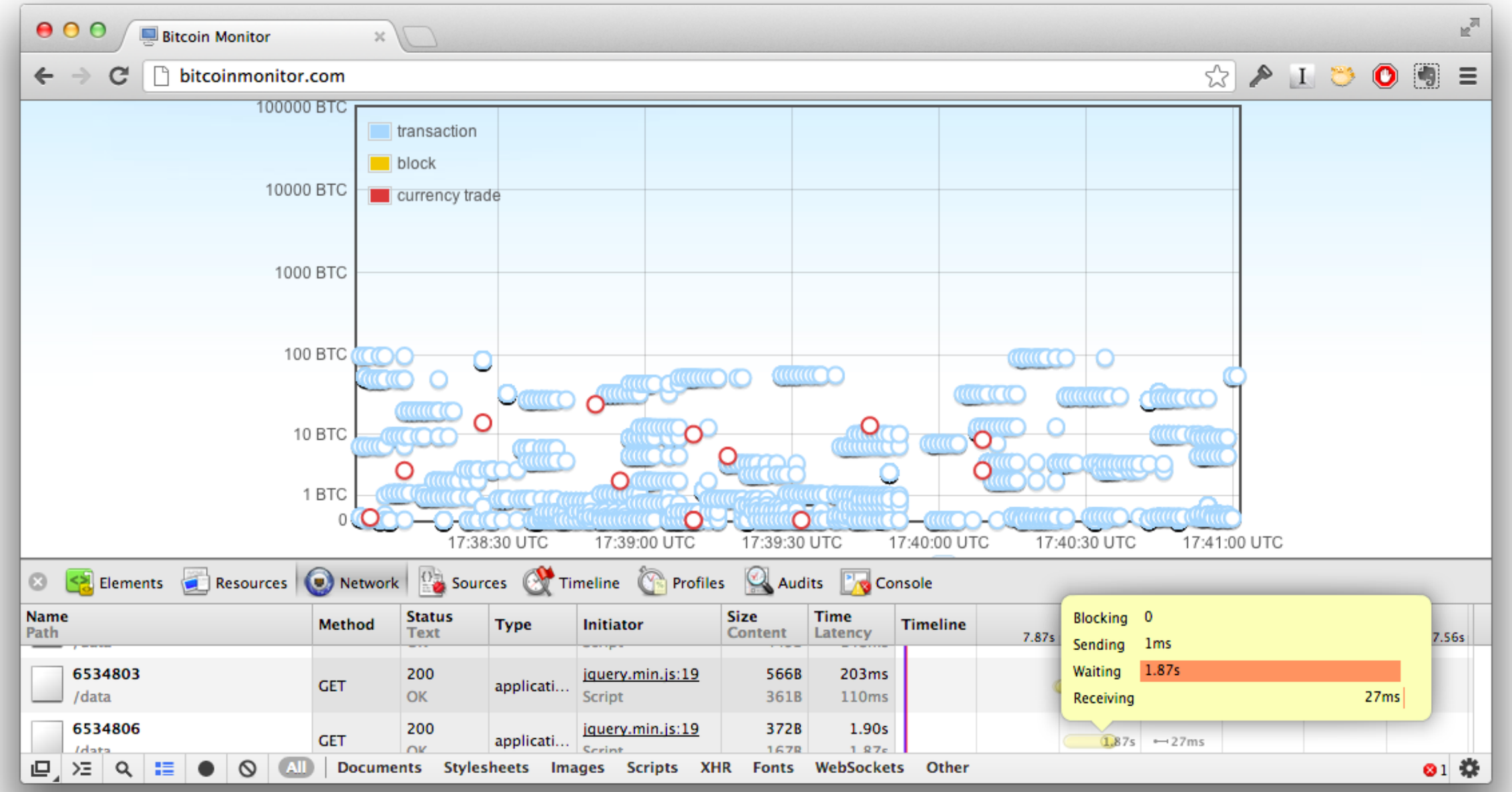
What's in a Frame?



http://www.ietf.org/rfc/rfc6455.txt

bitcoin-rt

- visualize Bitcoin transactions in real time
- inspired by original bitcoinmonitor.com


Bitcoin Monitor
bitcoinmonitor.com
100000 BTC
10000 BTC
1000 BTC
100 BTC
10 BTC
1 BTC
0
transaction
block
currency trade
17:38:30 UTC
17:39:00 UTC
17:39:30 UTC
17:40:00 UTC
17:40:30 UTC
17:41:00 UTC
Elements
Resources
Network
Sources
Timeline
Profiles
Audits
Console
Name
Path
Method
Status
Text
Type
Initiator
Size
Content
Time
Latency
Timeline
6534803
/data
GET
200
OK
applicati...
jquery.min.js:19
Script
566B
361B
203ms
110ms
6534806
GET
200
applicati...
jquery.min.js:19
372B
1.90s
Blocking 0
Sending 1ms
Waiting 1.87s
Receiving 27ms
All
Documents
Stylesheets
Images
Scripts
XHR
Fonts
WebSockets
Other

bitcoin-rt vs bitcoinmonitor

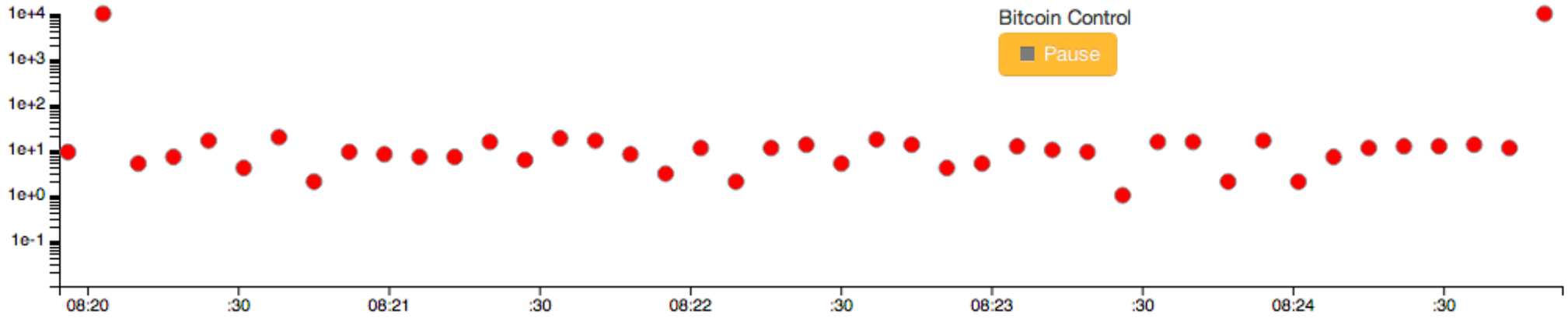
- WebSockets instead of Long Polling
- d3.js (http://d3js.org/) instead of JQuery UI
- MongoDB for persistence

bitcoin-rt – Node.js

localhost:1337

bitcoin-rt Home About

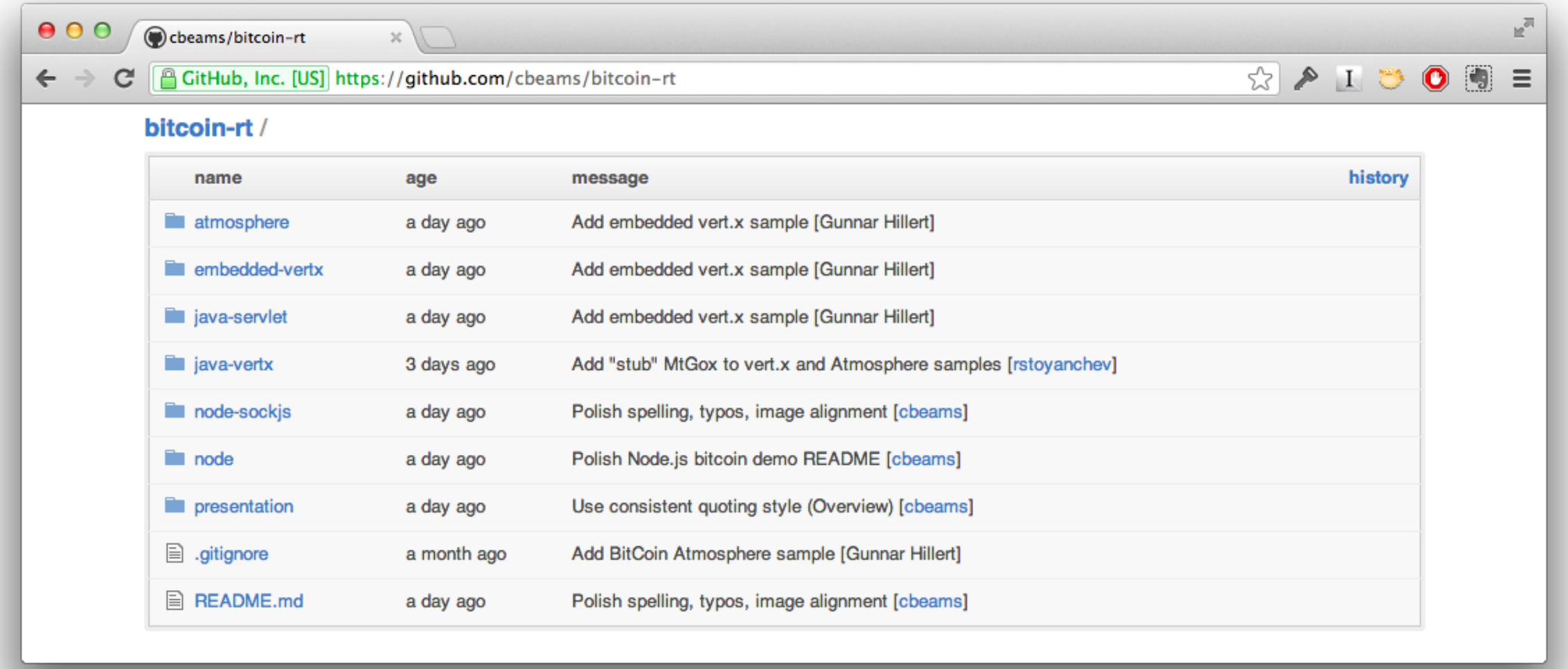
bitcoin-rt Node.js



springsource A division of vmware

bitcoin-rt implementations

- **Node.js** – http://nodejs.org/
- **Node.js + SockJS** – http://sockjs.org
- **Java + Tomcat** native WebSocket API
- **Java + Atmosphere** – https://github.com/Atmosphere
- **Java + Vert.x** – http://vertx.io/



https://github.com/cbeams/bitcoin-rt

bitcoin-rt

Node.js demo

WebSocket benefits

- more resource-efficient
- lower-latency data
- conceptually simpler

If WebSocket is so great...

- Why does bitcoinmonitor use long polling?
- What about other sites?
  - Asana.com
  - Meteor (http://www.meteor.com)

```
self.socket = new SockJS(self.url, undefined, {
      debug: false, protocols_whitelist: [
        // only allow polling protocols. no websockets or streaming.
        // streaming makes safari spin, and websockets hurt chrome.
        'xdr-polling', 'xhr-polling', 'iframe-xhr-polling', 'jsonp-polling'
      ]});
```

github.com/meteor/meteor/blob/master/packages/stream/stream_client.js

Browser Support

Web Sockets - Working Draft

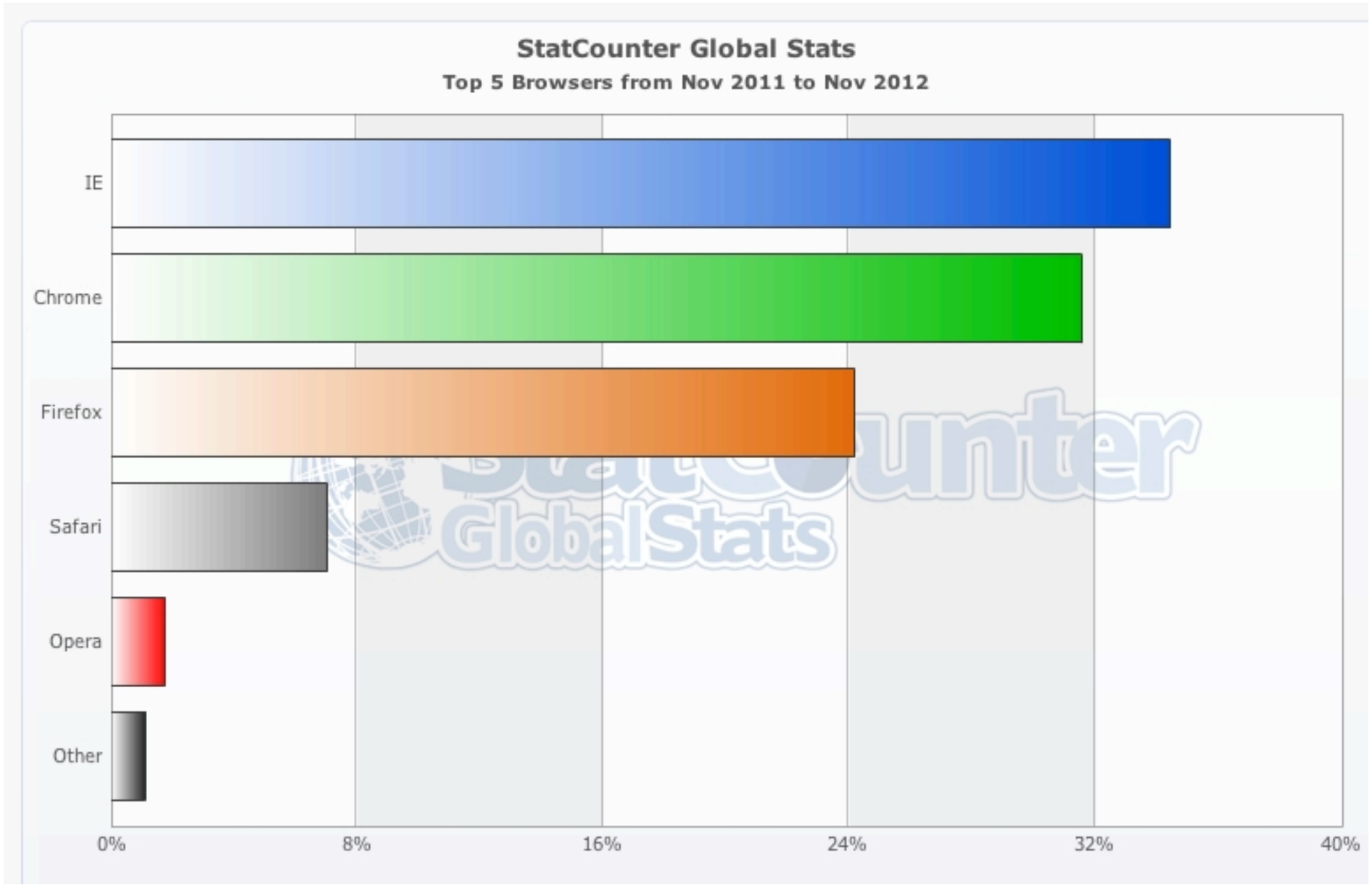
Bidirectional communication technology for web apps

*Usage stats:	Global
Support:	55.2%
Partial support:	5.82%
Total:	61.02%

	IE	Firefox	Chrome	Safari	Opera	IOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android
19 versions back			4.0									
18 versions back			5.0									
17 versions back		2.0	6.0									
16 versions back		3.0	7.0									
15 versions back		3.5	8.0									
14 versions back		3.6	9.0									
13 versions back		4.0	10.0									
12 versions back		5.0	11.0									
11 versions back		6.0 Moz	12.0									
10 versions back		7.0 Moz	13.0		9.0							
9 versions back		8.0 Moz	14.0		9.5-9.6							
8 versions back		9.0 Moz	15.0		10.0-10.1							
7 versions back		10.0 Moz	16.0		10.5							
6 versions back		11.0	17.0		10.6							
5 versions back	5.5	12.0	18.0	3.1	11.0			2.1				
4 versions back	6.0	13.0	19.0	3.2	11.1	3.2		2.2		10.0		
3 versions back	7.0	14.0	20.0	4.0	11.5	4.0-4.1		2.3		11.0		
2 versions back	8.0	15.0	21.0	5.0	11.6	4.2-4.3		3.0		11.1		
Previous version	9.0	16.0	22.0	5.1	12.0	5.0-5.1		4.0		11.5		
Current	10.0	17.0	23.0	6.0	12.1	6.0	5.0-7.0	4.1	7.0	12.0	18.0	15.0
Near future		18.0	24.0		12.5				10.0	12.1		
Farther future		19.0	25.0									

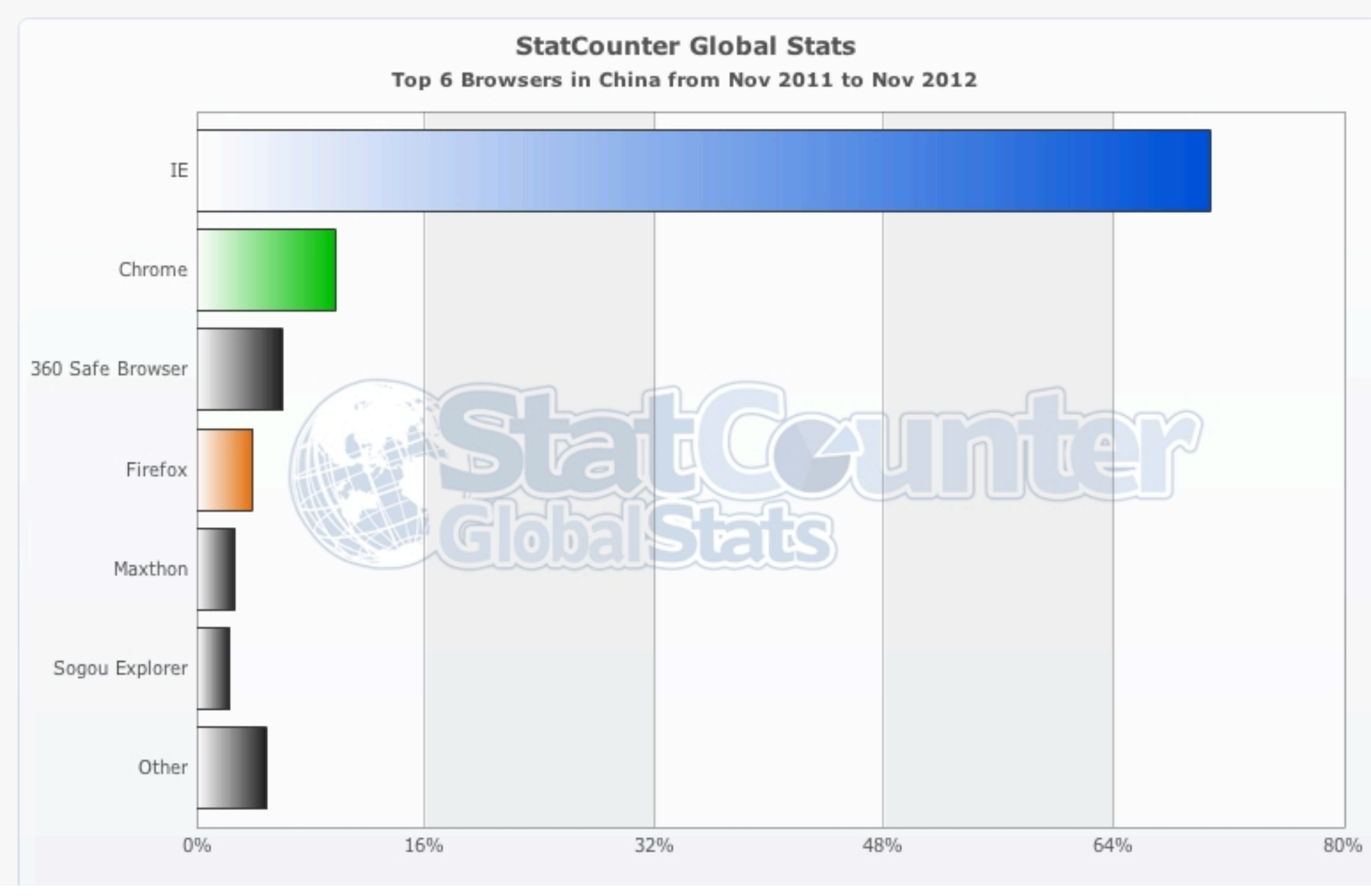
http://caniuse.com/#feat=websockets (Dec 17, 2012)

Browser Share World-Wide



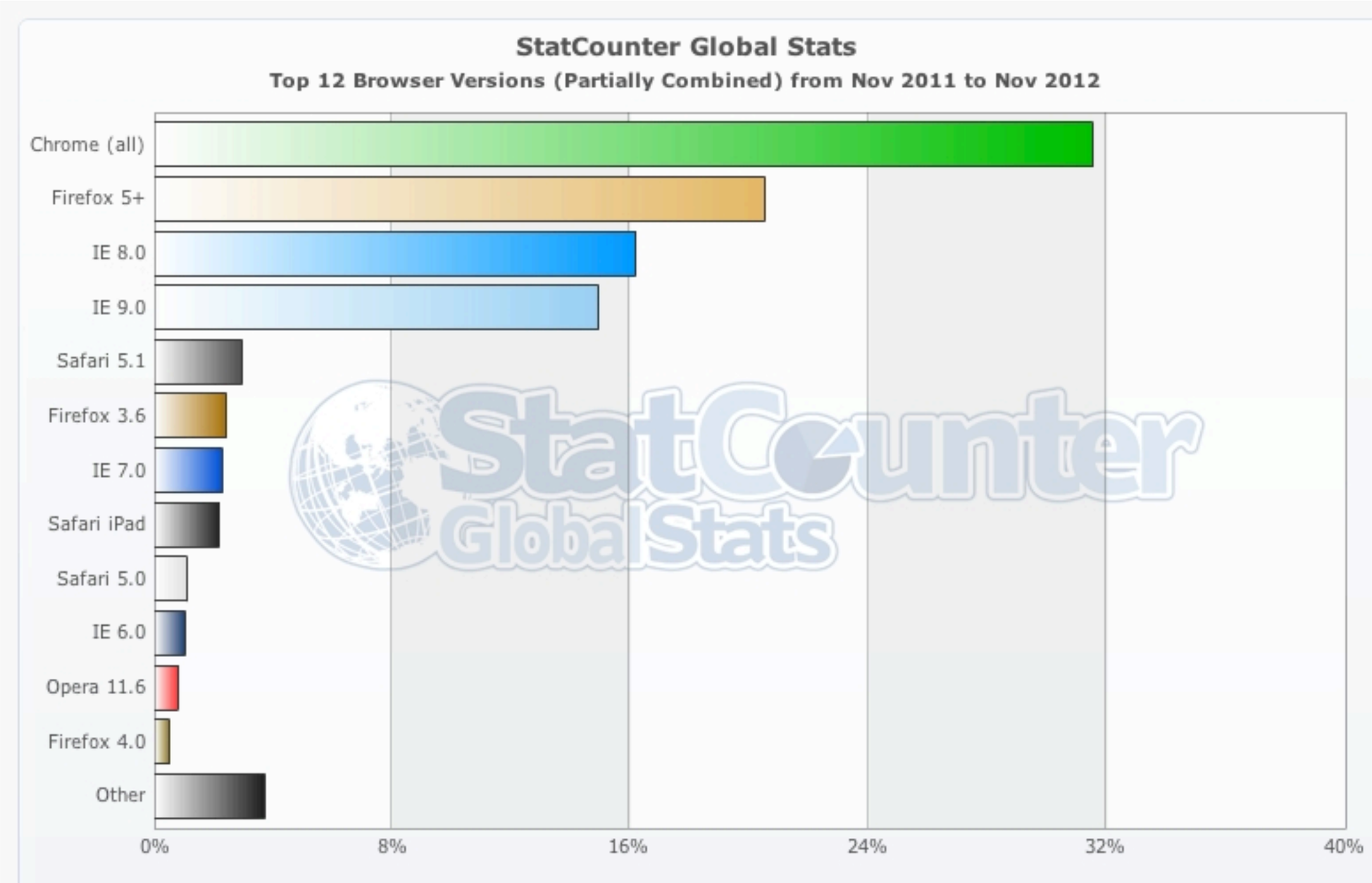
http://gs.statcounter.com/

Browser Share China



http://gs.statcounter.com/

Browser Versions World-Wide



http://gs.statcounter.com/

HTTP Proxies

- Content caching, internet connectivity, filtering
- Can monitor or close connections, buffer unencrypted traffic
- Designed for HTTP-based document transfer
- Not for long-lived connections

Proxy Traversal

"Today, most transparent proxy servers will not yet be familiar with the Web Socket protocol and these proxy servers will be unable to support the Web Socket protocol"

Peter Lubbers, in a 2010 InfoQ article

Proxy Issues

- Explicit proxies with HTTP Connect
- Transparent proxies propagation of *Upgrade* header
- Retaining the *Connection* header
- WebSocket frames vs HTTP traffic

A Few Rules of Thumb

- “wss:” provides a much better chance of success
- Same for browsers using explicit proxies
- Transparent proxies can support WebSocket
- But must be configured explicitly

Keeping Connections Alive

- Internet inherently unreliable
- Both server and client can go away
- Wireless connection may fade out
- and so on

A New Set of Challenges

- Keep-alive ("ping!")
- Heartbeat ("I'm still here!")
- Message delivery guarantee
- Buffering

How Did We Get Here?

Some History

- 1996 - Java Applets/Netscape 2.0
- 1999/2000 - XMLHttpRequest (XHR)
- 2003 - Macromedia/Adobe Flash (RTMP Protocol)

Comet

- March 2006 - Comet - Alex Russell
- event-driven, server-push data streaming
- e.g. in GMail's GTalk interface

Comet

- XHR long-polling / XHR multipart-replace / XHR Streaming
- htmlfile ActiveX Object
- Server-sent events (SSE) - Part of HTML5/W3C (EventSource)
  - http://www.html5rocks.com/en/tutorials/eventsource/basics/

Path to WebSockets

- 2007 - TCPConnection API and protocol (Ian Hickson)
- WebSocket - First public draft January 2008

IETF Standardization

Network Working Group

- 2009-Jan - hixie-00
- 2010-Feb - hixie-75 - Chrome 4
- 2010-May - hixie-76 - Disabled in FF/Opera

HyBi Working Group

- 2010-May - hybi-00 - Same as hixie-76
- 2011-April - hybi-07 - Firefox 6
- 2011-Dec - **RFC6455**

RFC 6455
The WebSocket Protocol

Final Version: Dec 2011

http://tools.ietf.org/html/rfc6455

WebSocket Protocol Details

- TCP-based protocol
- HTTP used solely for upgrade request (**Status Code 101**)
- Bi-directional, full-duplex
- Data Frames can be **Text** (UTF-8) or arbitrary **Binary** data

WebSocket Schemes

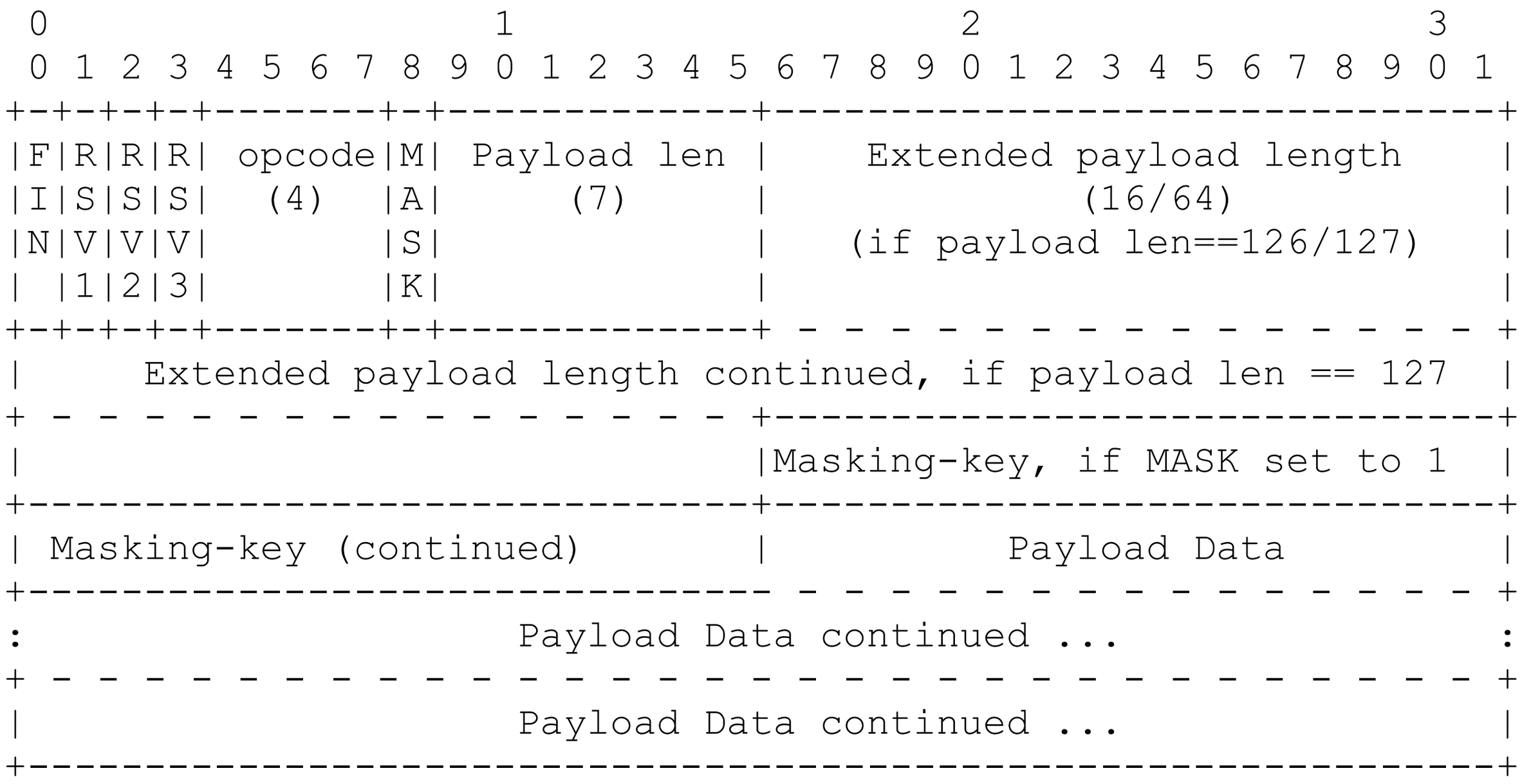
- Unencrypted: **ws://**
- Encrypted: **wss://**

- **Use encrypted scheme**

WebSocket Handshake

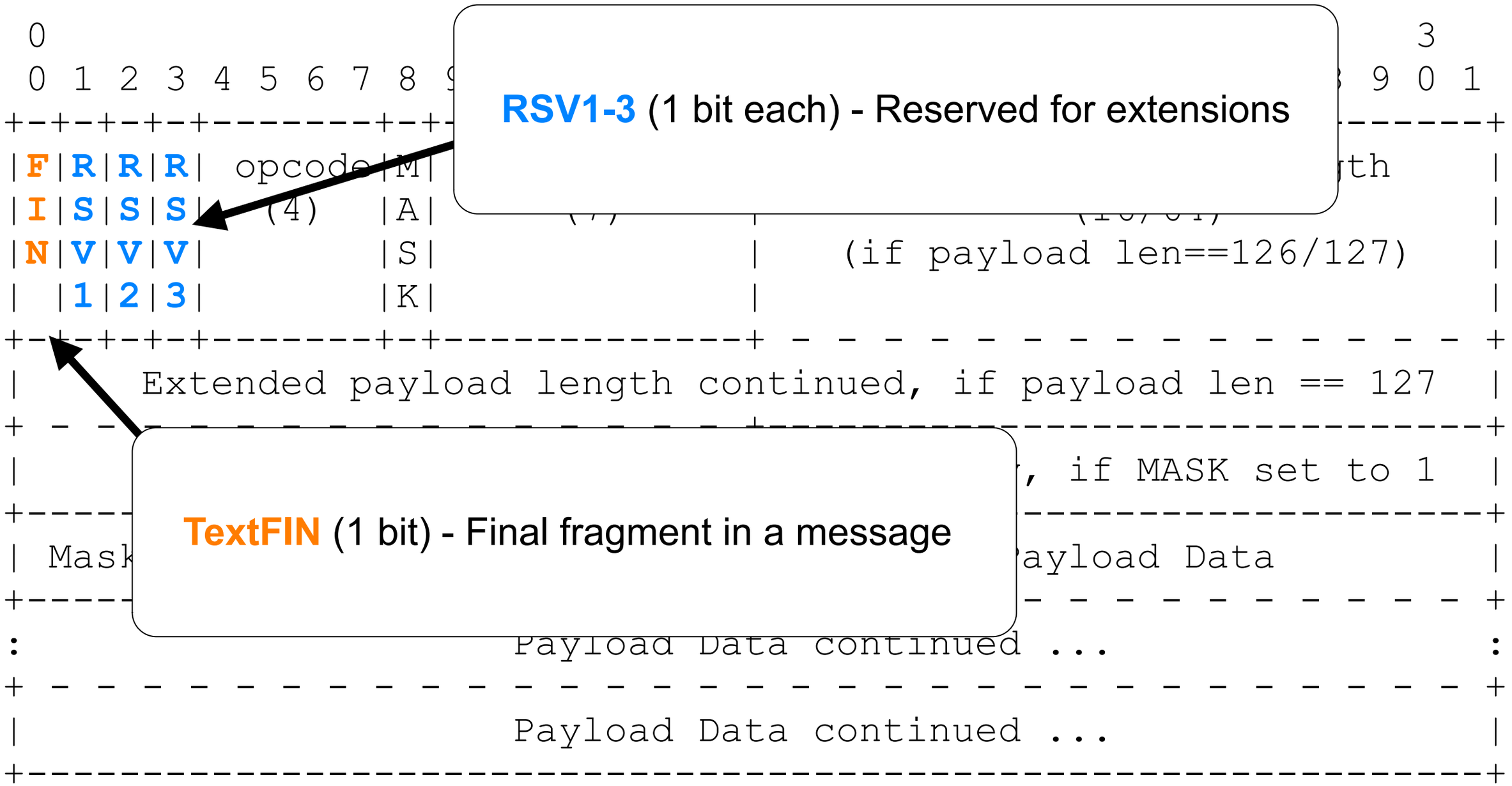
- Request: Sec-WebSocket-Key Header
- Response - 258EAFA5-E914-47DA-95CA-C5AB0DC85B11
- Appended to key + SHA-1 + base64
- Sec-WebSocket-Accept Header

WebSocket Protocol Details



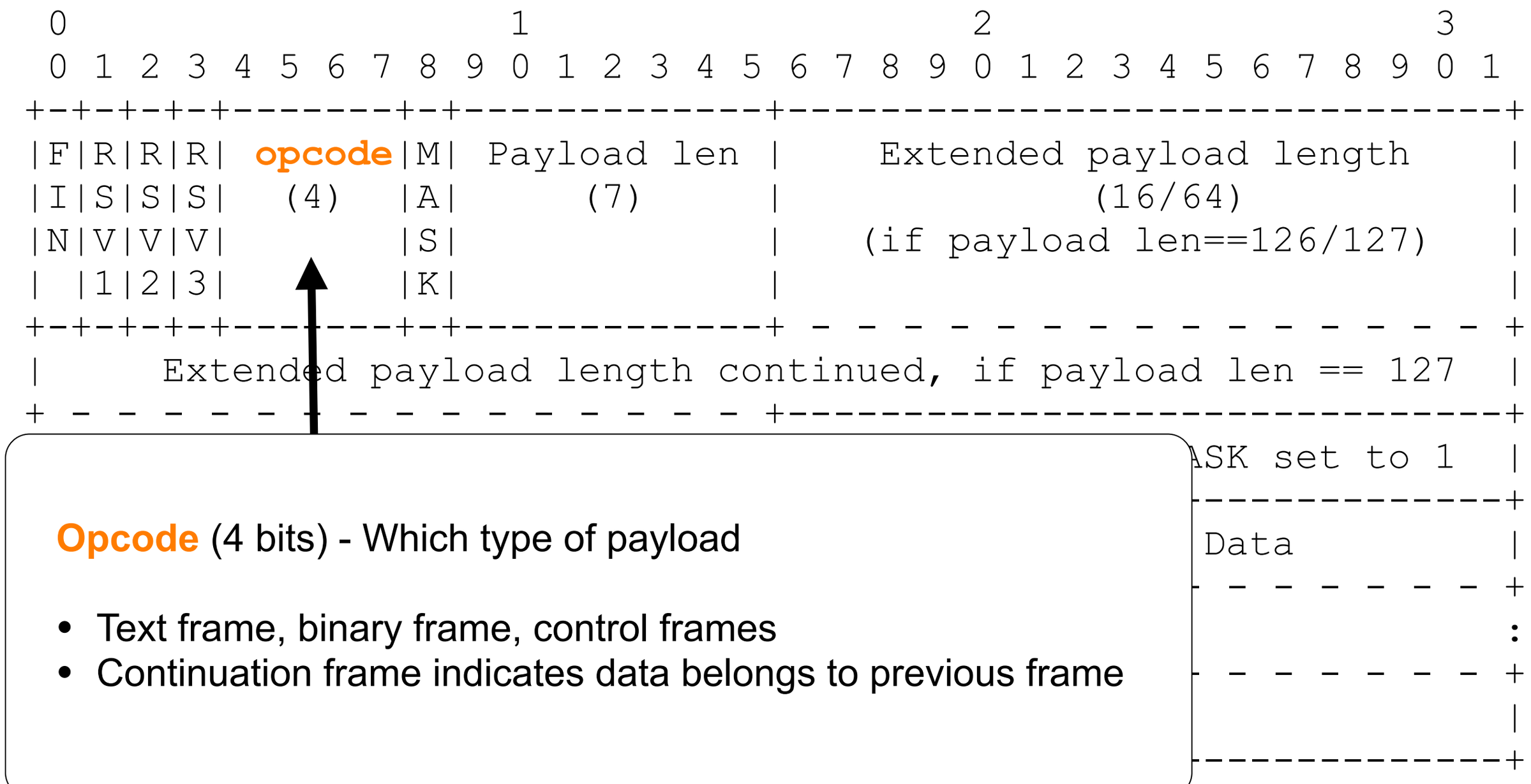
http://www.ietf.org/rfc/rfc6455.txt

WebSocket Protocol Details



http://www.ietf.org/rfc/rfc6455.txt

WebSocket Protocol Details

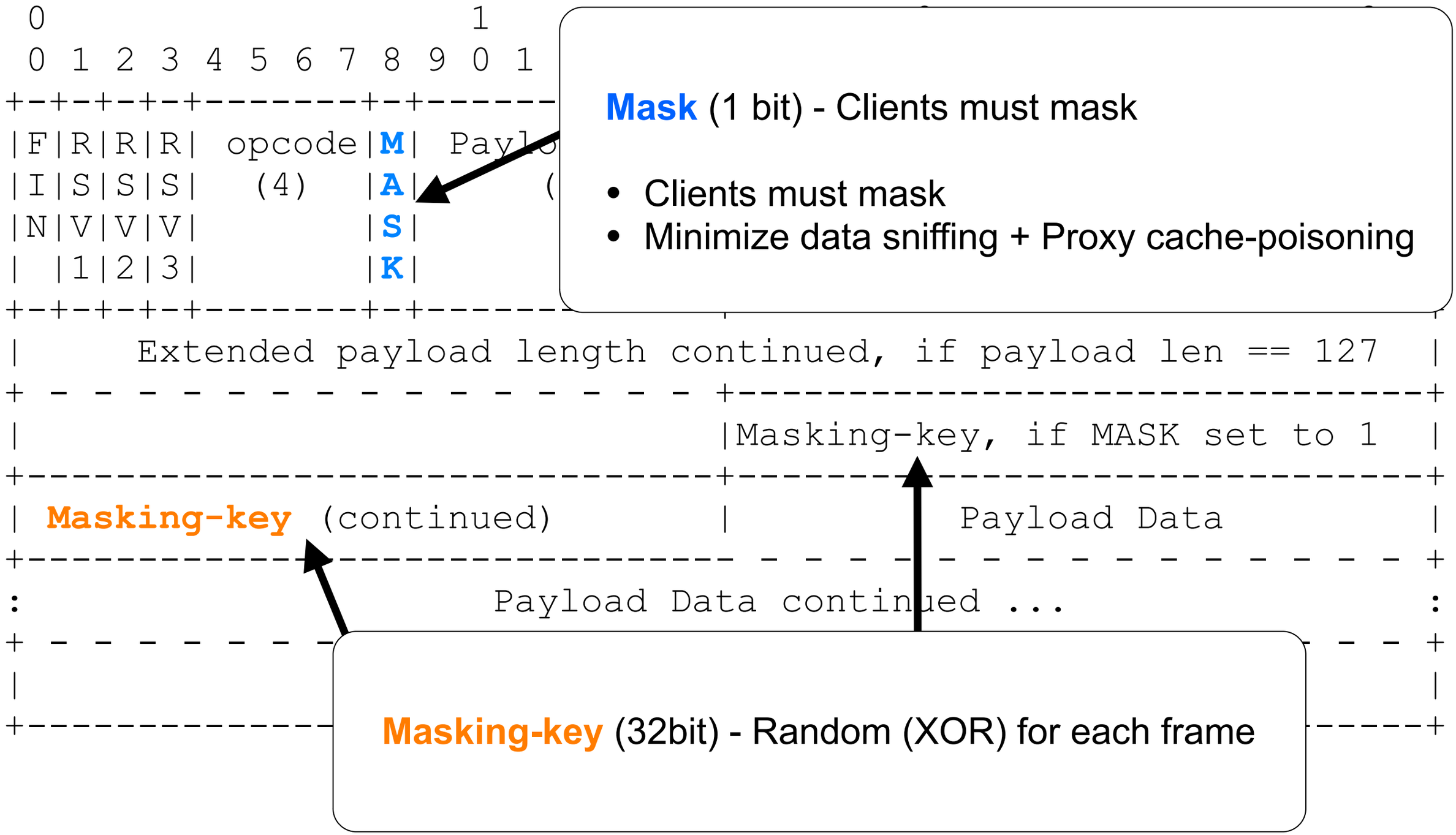


Opcode (4 bits) - Which type of payload

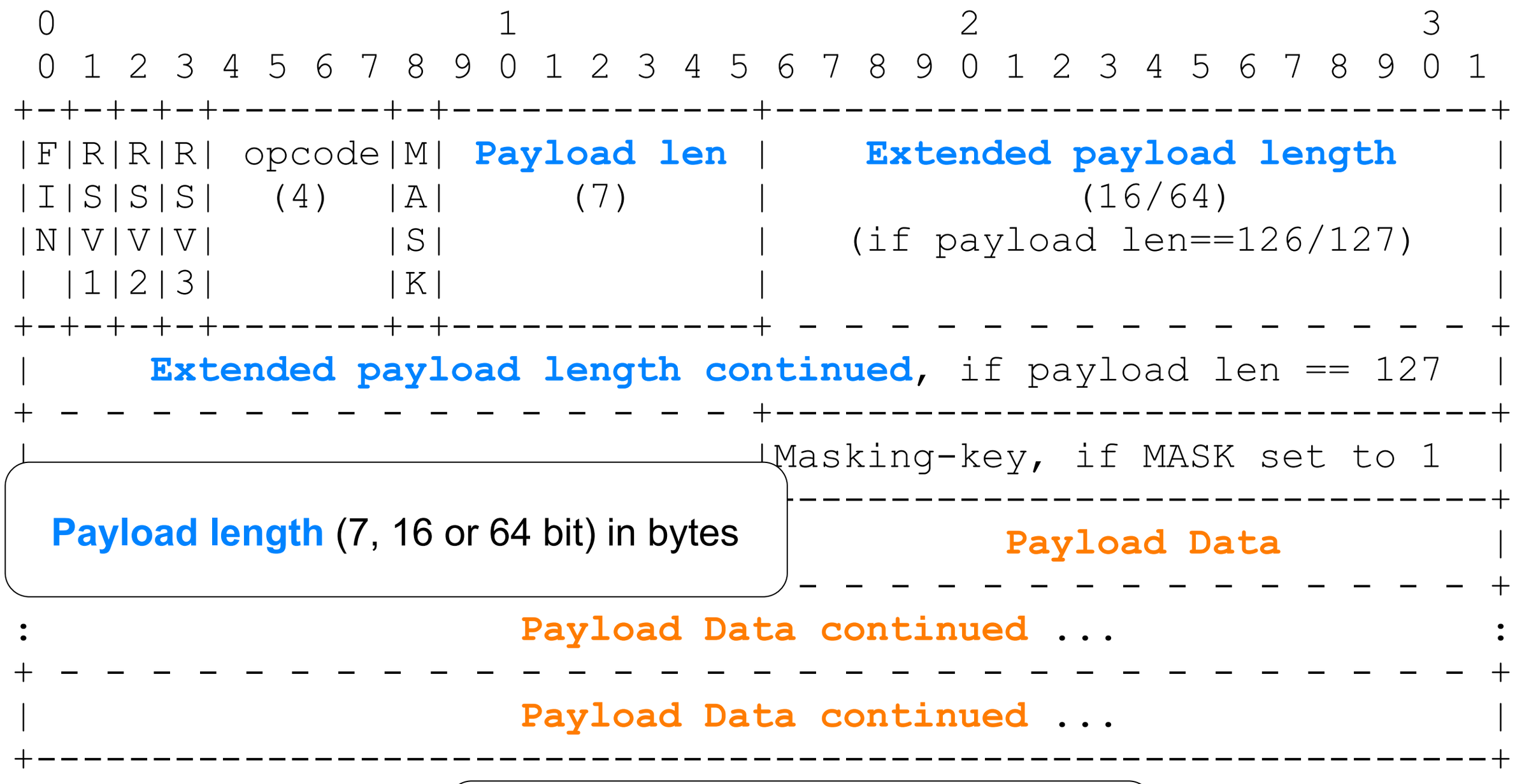
- Text frame, binary frame, control frames
- Continuation frame indicates data belongs to previous frame

http://www.ietf.org/rfc/rfc6455.txt

WebSocket Protocol Details



WebSocket Protocol Details



Payload length (7, 16 or 64 bit) in bytes

Extension data + Application data

WebSocket Control Frames

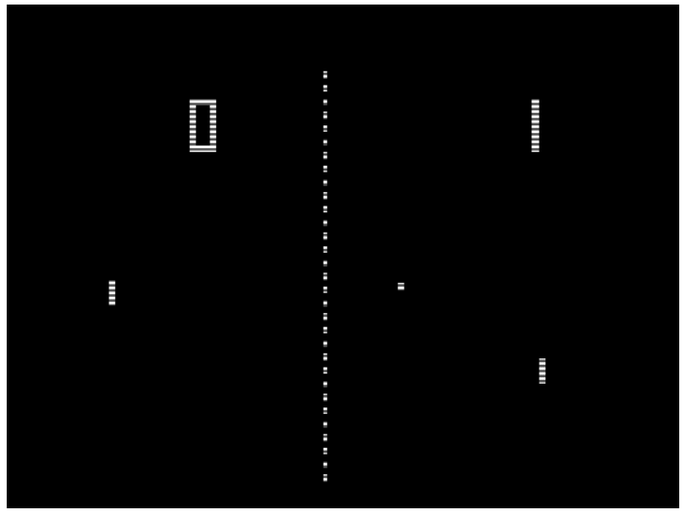
- Communicate state about the WebSocket
- Close (0x8)
- Ping (0x9)
- Pong (0xA)
- More possible in future
- 125 bytes or less

Close Frame

- Terminates WebSocket connection
- Can contain a body (UTF-8 encoded)
- Defines a set of Status Codes, e.g:
  - 1000 = normal closure
  - 1001 = endpoint is “going away”

Ping + Pong Frame

- Serves as **keepalive** (Ping followed by Pong)
- Check whether the remote endpoint is still responsive
- Can be sent at any time (WebSocket established, before close)
- Just Pongs (unsolicited) = unidirectional **heartbeat**



WebSocket Extensions

- WebSocket Per-frame Compression (Draft)
- Multiplexing Extension (Draft)
- Extensions Header: **Sec-WebSocket-Extensions**

- Used in the opening handshake (HTTP)

Multiplexing Extension (MUX)

- http://tools.ietf.org/html/draft-ietf-hybi-websocket-multiplexing-08
- Separate logical connections over underlying transport connection

Sub-Protocols

- Sub-Protocol Header: **Sec-WebSocket-Protocol**
- IANA Registry:

  http://www.iana.org/assignments/websocket/websocket.xml

  – STOMP
  – WAMP
  – soap (WTF?)

HTML5 WebSockets =

W3C API + IETF Protocol

The WebSocket API

- W3C Candidate Recommendation 20 Sep 2012
- http://www.w3.org/TR/websockets/
- Browser client-side API

The WebSocket API

- Binary data supported: **Blob** or **ArrayBuffer** format
- Can inspect extensions (read-only)
- No support for ping/pong frames

The readyState attribute

- **CONNECTING** (0) - Connection not yet established
- **OPEN** (1) - Connection is established + communication possible
- **CLOSING** (2) - Connection going through closing handshake / close() method called
- **CLOSED** (3) - Connection is closed / could not be opened

Event Handlers

- **onopen**
- **onmessage**
- **onerror**
- **onclose**

Code Sample

```
var socket = new WebSocket(
   'ws://localhost:8080/bitcoin-java-
servlet/tomcat');
...
socket.onmessage = function(event) {
        console.log(event.data);
        var trade = JSON.parse(event.data);
        ...
    };
...
```

JSR 356: Java API for WebSocket

- Early Draft Review, latest version Dec 2012
- http://jcp.org/en/jsr/detail?id=356

Non-Java Solutions

Non-Java Solutions

- **Node.js websocket package**
  - https://npmjs.org/package/websocket

- **Socket.IO**
  - http://socket.io

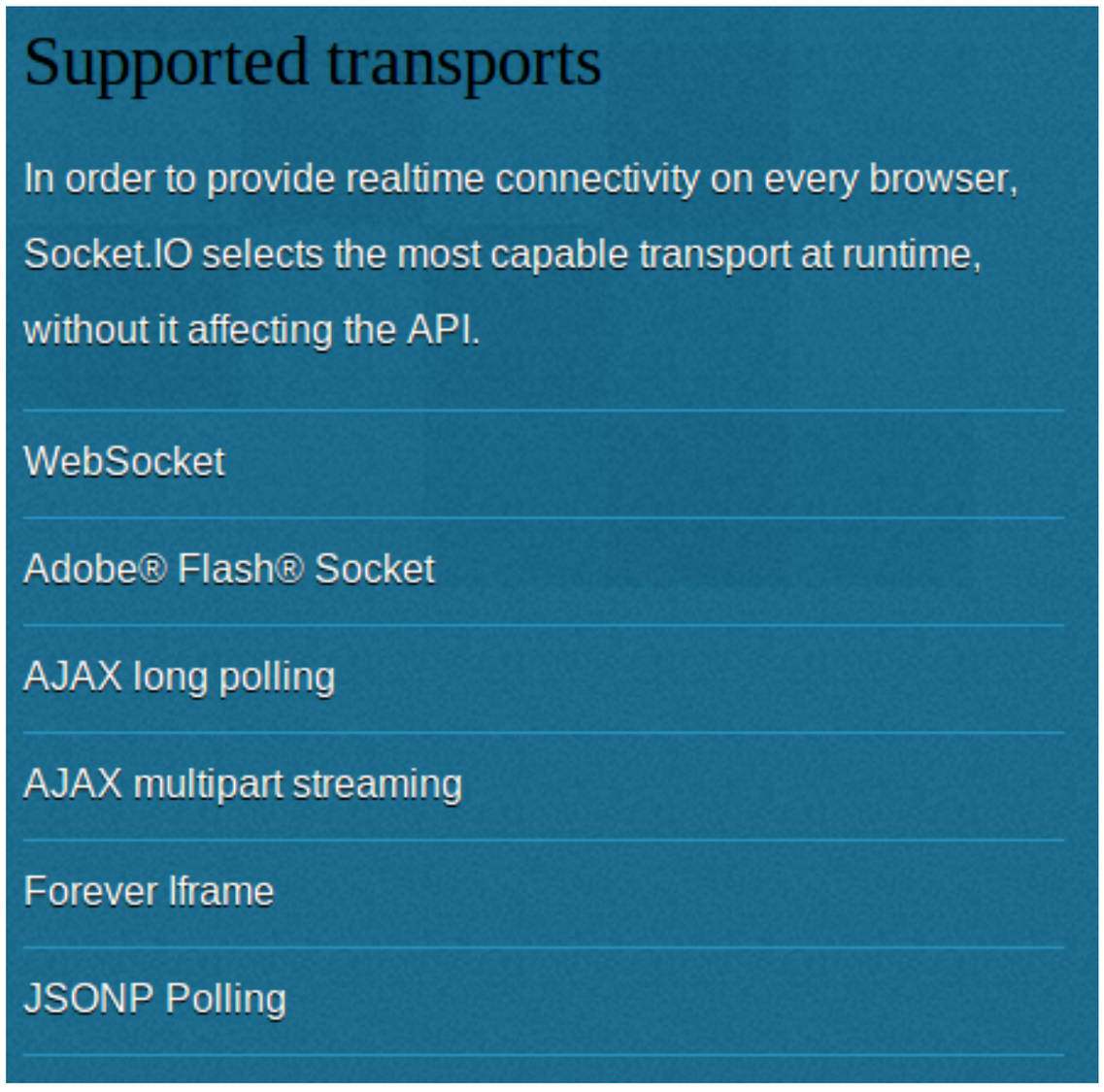
- **SockJS**
  - http://sockjs.org

More Than Just WebSockets

- XHR streaming
- XHR long polling
- Hidden iframe
- Flash socket
- Polling

Socket.IO

- Engine.IO
- Socket.IO

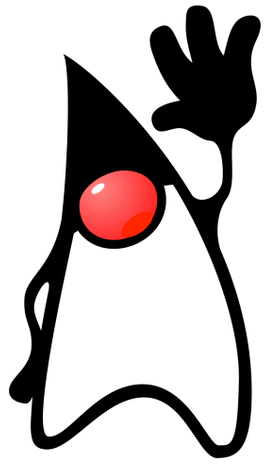


SockJS Transports

Browser	*Websockets*	*Streaming*	*Polling*
IE 6, 7	no	no	jsonp-polling
IE 8, 9 (cookies=no)	no	xdr-streaming †	xdr-polling †
IE 8, 9 (cookies=yes)	no	iframe-htmlfile	iframe-xhr-polling
IE 10	rfc6455	xhr-streaming	xhr-polling
Chrome 6-13	hixie-76	xhr-streaming	xhr-polling
Chrome 14+	hybi-10 / rfc6455	xhr-streaming	xhr-polling
Firefox <10	no ‡	xhr-streaming	xhr-polling
Firefox 10+	hybi-10 / rfc6455	xhr-streaming	xhr-polling
Safari 5	hixie-76	xhr-streaming	xhr-polling
Opera 10.70+	no ‡	iframe-eventsource	iframe-xhr-polling
Konqueror	no	no	jsonp-polling

Socket.IO vs SockJS

- Socket.IO more popular, SockJS gaining ground
- SockJS focused on transports, horizontal scalability
- Discussion thread



Where We Are In Java Land

Tomcat

- WebSocketServlet
- Since 7.0.27 (03/2012)
- Backport to 6.0.35 Issue 52918
- Fairly minimal, server-side only

- http://tomcat.apache.org/tomcat-7.0-doc/web-socket-howto.html

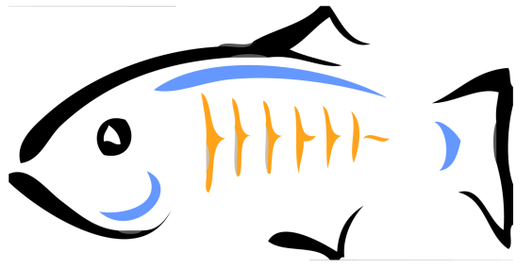
bitcoin-rt: Tomcat demo

Jetty

- Since Jetty 7.x (early adoption, complex)
- Revised in Jetty 9
  - http://webtide.intalio.com/2012/10/jetty-9-updated-websocket-api/
- Builds on Java 7, messages not frames, annotations

  http://download.eclipse.org/jetty/stable-7/apidocs/org/eclipse/jetty/websocket/package-summary.html

Glassfish



- Since 3.1 (02/2011)
- Exposes frames, server-side only
- Like with earlier Jetty versions, a major revision is likely

- http://antwerkz.com/glassfish-web-sockets-sample/

Other Implementations

- **Atmosphere**
  https://github.com/Atmosphere/atmosphere
- **jWebSocket**
  http://jwebsocket.org/
- **Netty.Io**
  https://netty.io/
- **vert.x**
  http://vertx.io/
- **Grizzly**
  http://grizzly.java.net/

Client Side

- **AsyncHttpClient**
  https://github.com/sonatype/async-http-client
- **Jetty**
- **Netty**
- **vert.x**
- **Grizzly**

Java API for WebSocket (JSR-356)

- Original discussion started in JSR-340 (Servlet 3.1)
- Later split out into separate spec
- Servlet spec will have an upgrade option
- JSR-356 will not require Servlet API

What's under discussion

- Client and server-side API
- Use of annotations (or use API directly)
- Support for extensions
- Security considerations
- Thread model

What's under discussion

- Client and server-side API
- Use of annotations (or use API directly)
- Support for extensions
- Security considerations
- Thread model

Resources

- All drafts so far
  http://java.net/projects/websocket-spec/downloads/directory/Spec%20javadoc%20Drafts
- The latest v010 Early Draft Review
  http://java.net/projects/websocket-spec/downloads/directory/Spec%20javadoc%20Drafts/v010
- Mailing list archives
  http://java.net/projects/websocket-spec/lists

Spring Integration WebSocket Support

- Atmosphere based Extension (Coming)
- WebSocket implementation using TCP Adapters (demo)
- Considering adding Client Support (SockJS)
- Event Bus support (Integration with Integration.js)

Building a Non-Trivial Application

A Few Conclusions

- WebSocket technology is promising
- Not a silver bullet
- Complement to REST
- Potential replacement for Comet techniques
- But the need for fallback options will persist

A Few Conclusions

- Integrating WebSockets into a real app is not yet trivial
- But now is the time to begin thinking about it
- “Pure WebSocket” applications in the wild unlikely

Predictions

- A consolidation of 'fallback protocols'
- Leading to wide adoption in various application frameworks
- SockJS currently the most promising effort
  - https://github.com/sockjs/sockjs-protocol

Many questions remain

- Usage patterns
- Higher-level protocols
- XMPP, AMQP, JMS, …

Building a real app today

- Commercial vendors have a lot to offer
- Particularly KAAZING
  - blog: http://blog.kaazing.com/
  - http://www.websocket.org/

- Doing Mobile? Consider Push Technologies
  - Apple Push Notification Service (APNS)
  - Google Cloud Messaging for Android (GCM)
  - Consider URBAN AIRSHIP
  - Spring Mobile provides early support: https://github.com/SpringSource/spring-mobile-urbanairship

Predictions: Java

- JSR-356 will be important
- Frameworks have a big role to play
- Atmosphere is there today
- Dedicated Spring support in consideration

Questions?

Thanks!

http://twitter.com/ghillert

http://cbeams.github.com/bitcoin-rt