JWebSockets: On the road to WebSockets

Brian Gilstrap

Senior Platform Engineer

What are WebSockets?

What are WebSockets?

- Being defined by W3C; let's let them tell us:
  - http://dev.w3.org/html5/websockets/

What are WebSockets?

- Being defined by W3C; let's let them tell us:
  - http://dev.w3.org/html5/websockets/ Hmmm*

What are WebSockets?

- Being defined by W3C; let's let them tell us:
  - http://dev.w3.org/html5/websockets/ Hmmm*
- And the IETF WebIDL spec:
  - http://www.w3.org/TR/WebIDL/

What are WebSockets?

- Being defined by W3C; let's let them tell us:
  - http://dev.w3.org/html5/websockets/ Hmmm*
- And the IETF WebIDL spec:
  - http://www.w3.org/TR/WebIDL/ Errr*

What are WebSockets?

- Being defined by W3C; let's let them tell us:
  - http://dev.w3.org/html5/websockets/ Hmmm*
- And the IETF WebIDL spec:
  - http://www.w3.org/TR/WebIDL/ Errr*
- And the IETF:
  - http://tools.ietf.org/pdf/draft-ietf-hybi-thewebsocketprotocol-09.pdf

What are WebSockets?

- Being defined by W3C; let's let them tell us:
  - http://dev.w3.org/html5/websockets/ Hmmm*
- And the IETF WebIDL spec:
  - http://www.w3.org/TR/WebIDL/ Errr*
- And the IETF:
  - http://tools.ietf.org/pdf/draft-ietf-hybi-thewebsocketprotocol-09.pdf Ack!*

What are WebSockets?

- Being defined by W3C; let's let them tell us:
  - http://dev.w3.org/html5/websockets/ Hmmm*
- And the IETF WebIDL spec:
  - http://www.w3.org/TR/WebIDL/ Errr*
- And the IETF:
  - http://tools.ietf.org/pdf/draft-ietf-hybi-thewebsocketprotocol-09.pdf Ack!*

* Not really a fail - the specs aren't meant to be an introduction

Geez, what <u>are</u> they?

Geez, what <u>are</u> they?

- Maybe Wikipedia can help:
  - http://en.wikipedia.org/wiki/WebSockets

Geez, what <u>are</u> they?

- Maybe Wikipedia can help:
  - http://en.wikipedia.org/wiki/WebSockets
- Okay, that's not great but maybe we can work with it.

The Bullet Points

- Bi-directional, full-duplex communications channels
- Over a single TCP socket.
- Designed to be implemented in web browsers and web servers.
- API - being standardized by the W3C
- Protocol - being standardized by the IETF.

Why WebSockets?

Why WebSockets?

- Each of these features is there for a reason

Why WebSockets?

- Each of these features is there for a reason
- To understand those reasons, let's review some history

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.0

HTTP 1.1

HTTP 1.1

Client
(browser)

Server

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

HTTP 1.1

Client
(browser)

Server

XMLHttpRequest

XMLHttpRequest

Client
(browser)

Server

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

XMLHttpRequest

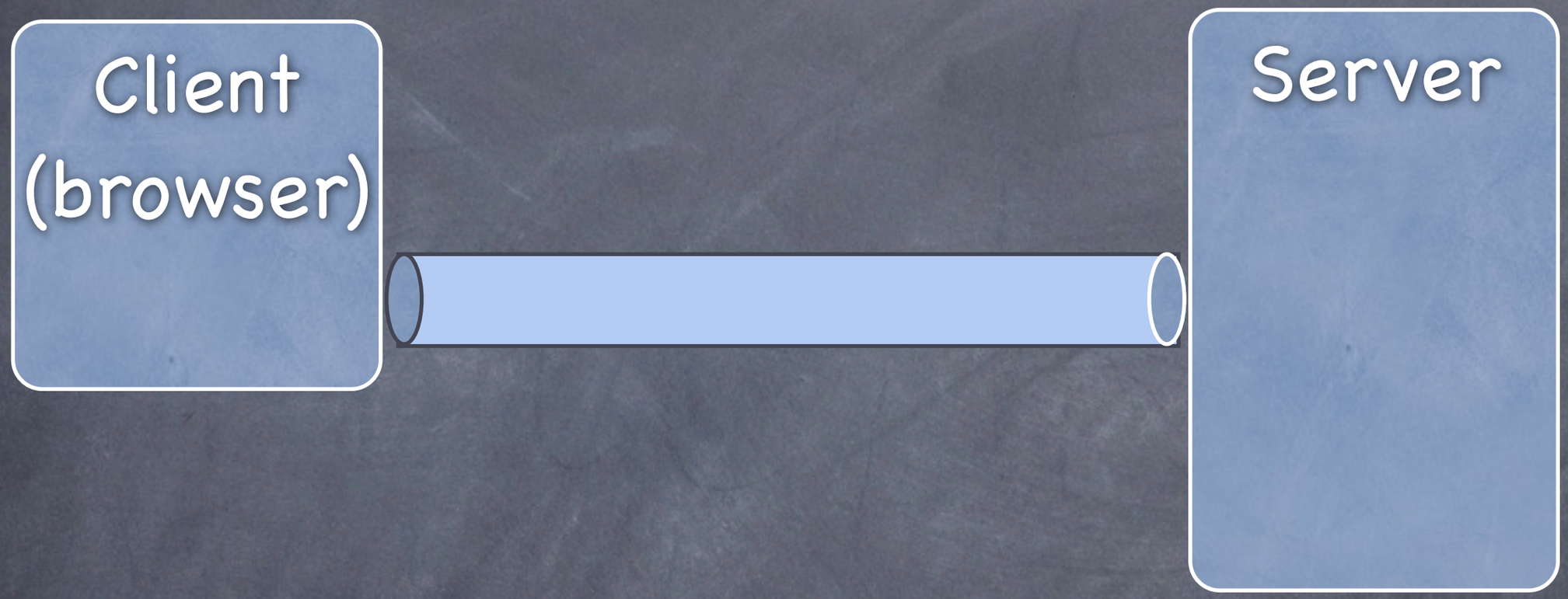
Comet

Comet

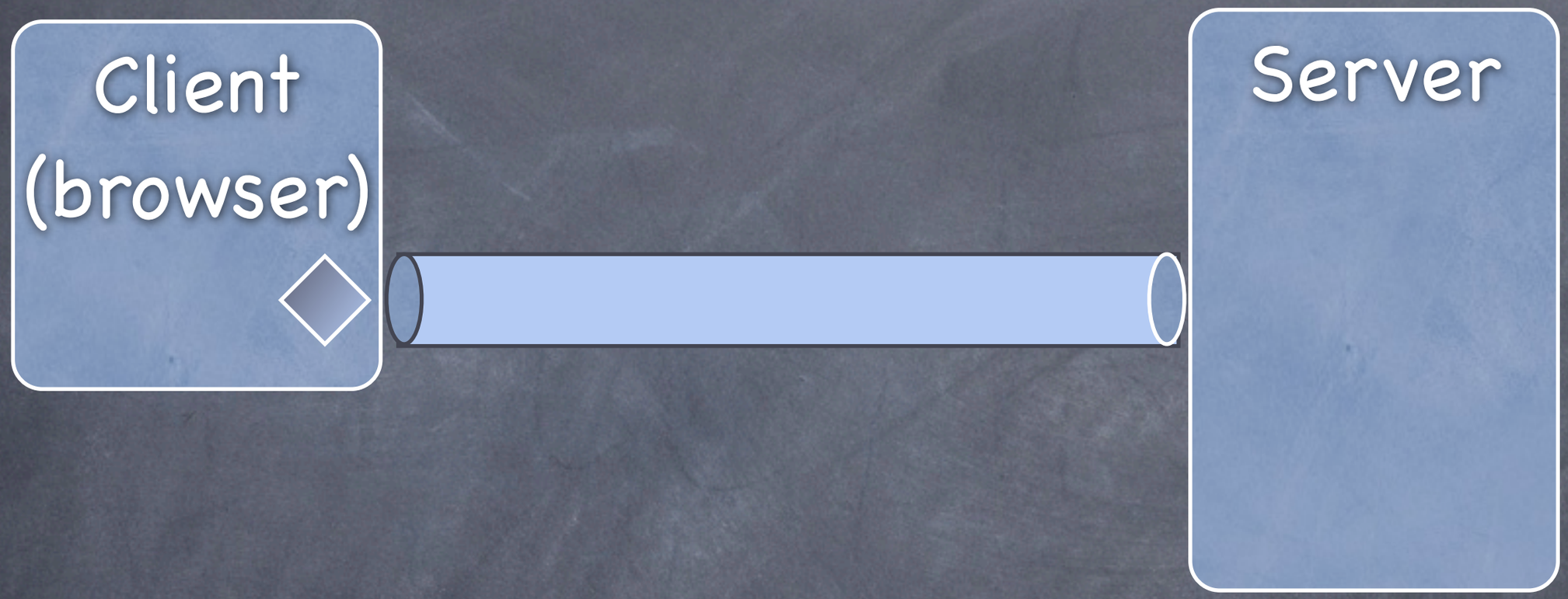
Client
(browser)

Server

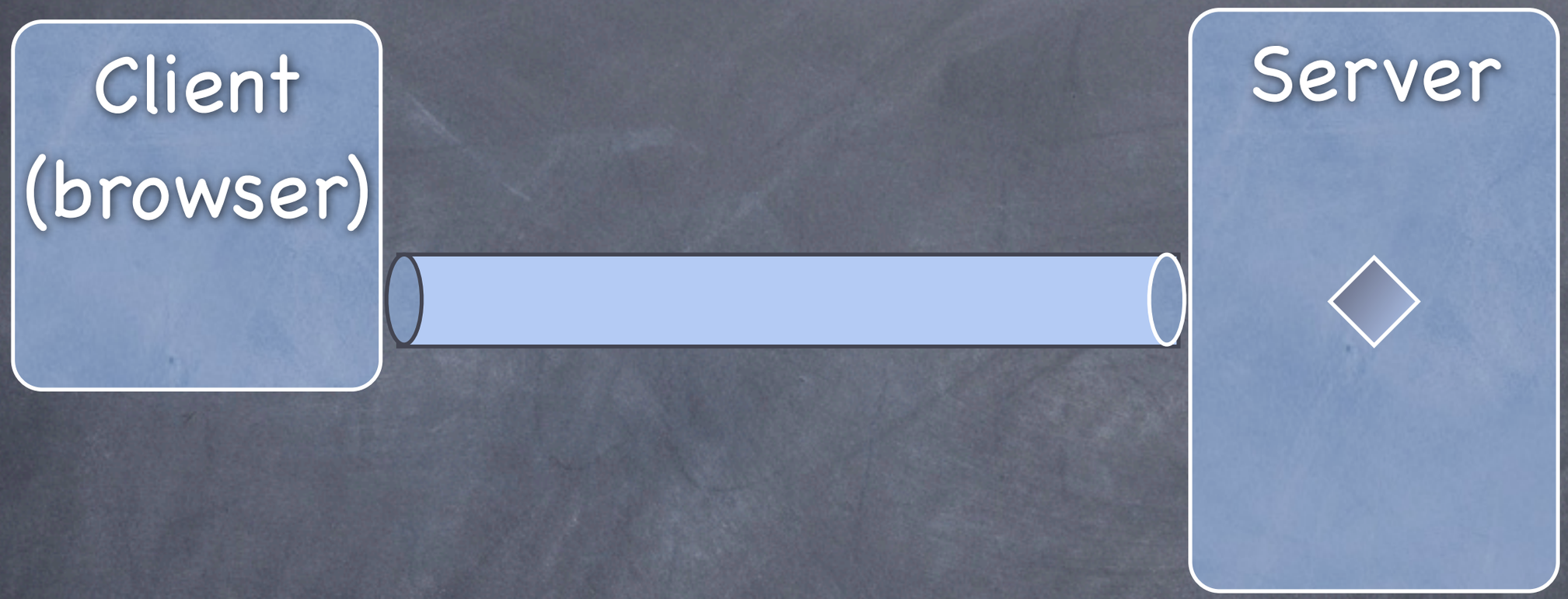
Comet



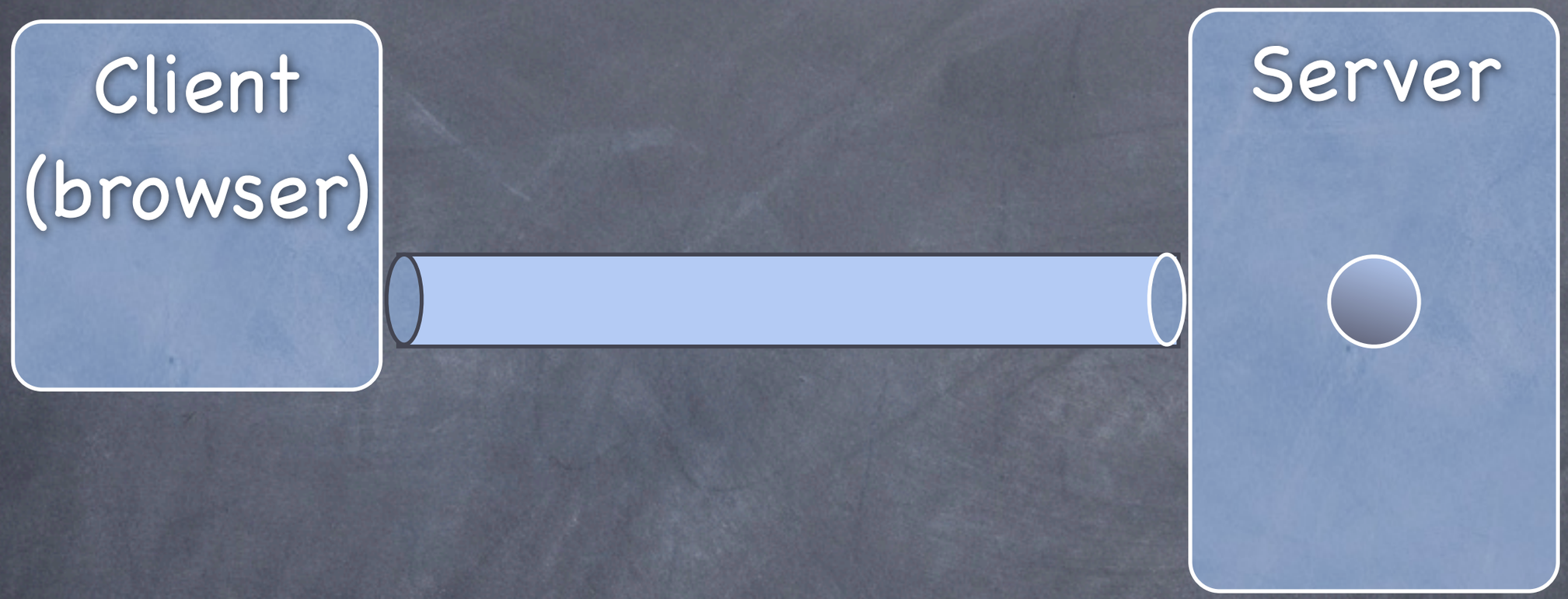
Comet



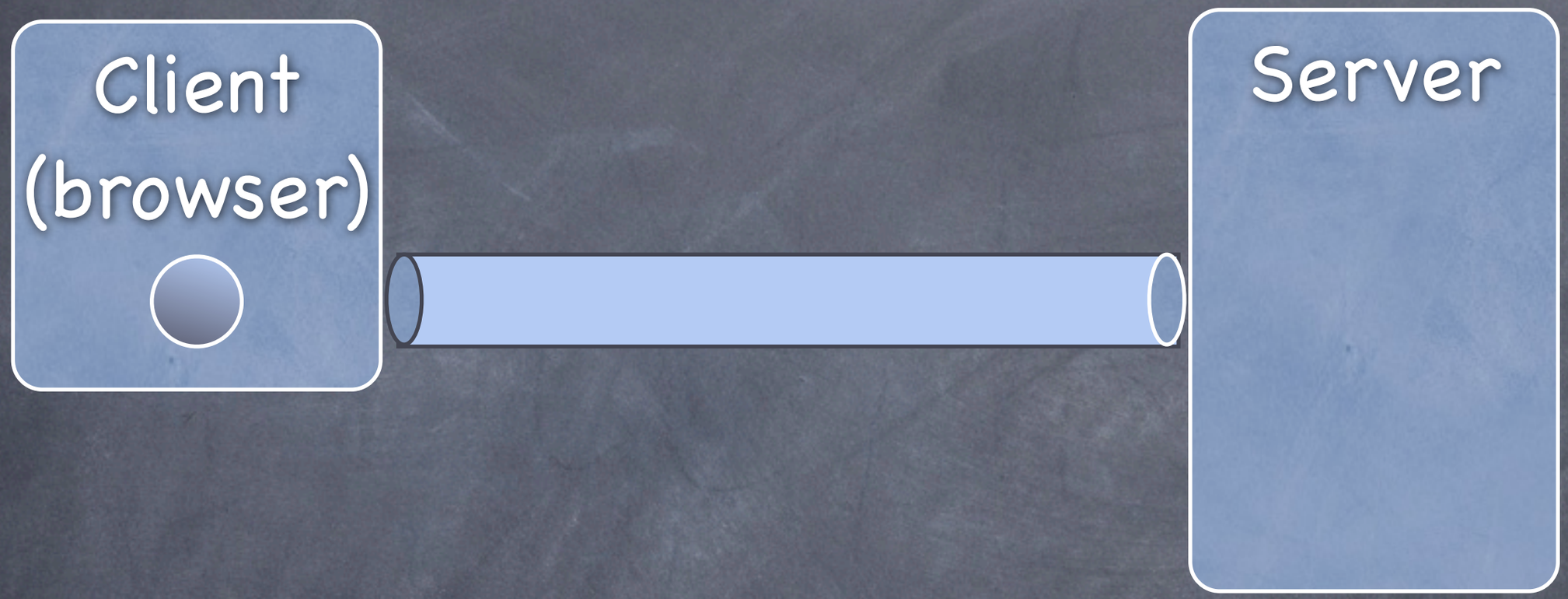
Comet



Comet



Comet

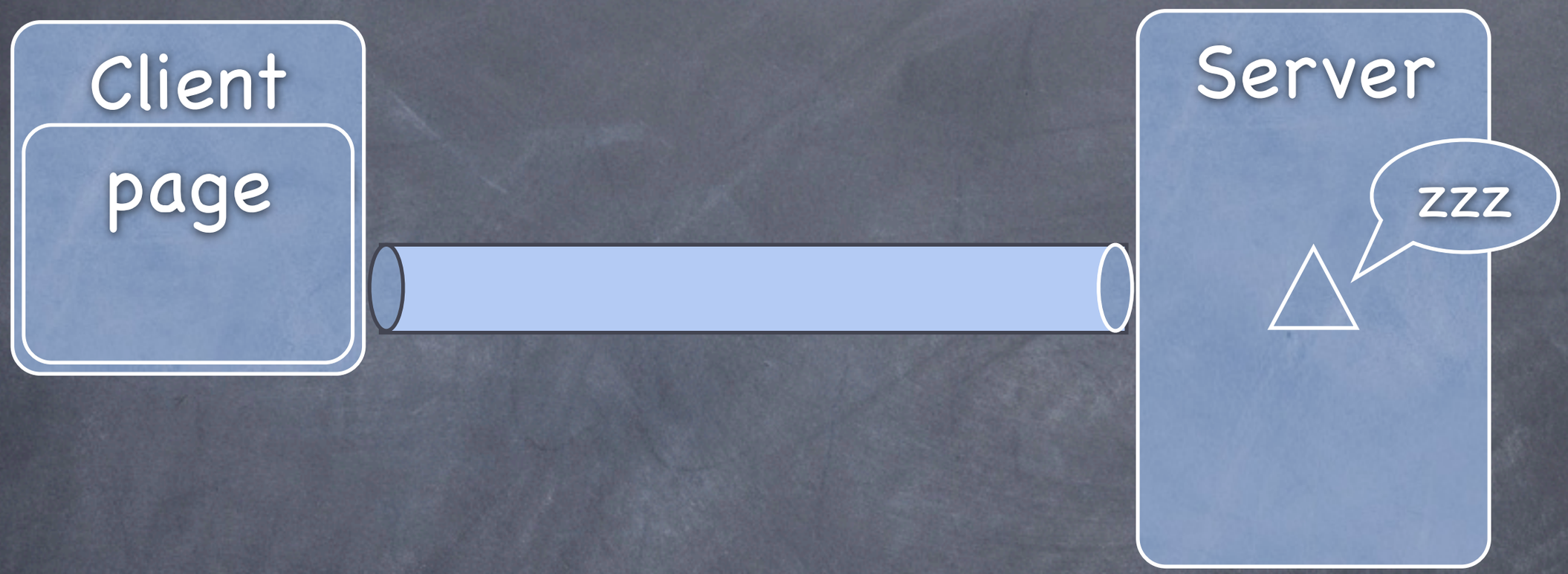


Comet

Comet

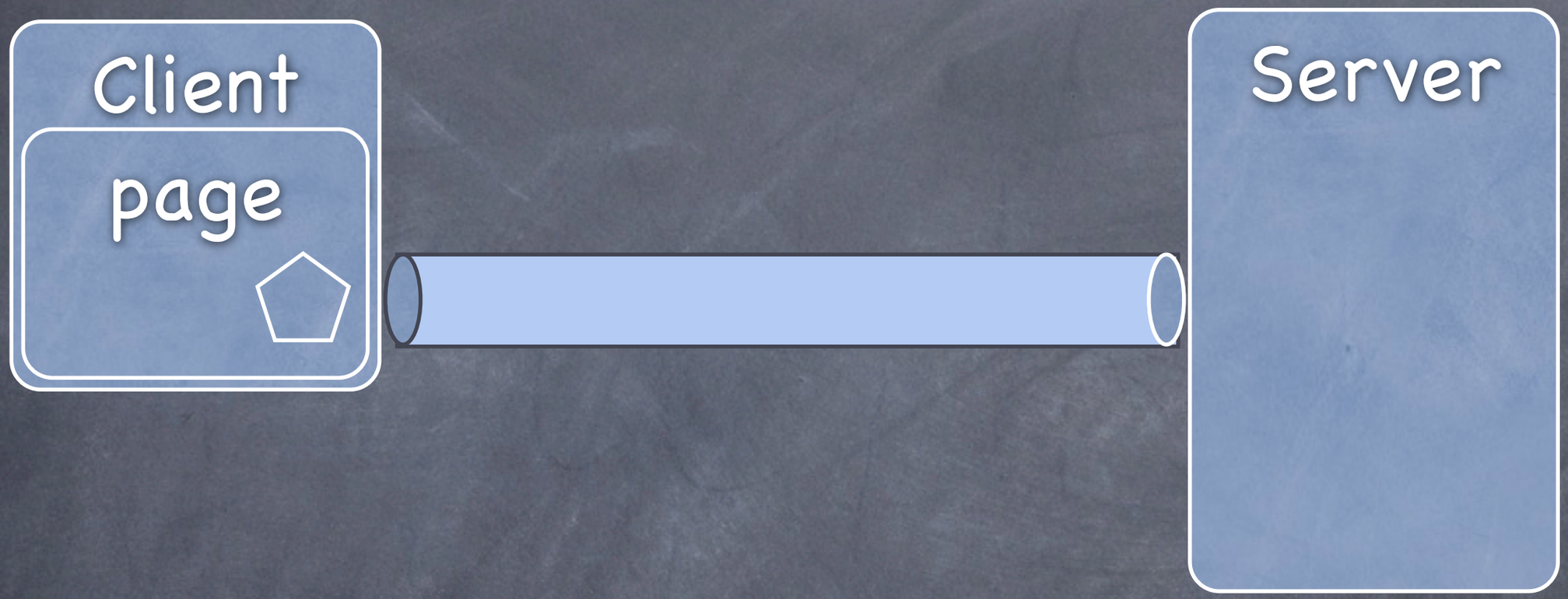
Comet

Comet

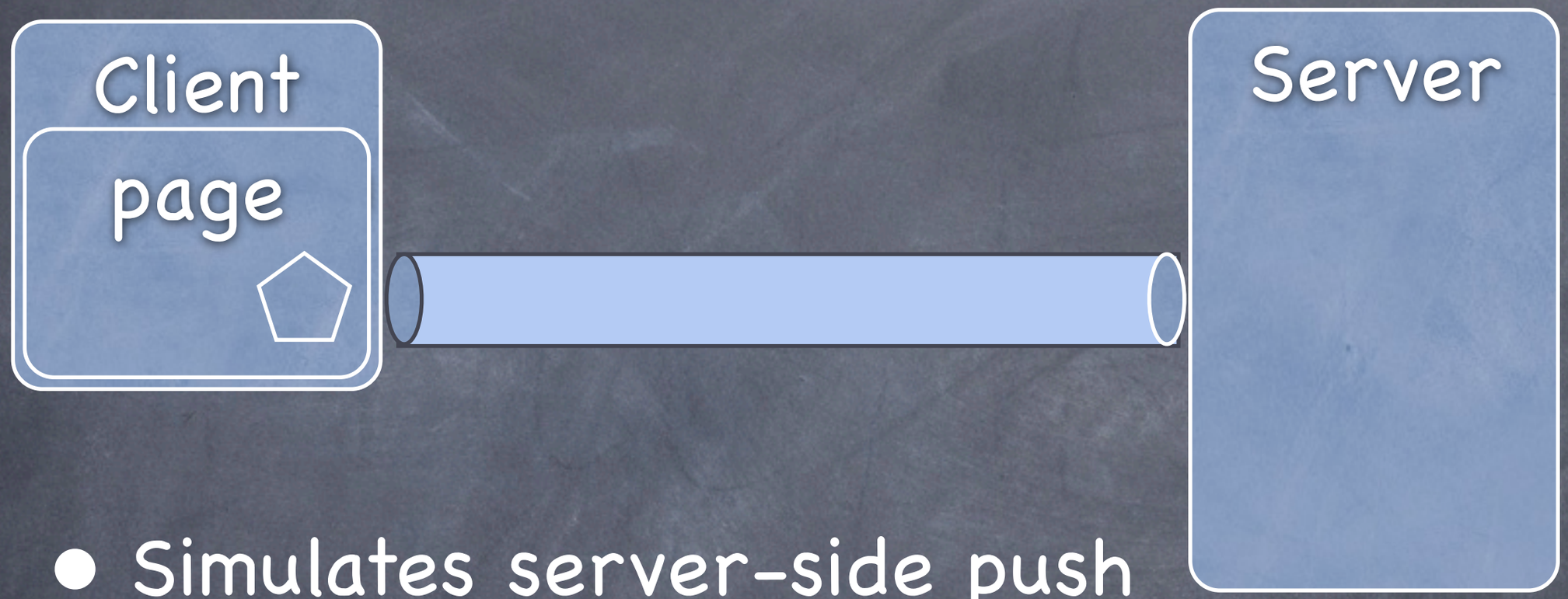


Comet

Comet



Comet



- Simulates server-side push
- Uses “streaming” or “long polling”
- Long polling times out
- Streaming is browser-specific

Revisiting features

- Bi-directional, full-duplex communications channels
- Over a single TCP socket.
- Designed to be implemented in web browsers and web servers.
- API - being standardized by the W3C
- Protocol - being standardized by the IETF.

Bi-Directional

Bi-Directional

- Already had client-initiated communication

Bi-Directional

- Already had client-initiated communication
- Now get server-initiated communication

Bi-Directional

- Already had client-initiated communication
- Now get server-initiated communication
- Without losing client-initiated requests

Full-duplex

Full-duplex

- Anyone know what “full-duplex” means?

Full-duplex

- Anyone know what "full-duplex" means?
- No coordination of whose "turn" it is to send

Full-duplex

- Anyone know what "full-duplex" means?
- No coordination of whose "turn" it is to send
- No "speaking on top of each other"

Over a single TCP socket

- Sockets support bi-directional full-duplex communication
- Why use more than you need to?
- Well-established in the internet
- No “halfway open” problems

Designed for Browsers and Servers

- Duh...

API

- Standardize the API for web browsers
- Security Issues
- API behavior

API (2)

```
[Constructor(in DOMString url, in optional DOMString protocols),
 Constructor(in DOMString url, in optional DOMString[] protocols)]
interface WebSocket {
  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 Function? onopen;
           attribute Function? onerror;
           attribute Function? onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] in optional unsigned short code,
             in optional DOMString reason);

  // messaging
           attribute Function? onmessage;
           attribute DOMString binaryType;
  void send(in DOMString data);
  void send(in ArrayBuffer data);
  void send(in Blob data);
};
WebSocket implements EventTarget;
```

API (2)

```
[Constructor(in DOMString url, in optional DOMString protocols),
 Constructor(in DOMString url, in optional DOMString[] protocols)]
interface WebSocket {
  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 Function? onopen;
           attribute Function? onerror;
           attribute Function? onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] in optional unsigned short code,
             in optional DOMString reason);

  // messaging
           attribute Function? onmessage;
           attribute DOMString binaryType;
  void send(in DOMString data);
  void send(in ArrayBuffer data);
  void send(in Blob data);
};
WebSocket implements EventTarget;
```

API (2)

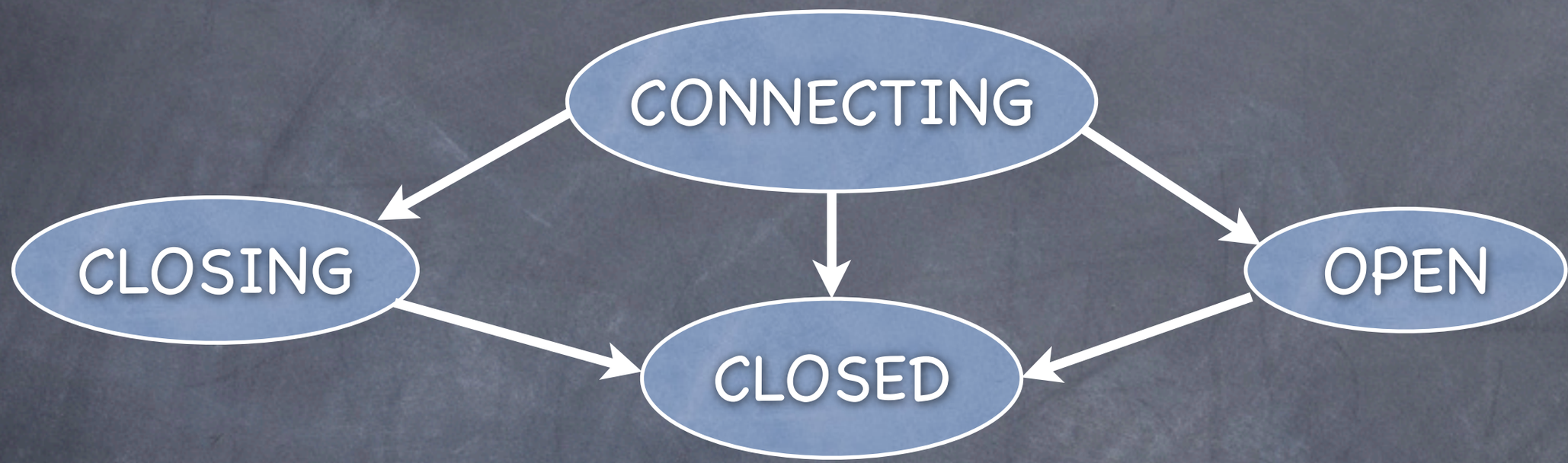
```
[Constructor(in DOMString url, in optional DOMString protocols),
 Constructor(in DOMString url, in optional DOMString[] protocols)]
interface WebSocket {
  readonly attribute DOMString url;

  // ready state
  const unsigned
  const unsigned
  const unsigned
  const unsigned
  readonly attri
  readonly attri

  // networking
          attri
          attri
          attri
  readonly attri
  readonly attri
  void close([Cl
            in

  // messaging
          attribute Function? onmessage;
          attribute DOMString binaryType;
  void send(in DOMString data);
  void send(in ArrayBuffer data);
  void send(in Blob data);
};
WebSocket implements EventTarget;
```

DOM Level 2 [edit]

The W3C designed a more flexible event handling model in DOM Level 2.

Name	Description	Argument type	Argument name
addEventListener	Allows the registration of event listeners on the event target.	DOMString	type
		EventListener	listener
		boolean	useCapture
removeEventListener	Allows the removal of event listeners from the event target.	DOMString	type
		EventListener	listener
		boolean	useCapture
dispatchEvent	Allows to send the event to the subscribed event listeners.	Event	evt

API (3)



- Three types of data supported
  - UTF-8 text
  - Blob
  - ArrayBuffer

Protocol

- Protocol negotiation
- Security issues
- Efficient implementations
- Sane behavior in the face of network issues
- Multi-hop issues (proxy traversal)

Protocol Negotiation Problem

- Browsers talk HTTP(S)
- Companies often limit browser access to ports 80 & 443
- How to bootstrap the socket from a browser?

Answer: Pretend to be HTTP

- Client

```
GET /chat HTTP/1.1 Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 8
```

- Server

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

Answer: Pretend to be HTTP

- Client

```
GET /chat HTTP/1.1 Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Sec-WebSocket-Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 8
```

- Server

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

WebSocket Messages

- After a successful handshake, client and server switch to sending "messages"
- Messages are conceptual
- Ride on top of "frames"

WebSocket Frames

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/63)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

WebSocket Frames (2)

- Protocol defines 6 frame types
  - text frames
  - binary frames
  - continuation frames
  - ping
  - pong
  - connection close
- Reserves 10 additional frame types for future use

WebSocket Frames (3)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/63)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

WebSocket Frames (3)

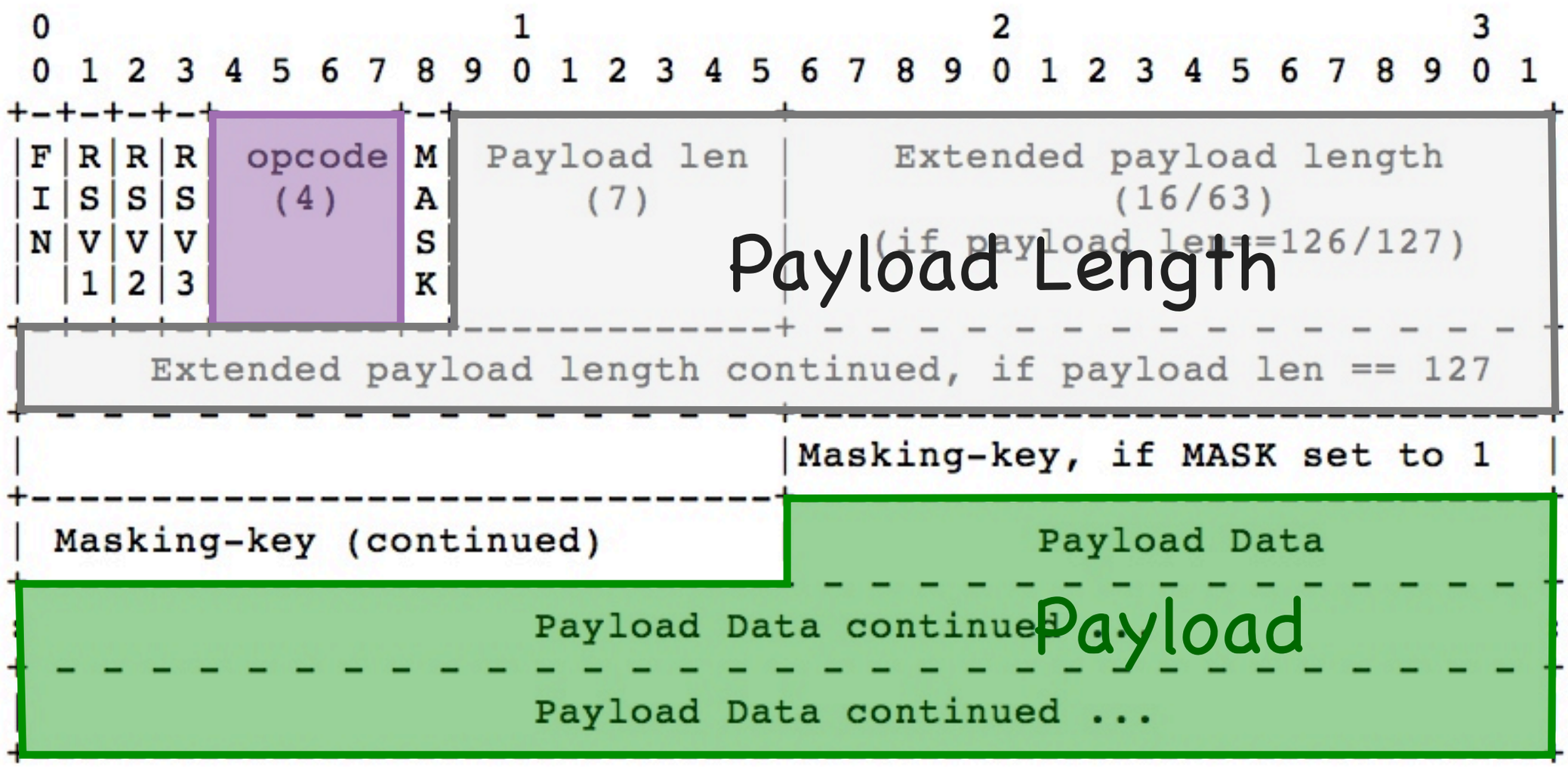
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/63)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

WebSocket Frames (3)

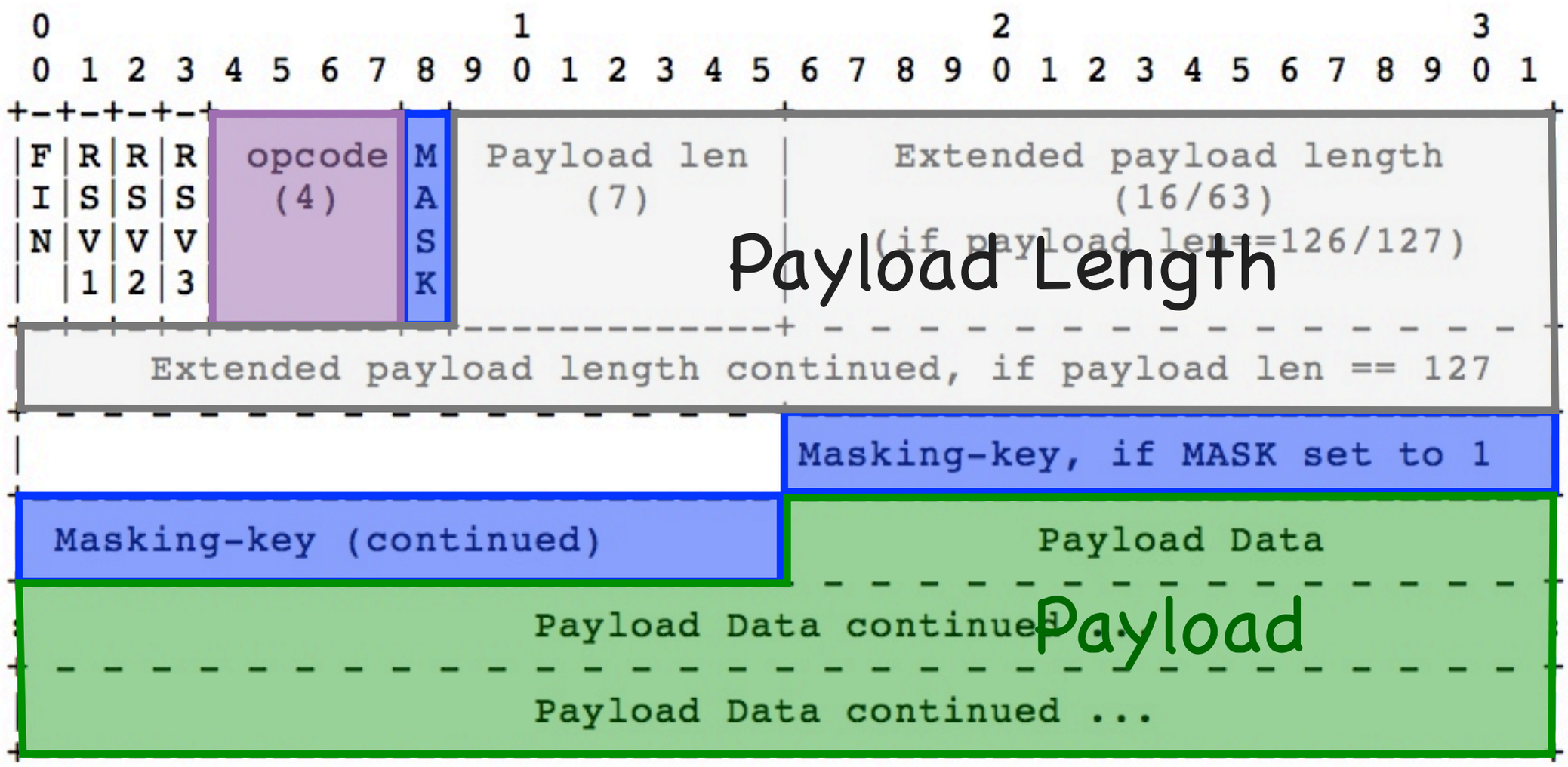
```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-------+-+-------------+-------------------------------+
|F|R|R|R| opcode|M| Payload len |    Extended payload length    |
|I|S|S|S|  (4)  |A|     (7)     |             (16/63)           |
|N|V|V|V|       |S|             |   (if payload len==126/127)   |
| |1|2|3|       |K|             |                               |
+-+-+-+-+-------+-+-------------+ - - - - - - - - - - - - - - - +
|     Extended payload length continued, if payload len == 127  |
+ - - - - - - - - - - - - - - - +-------------------------------+
|                               |Masking-key, if MASK set to 1  |
+-------------------------------+-------------------------------+
| Masking-key (continued)       |          Payload Data         |
+-------------------------------- - - - - - - - - - - - - - - - +
:                     Payload Data continued ...                :
+ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - +
|                     Payload Data continued ...                |
+---------------------------------------------------------------+
```

Payload Length

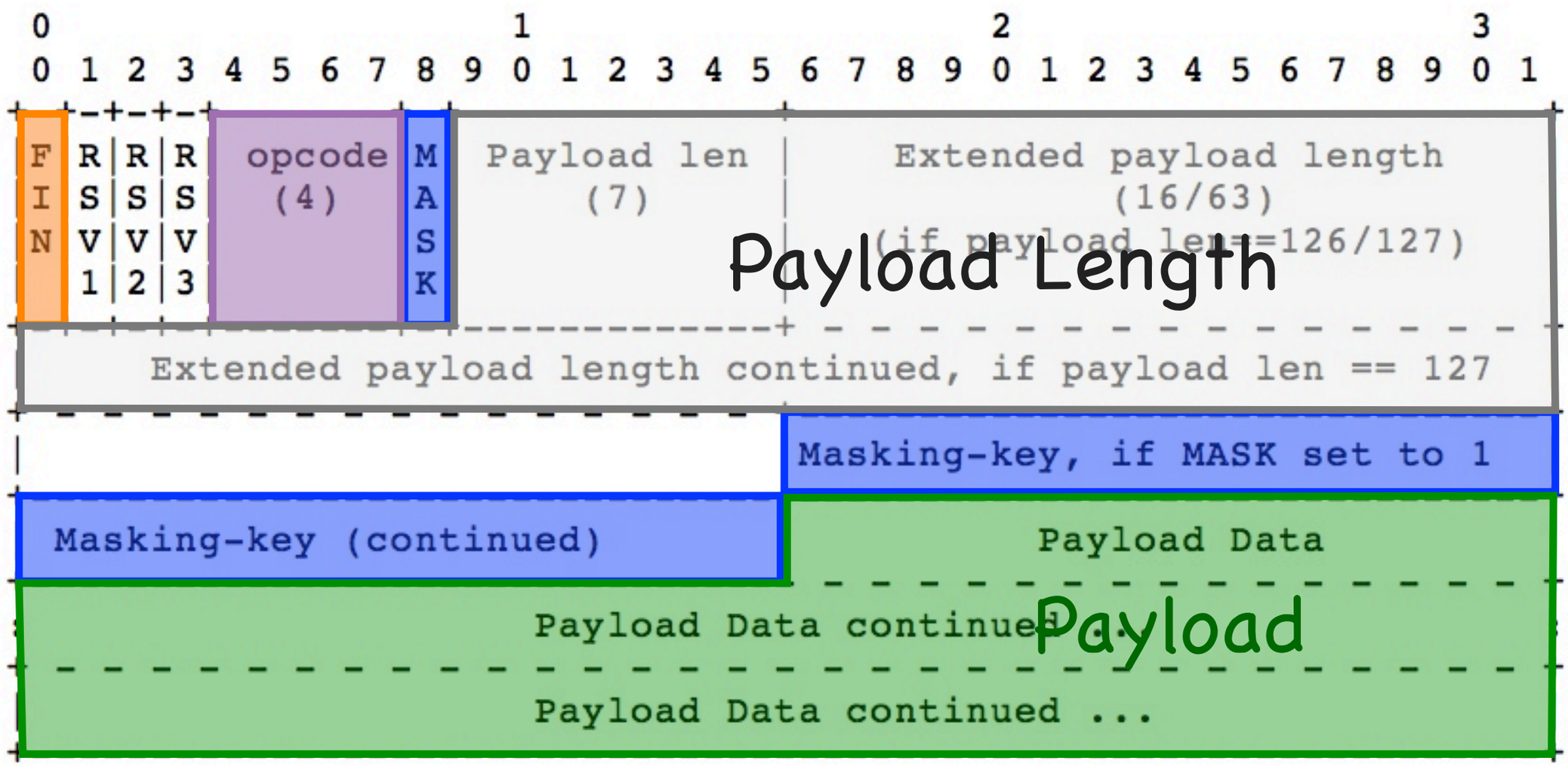
WebSocket Frames (3)



WebSocket Frames (3)



WebSocket Frames (3)



Supported Browsers

Browser support [edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

Supported Browsers

Browser support

[edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

Supported Browsers

Browser support

[edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

Supported Browsers

Browser support

[edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

Not Yet Ready For Prime-time Player?

Phew!

Phew!

- Okay, we've got WebSockets, sort of
- What about jWebSocket?

jWebSocket

- Client-side - JavaScript
- Server-side - Java
- Frameworks on top of both
  - Substantial feature set beyond WebSockets
- You can implement raw WebSockets

jWebSocket Browser Support

Latest update of this page: **2010-11-18** Please also refer to jWebSocket Cross-Browser-Compatibility. Currently the following browsers are supported by jWebSocket. Chrome 4.0.429, 5+, 6+, 7+, 8+ as well as Safari 5, Firefox 3.7a6, 4.0b and Opera 10.7, 11.0 already support native HTML5 WebSockets - all common other brothers compatible with FlashBridge.

Browser	Version	SSL wss://	Comment
Chrome	**8.0.x** **7.0.x** **6.0.x** **5.0.x** **4.0.249+**	**8.0.x** **7.0.x**	**Chrome includes native WebSockets since 4.0.249. SSL (wss) approved with Jetty 8.0.** Tested and verified by jWebSocket team 2010-02-28. Download latest Google Chrome here.
Safari	**5.x** **4.x**	-	**Safari 5.x includes native WebSockets** - tested and verified by jWebSocket team 2010-06-08. Safari 4.x does not yet provide the WebSocket class. Installing Adobe Flash Player is a reliable, quick and simple workaround - absolutely transparent to your application.
Firefox	**4.0b1+** **3.7a6** **3.0+**	-	**Firefox 3.7a6 and 4.0b1+ now include native WebSockets** - tested and verified by jWebSocket team 2010-07-14, including new Security features (Sec-WebSocket-Key header processing), Firefox update available since Nightly Build 0.9.0.0620. Firefox versions prior to 3.7a6 tested with FlashBridge and verified by jWebSocket team 2010-03-28. Installing Adobe Flash Player is a reliable, quick and simple workaround - absolutely transparent to your application. Also successfully tested on Ubuntu 10.04 and gentoo Linux with Firefox 3.6+.
Opera	**11.0b** **10.7b** **9+** **10+**	-	**Opera includes native WebSockets since 10.7.9067 (dev internal), 11.0 beta working as well!** Tested and verified by jWebSocket team 2010-10-18. Older versions successfully tested with FlashBridge. Download latest official Opera Browser here. latest dev internal Version here.
Internet Explorer with FlashBridge	**6/7/8**	-	Tested and verified by jWebSocket team 2010-03-31. Download latest Adobe Flash Player here
Internet Explorer with Chrome Frame 4/5/6/7	**6/7/8**	-	CF4 tested and verified by jWebSocket team 2010-02-04. Download Google Chrome Frame for Internet Explorer here (for IE 6/7/8 only). CF5 tested and verified by jWebSocket team 2010-06-09, including new Security features (Sec-WebSocket-Key header processing), available since Nightly Build 0.9.0.0609. Download Google Chrome Frame 5 (beta from Developer Channel) for Internet Explorer here (for IE 6/7/8 only).

jWebSocket Browser Support

Latest update of this page: **2010-11-18** Please also refer to jWebSocket Cross-Browser-Compatibility. Currently the following browsers are supported by jWebSocket. Chrome 4.0.429, 5+, 6+, 7+, 8+ as well as Safari 5, Firefox 3.7a6, 4.0b and Opera 10.7, 11.0 already support native HTML5 WebSockets - all common other brothers compatible with FlashBridge.

Browser	Version	SSL wss://	Comment
Chrome	**8.0.x** **7.0.x** **6.0.x** **5.0.x** **4.0.249+**	**8.0.x** **7.0.x**	**Chrome includes native WebSockets since 4.0.249. SSL (wss) approved with Jetty**
Safari	**5.x** **4.x**	-	
Firefox	**4.0b1+** **3.7a6** **3.0+**	-	
Opera	**11.0b** **10.7b** **9+** **10+**	-	
Internet Explorer with FlashBridge	**6/7/8**	-	
Internet Explorer with Chrome Frame 4/5/6/7	**6/7/8**	-	(Sec-WebSocket-Key header processing), available since Nightly Build 0.9.0.0609. Download Google Chrome Frame 5 (beta from Developer Channel) for Internet Explorer here (for IE 6/7/8 only).

Browser support [edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

jWebSocket Browser Support

Latest update of this page: **2010-11-18** Please also refer to jWebSocket Cross-Browser-Compatibility. Currently the following browsers are supported by jWebSocket. Chrome 4.0.429, 5+, 6+, 7+, 8+ as well as Safari 5, Firefox 3.7a6, 4.0b and Opera 10.7, 11.0 already support native HTML5 WebSockets - all common other brothers compatible with FlashBridge.

Browser	Version	SSL wss://	Comment
Chrome	**8.0.x** **7.0.x** **6.0.x** **5.0.x** **4.0.249+**	**8.0.x** **7.0.x**	**Chrome includes native WebSockets since 4.0.249. SSL (wss) approved with Jetty**
Safari	**5.x** **4.x**	-	
Firefox	**4.0b1+** **3.7a6** **3.0+**	-	
Opera	**11.0b** **10.7b** **9+** **10+**	-	
Internet Explorer with FlashBridge	**6/7/8**	-	
Internet Explorer with Chrome Frame 4/5/6/7	**6/7/8**	-	(Sec-WebSocket-Key header processing), available since Nightly Build 0.9.0.0609. Download Google Chrome Frame 5 (beta from Developer Channel) for Internet Explorer here (for IE 6/7/8 only).

Browser support

[edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

jWebSocket Browser Support

Latest update of this page: **2010-11-18** Please also refer to jWebSocket Cross-Browser-Compatibility. Currently the following browsers are supported by jWebSocket. Chrome 4.0.429, 5+, 6+, 7+, 8+ as well as Safari 5, Firefox 3.7a6, 4.0b and Opera 10.7, 11.0 already support native HTML5 WebSockets - all common other brothers compatible with FlashBridge.

Browser	Version	SSL wss://	Comment
Chrome	**8.0.x** **7.0.x** **6.0.x** **5.0.x** **4.0.249+**	**8.0.x** **7.0.x**	**Chrome includes native WebSockets since 4.0.249. SSL (wss) approved with Jetty**
Safari	**5.x** **4.x**	-	
Firefox	**4.0b1+** **3.7a6** **3.6**	-	
Opera	**11.0b** **10.70** **9+** **10**	-	
Internet Explorer with FlashBridge	**6/7/8**	-	
Internet Explorer with Chrome Frame 4/5/6/7	**6/7/8**	-	

jWebSocket FlashBridge?

jWebSocket FlashBridge?

(Sec-WebSocket-Key header processing), available since Nightly Build 0.9.0.0609. Download Google Chrome Frame 5 (beta from Developer Channel) for Internet Explorer here (for IE 6/7/8 only).

Browser support [edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

Microsoft supports the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

jWebSocket Browser Support

Latest update of this page: **2010-11-18** Please also refer to jWebSocket Cross-Browser-Compatibility. Currently the following browsers are supported by jWebSocket. Chrome 4.0.429, 5+, 6+, 7+, 8+ as well as Safari 5, Firefox 3.7a6, 4.0b and Opera 10.7, 11.0 already support native HTML5 WebSockets - all common other brothers compatible with FlashBridge.

Browser	Version	SSL wss://	Comment
Chrome	8.0.x 7.0.x 6.0.x 5.0.x 4.0.249+	8.0.x 7.0.x	**Chrome includes native WebSockets since 4.0.249. SSL (wss) appr**
Safari	5.x 4.x	-	
Firefox	4.0b1+ 3.7a6	-	
Opera	1		
Internet Explorer with Flas	6/7/8	-	
Internet Explorer with Chrome Frame 4/5/6/7	6/7/8	-	

jWebSocket FlashBridge?

jWebSocket FlashBridge?

Your Mileage May Vary :-)

Browser support [edit]

- Chrome 4
- Safari 5 (includes iOS 4.2)

The following browsers originally supported WebSocket, but have since disabled the protocol by default. Chrome also plans to disable the WebSocket if actual exploit code appears before the protocol is revised.[10]

- Firefox 4[11]
- Opera 11[12]

Currently, two browsers still support the outdated draft-ietf-hybi-thewebsocketprotocol-00. It has been disabled in Firefox and Opera due to security issues.

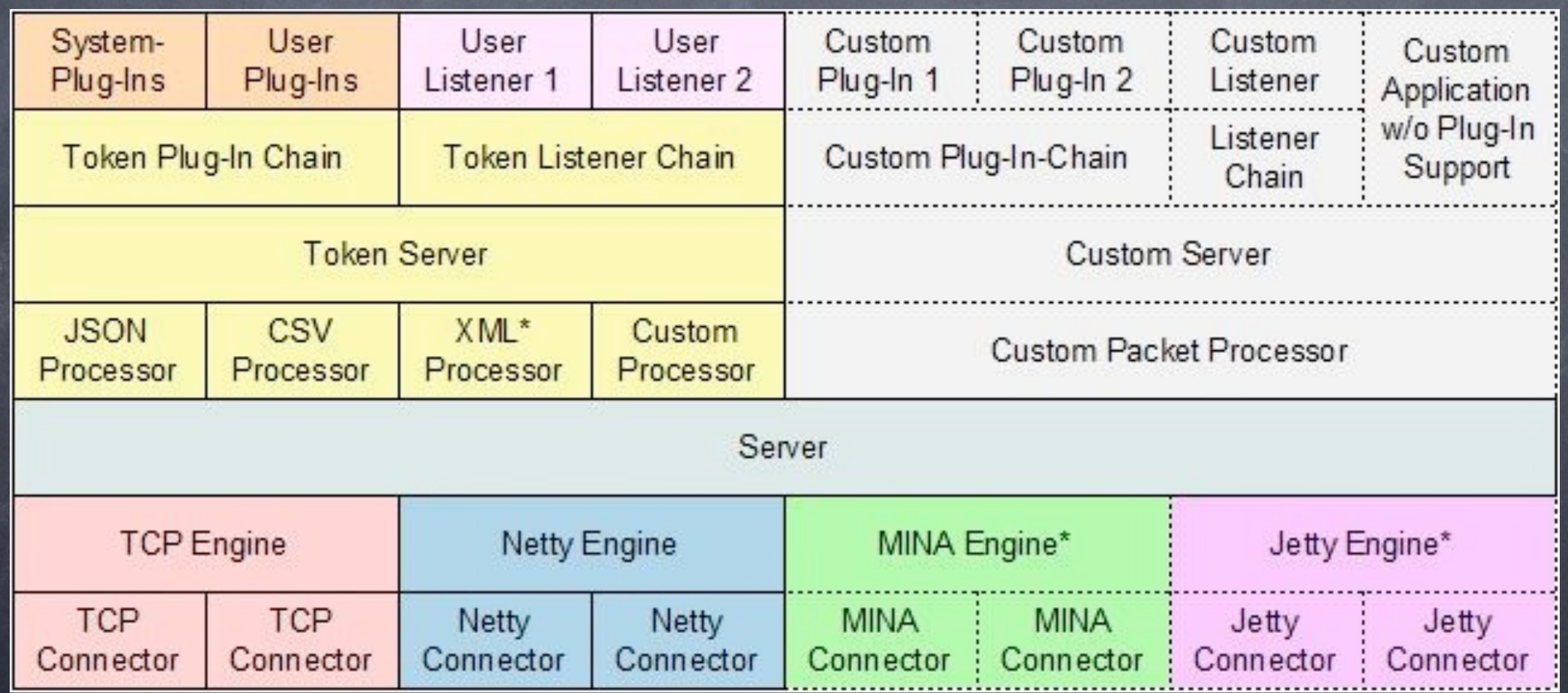
Microsoft implements the draft-ietf-hybi-thewebsocketprotocol-04 in Internet Explorer through a prototype HTML5 Labs.[13]

Implementation status

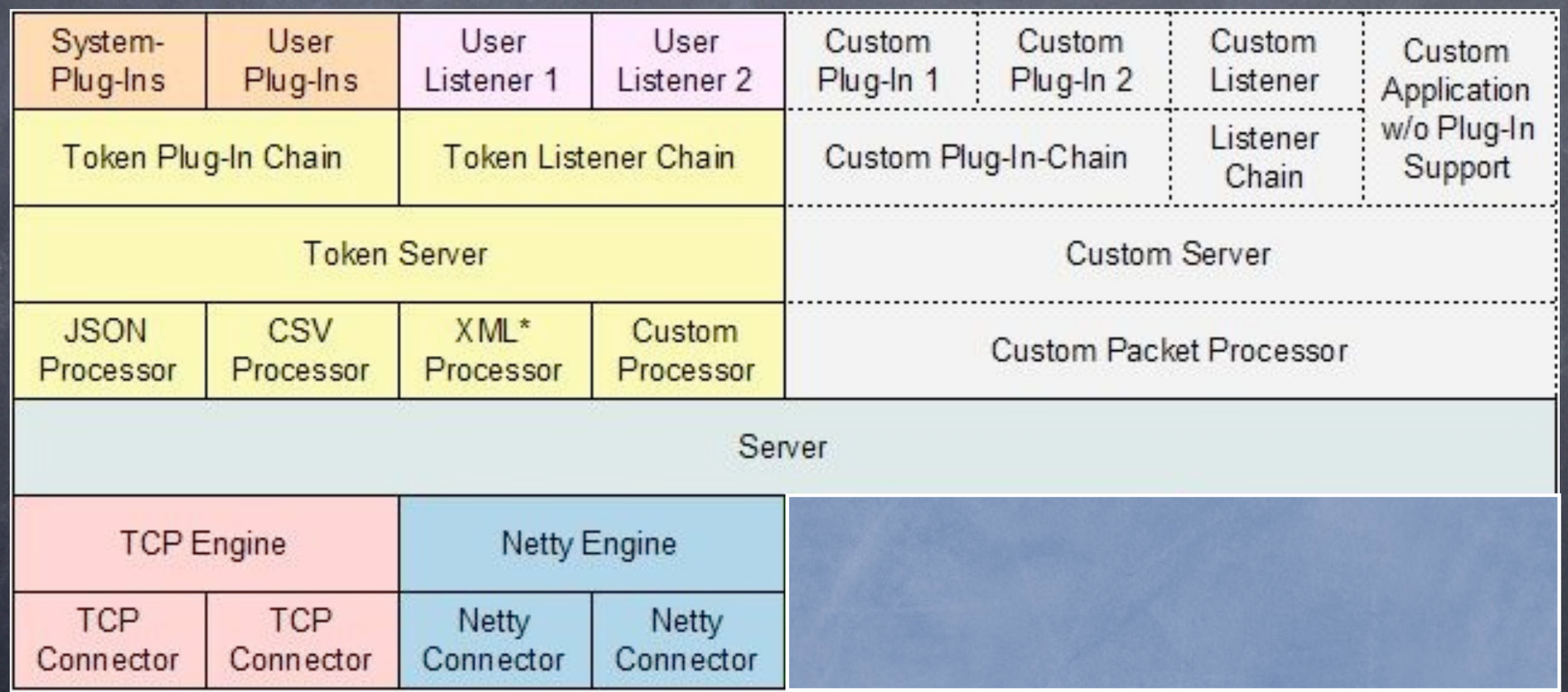
Protocol	Internet Explorer	Mozilla Firefox	Google Chrome	Safari	Opera	NetFront
hixie-75			4	5.0.0		
hixie-76 hybi-00		4.0 RC (DISABLED)	6	5.0.1	11.00 (DISABLED)	
hybi-06	HTML5 Labs[14]	dev[15]				
hybi-07		6.0 (Alpha)[16]				

(Sec-WebSocket-Key header processing), available since Nightly Build 0.9.0.0609. Download Google Chrome Frame 5 (beta from Developer Channel) for Internet Explorer here (for IE 6/7/8 only).

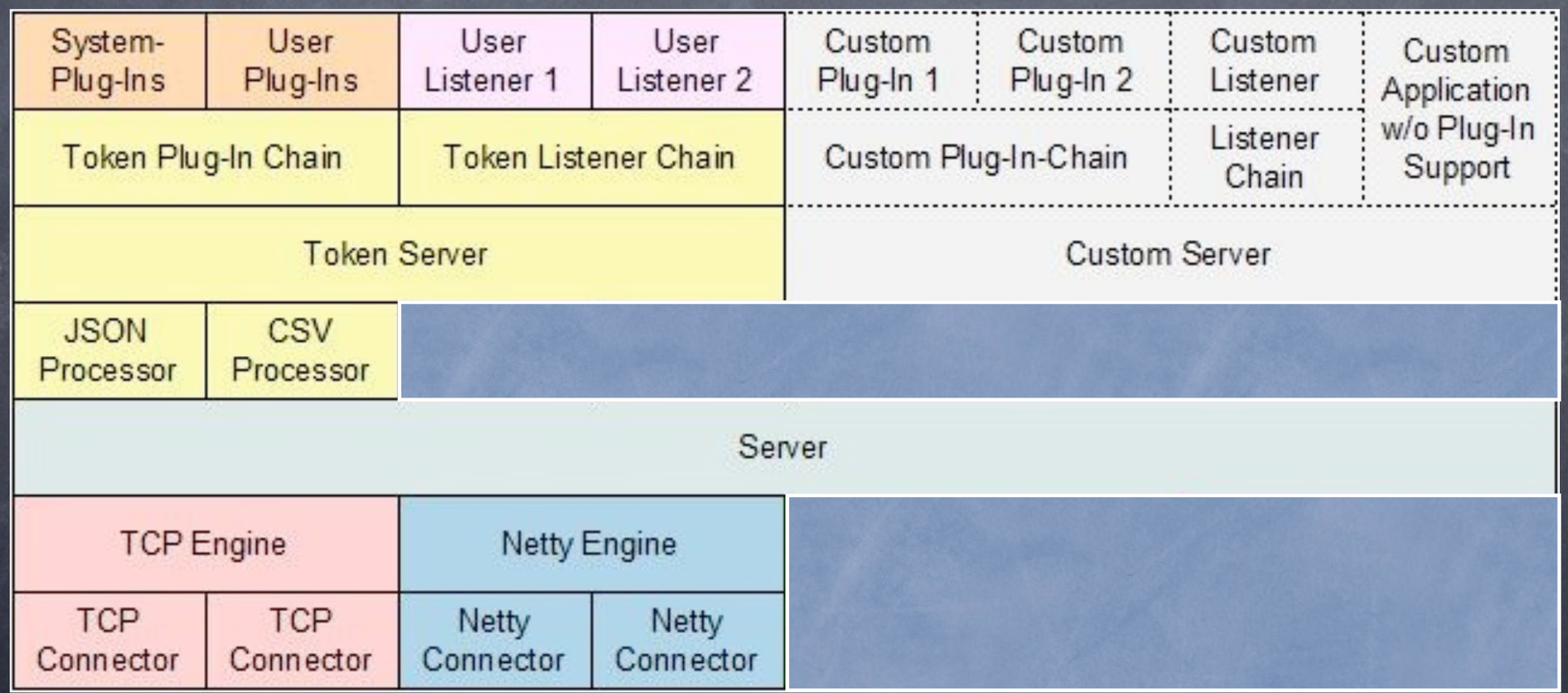
jWebSocket Server Architecture



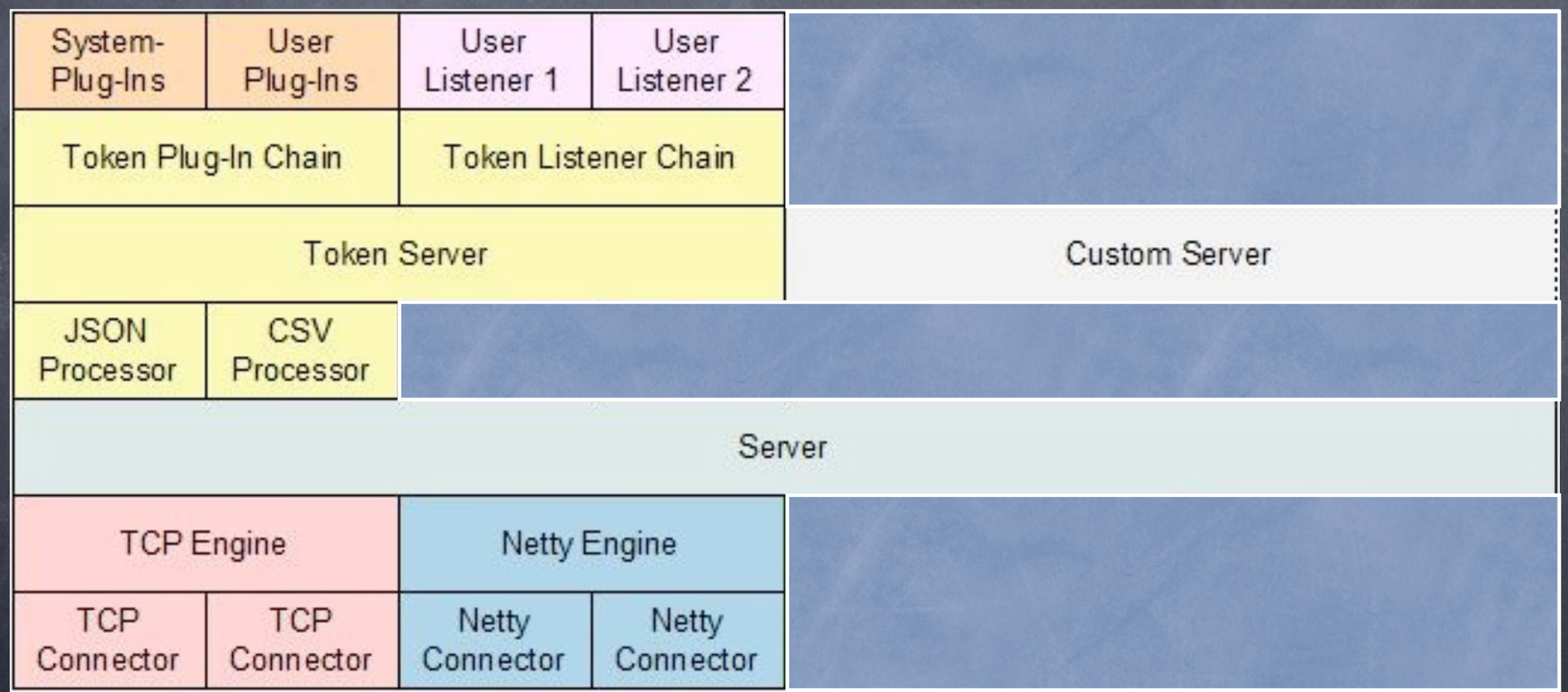
jWebSocket Server Architecture



jWebSocket Server Architecture



jWebSocket Server Architecture



jWebSocket Server Architecture (2)

jWebSocket Server Architecture (2)

- Engines for web containers

jWebSocket Server Architecture (2)

- Engines for web containers
  - currently TCP & Netty

jWebSocket Server Architecture (2)

- Engines for web containers
  - currently TCP & Netty
  - MINA and Jetty planned

jWebSocket Server Architecture (2)

- Engines for web containers
  - currently TCP & Netty
  - MINA and Jetty planned
- Plug-Ins - primary message processing model

jWebSocket Server Architecture (2)

- Engines for web containers
  - currently TCP & Netty
  - MINA and Jetty planned
- Plug-Ins - primary message processing model
- Streaming - chains of low-level client connection handlers (WebSocketConnector)

jWebSocket Server Architecture (2)

- Engines for web containers
  - currently TCP & Netty
  - MINA and Jetty planned
- Plug-Ins - primary message processing model
- Streaming - chains of low-level client connection handlers (WebSocketConnector)
- Tokens

jWebSocket Server Architecture (2)

- Engines for web containers
  - currently TCP & Netty
  - MINA and Jetty planned
- Plug-Ins - primary message processing model
- Streaming - chains of low-level client connection handlers (WebSocketConnector)
- Tokens
  - reconstituted information from a frame

jWebSocket Server Architecture (2)

- Engines for web containers
  - currently TCP & Netty
  - MINA and Jetty planned
- Plug-Ins - primary message processing model
- Streaming - chains of low-level client connection handlers (WebSocketConnector)
- Tokens
  - reconstituted information from a frame
  - JSON, CSV, XML

jWebSocket Server Architecture (3)

jWebSocket Server Architecture (3)

- Filters - low-level (not documented)

jWebSocket Server Architecture (3)

- Filters - low-level (not documented)
- Channels - implemented on top of plug-ins

jWebSocket Server Architecture (3)

- Filters - low-level (not documented)
- Channels - implemented on top of plug-ins
- Keep-Alive - avoid disconnects

WebSocket Opinions

- Absolutely needed
- Seems well-considered
- Security issues need to be sorted out

jWebSocket Opinions

- Core code seems to implement the draft WebSocket standard faithfully
- Has many features
- Layering and feature interactions could use better documentation
- Wish: Core server-side WebSocket support cleanly separated out
- Wish: Cleaner source code organization

Further Investigations?

- Jetty WebSocket Server
  - http://blogs.webtide.com/gregw/entry/jetty_websocket_server
- Resin Websockets
  - http://caucho.com/resin-4.0/examples/websocket-java/
- Webbit
  - https://github.com/joewalnes/webbit