



# HTML5 Realtime and WebSocket Code Lab



@peterlubbers  
@franksalim  
Kaazing

# WebSocket Demo



<http://demo.kaazing.com/racer/>



# WebSocket Demo

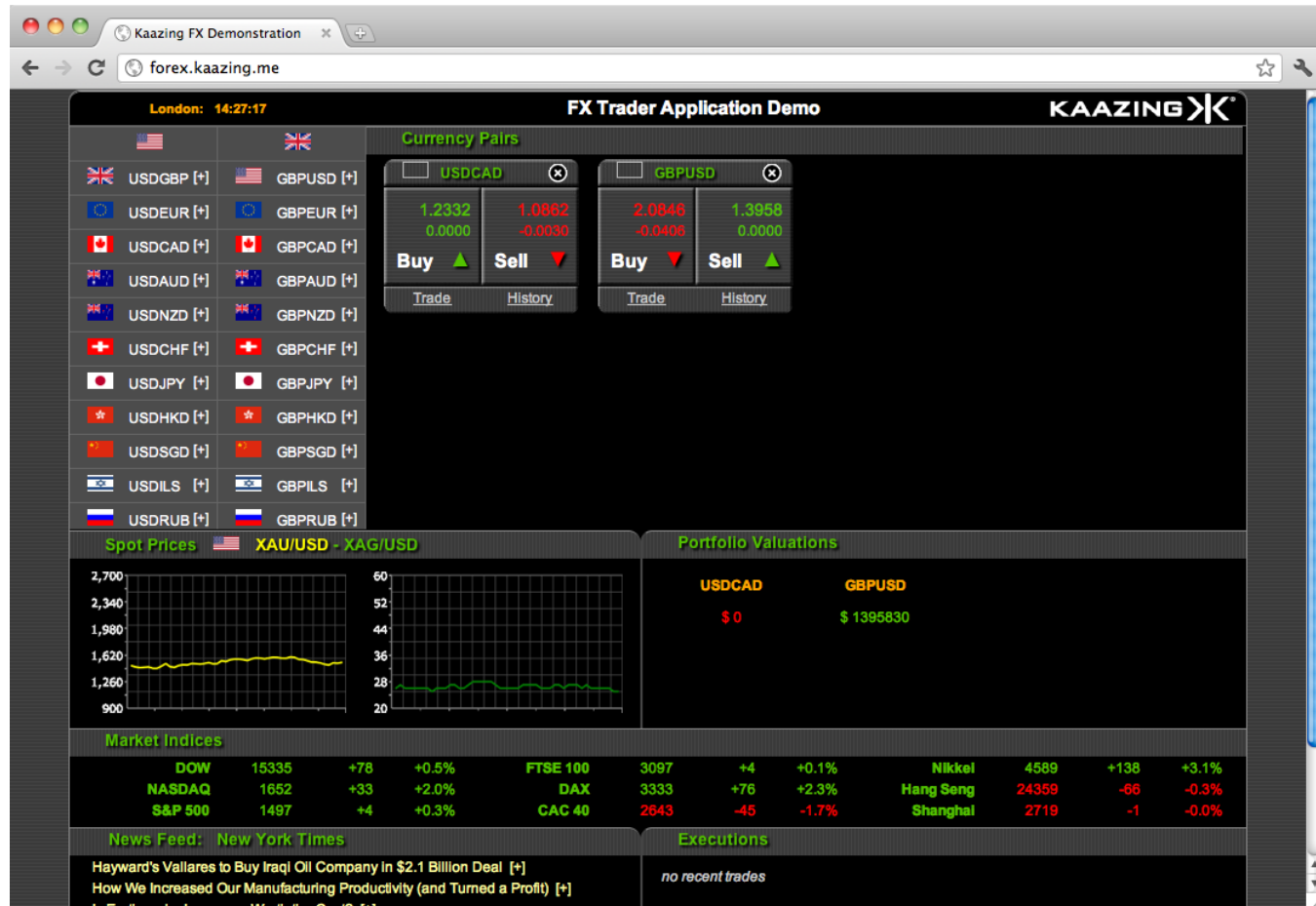
- Plink: <http://labs.dinahmoe.com/plink/#>



# WebSocket Demos

## FX Trader Application Demo

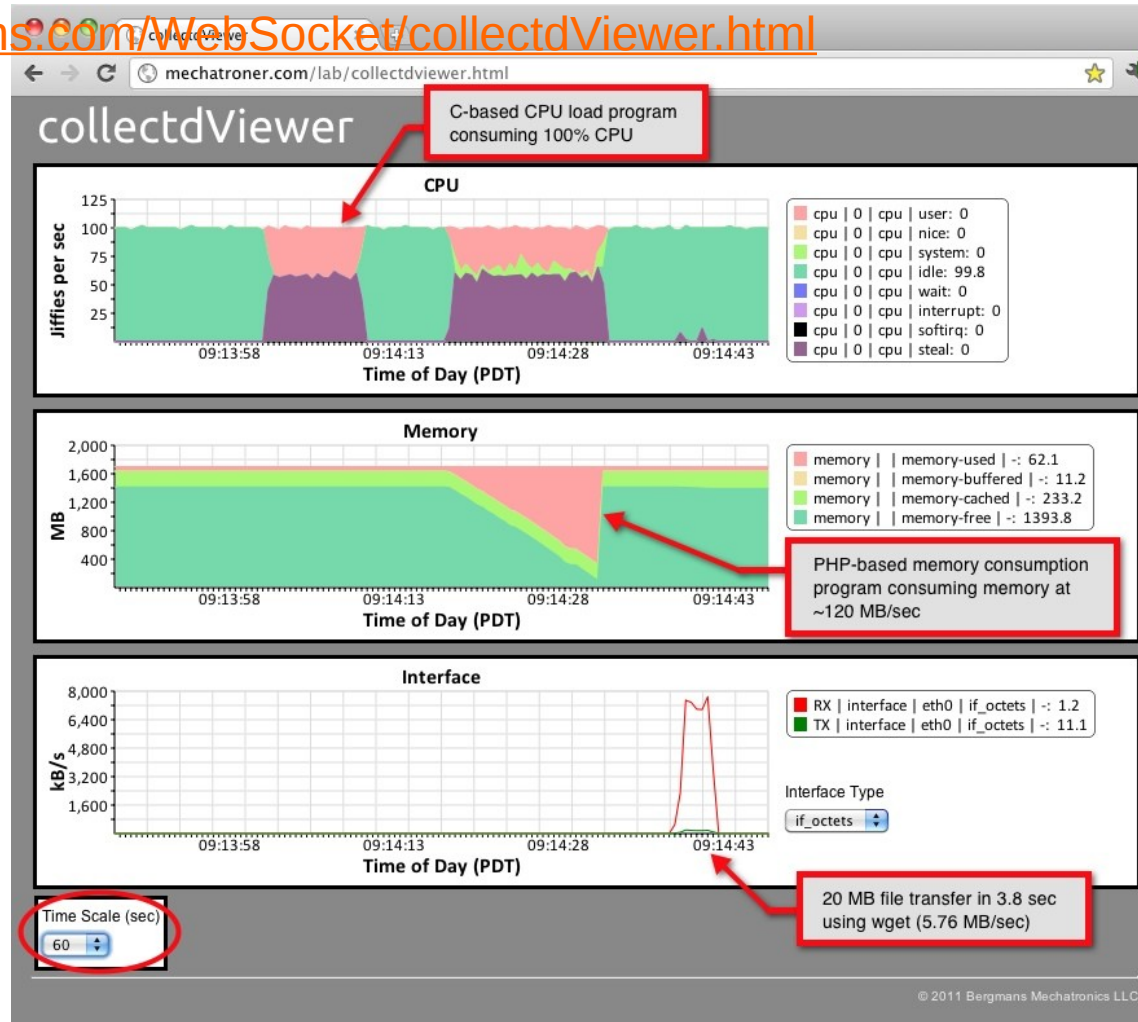
<http://demo.kaazing.me/forex>



# Example: CollectdViewer

## Server Monitor System

<http://bergmans.com/WebSocket/collectdViewer.html>



- Demos
- Introduction to WebSocket
- WebSocket API
- WebSocket Protocol
- Real-World WebSocket

# *Introduction to WebSocket*



- On the LAN: Reliable, real-time communication
- On the web: ?
  - Mostly idle
  - Mostly broadcast
  - *Nearly* real-time
- Web + WebSockets = reliable *and* real-time
- Approaches on the web:
  - WebSockets
  - Http-based



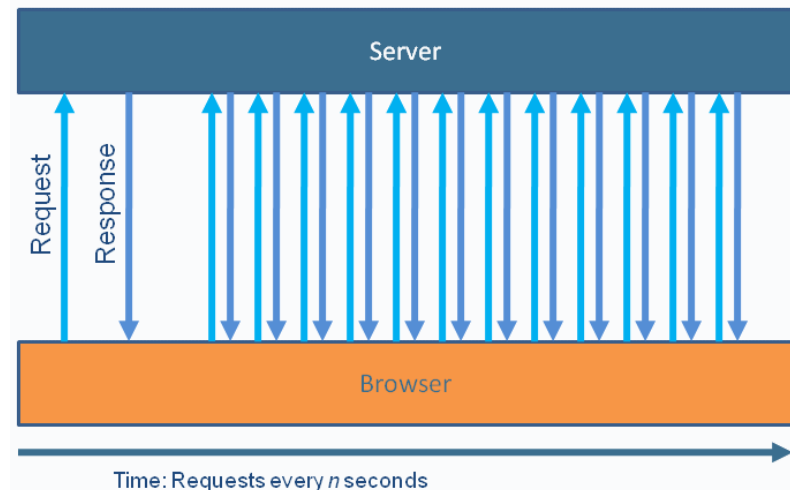
- HTTP is half-duplex
  - Traffic flows in only one direction at a time
  - Bidirectional communications are complicated to manage
- HTTP is stateless
  - Redundant information sent with each HTTP request and response



- AJAX (Asynchronous JavaScript + XML)
  - Content can change without loading the entire page
  - User-perceived low latency
- Comet
  - Technique for server push
  - Lack of a standard implementation
  - Comet adds lots of complexity



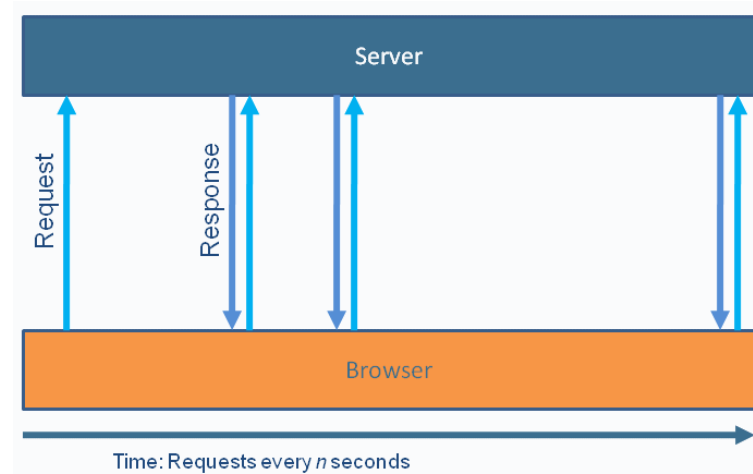
- Polling is "nearly real-time"
- Used in Ajax applications to simulate real-time communication
- Browser sends HTTP requests at regular intervals and immediately receives a response



# Long Polling

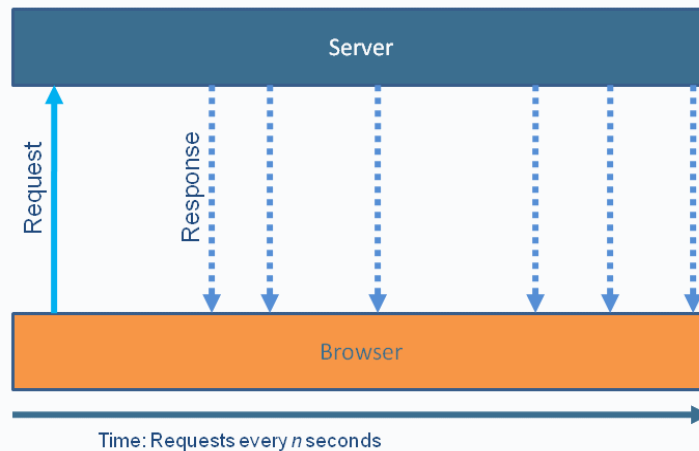
*a/k/a Asynchronous polling*

- Browser sends a request to the server, server keeps the request open for a set period
- Speed limited by response-request-response
- Request/response headers add overhead on the wire





- More efficient, but sometimes problematic
- Possible complications:
  - Proxies and firewalls
  - Response builds up and must be flushed periodically
  - Cross-domain issues to do with browser connection limits



# Comet Polling Example

The screenshot shows a Mozilla Firefox browser window titled "Lab 1: Fake-Time Stock". The address bar displays "http://localhost:8080/PollingStock/". The page content is a table of stock market data. Below the table, the browser's developer tools are open, showing the "Console" tab with a list of "GET PollingStock" requests to "localhost:8080". A red arrow points to the console with the text "Network Hammering!".

Company	Ticker	Price	Change	% Change
International Business Machines Corp.	IBM	86.76	0.33	0.4
Citigroup, Inc.	C	48.73	0.55	1.1
Boeing Co.	BA	64.08	1.21	1.9
Microsoft Corporation	MSFT	31.11	-0.47	-1.5
AT&T Inc.	T	29.36	0.11	0.4
Wal Mart Stores Inc.	WMT	36.51	-0.61	-1.6
Intel Corporation	INTC	20.46	0.17	0.8
Verizon Communications Inc.	VZ	35.00	0.32	0.9
Hewlett-Packard Co.	HPQ	33.27	0.00	0.0
3M Co.	MMM	72.57	0.00	0.0
McDonald's Corporation	MCD	41.45	-0.66	-1.6

Console Log:

- GET PollingStock 200 OK localhost:8080 320 B 93ms
- GET PollingStock 200 OK localhost:8080 320 B 9ms
- GET PollingStock 200 OK localhost:8080 320 B 87ms
- GET PollingStock 200 OK localhost:8080 320 B 92ms
- GET PollingStock 200 OK localhost:8080 320 B 91ms
- GET PollingStock 200 OK localhost:8080 320 B 87ms
- GET PollingStock 200 OK localhost:8080 320 B 96ms
- GET PollingStock 200 OK localhost:8080 320 B 12ms
- GET PollingStock 200 OK localhost:8080 320 B 10ms
- GET PollingStock 200 OK localhost:8080 320 B 11ms
- GET PollingStock 200 OK localhost:8080 320 B 4ms
- GET PollingStock 200 OK localhost:8080 320 B 38ms
- GET PollingStock 200 OK localhost:8080 320 B 117ms

Response:

MM:66.05,AT&T Inc.:T:29.49,International Business Machines Corp.:IBM:86.29,Citigroup, Inc.:C:48.73,McDonald's Corporation:MCD:41.42,Wal Mart Stores Inc.:WMT:40.70,Intel Corporation:INTC:19.36,Boeing Co.:BA:72.04,Microsoft Corporation:MSFT:29.46,Verizon Communications Inc.:VZ:35.62,Hewlett-Packard Co.:HPQ:33.30

# HTTP Request Headers

## Client

```
GET /PollingStock//PollingStock HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
rv:1.9.1.5) Gecko/20091102 Firefox/3.5.5
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Referer: http://localhost:8080/PollingStock/
Cookie: showInheritedConstant=false;
showInheritedProtectedConstant=false; showInheritedProperty=false;
showInheritedProtectedProperty=false; showInheritedMethod=false;
showInheritedProtectedMethod=false; showInheritedEvent=false;
showInheritedStyle=false; showInheritedEffect=false;
```

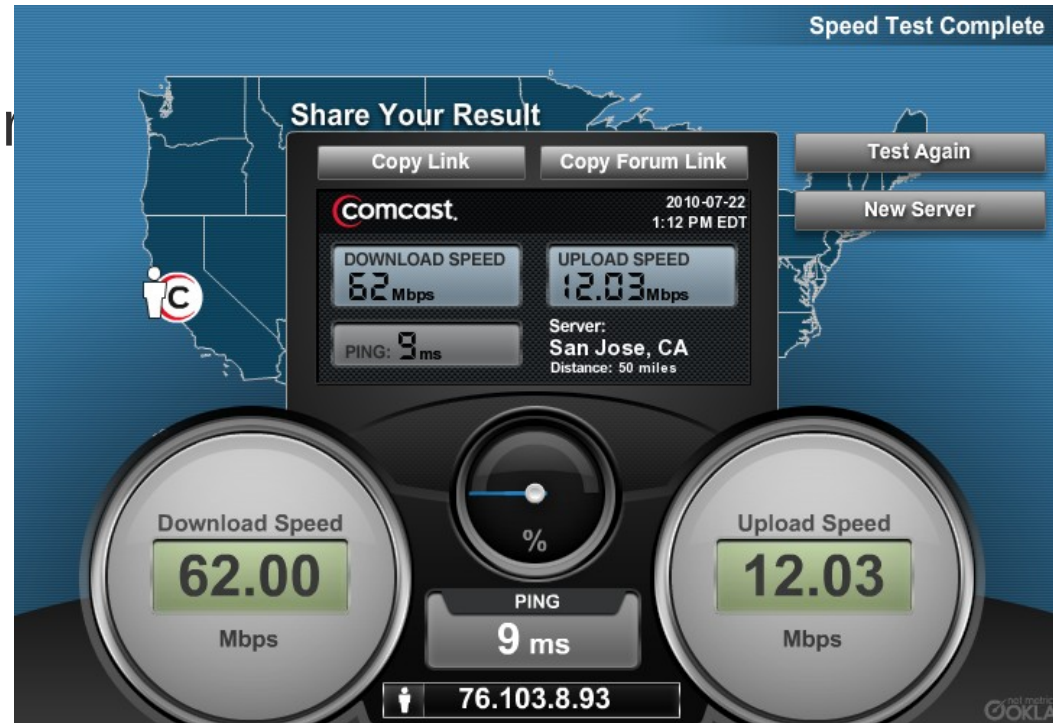
## Server

```
HTTP/1.x 200 OK
X-Powered-By: Servlet/2.5
Server: Sun Java System Application Server 9.1_02
Content-Type: text/html;charset=UTF-8
Content-Length: 321
Date: Sat, 07 Nov 2009 00:32:46 GMT
```

- Total overhead: 871 bytes (example)
- Often 2K+ bytes
  - e.g. cookies



- Most users have Internet connections where upload to download ratios are between 1:4 and 1:20
  - Result: 500 byte HTTP request header request could take as long to upload as 10 kB of HTTP response data takes to download



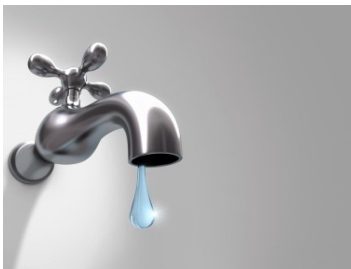
# HTTP Header Traffic Analysis

Client	Overhead Bytes	Overhead Mbps
1,000	871,000	~6,6*
10,000	8,710,000	~66
100,000	87,100,000	~665

\* 871,000 bytes = 6,968,000 bits = ~6.6 Mbps

*"Reducing kilobytes of data to 2 bytes...and reducing latency from 150ms to 50ms is far more than marginal. In fact, these two factors alone are enough to make WebSocket seriously interesting to Google."*

—Ian Hickson (Google, HTML5 spec lead)



- 
- Kazuo Whiteboard 2008
- The diagram illustrates a network protocol sequence between a client (C) and a server (S) via a proxy (K). The sequence of messages is as follows:
- Client (C) to Proxy (K):**
    - TCP open
    - create
    - created
    - connect
    - connected
    - frame
    - ack
    - close
  - Proxy (K) to Server (S):**
    - create
    - created
    - connect
    - connected
    - frame
    - ack
    - close
  - Server (S) to Proxy (K):**
    - create
    - created
    - connect
    - connected
    - frame
    - ack
    - close
  - Proxy (K) to Client (C):**
    - create
    - created
    - connect
    - connected
    - frame
    - ack
    - close
- A red circle highlights the 'K' box and the 'Web Socket' label. The diagram is on a whiteboard with a 'Kazuo Whiteboard 2008' label in the top left corner.



- Cross-web communications w/ remote host
  - Full-duplex (bi-directional), single socket
  - Shares port with existing HTTP content
  - Traverses firewalls and proxies
  - ws:// and wss://
- W3C API (Javascript)
- IETF Protocol

# USING THE WEBSOCKET API

## JavaScript

```
var status = document.getElementById("support");  
if (window.WebSocket) { // or Modernizr.websocket  
    status.innerHTML = "HTML5 WebSocket is supported";  
} else {  
    status.innerHTML = "HTML5 WebSocket is not supported";  
}
```

# Using the WebSocket API

## JavaScript

```
//Create new WebSocket
var mySocket = new WebSocket("ws://www.WebSocket.org");

// Associate listeners
mySocket.onopen = function(evt) {
};
mySocket.onclose = function(evt) {
    alert("closed w/ status: " + evt.code);
};
mySocket.onmessage = function(evt) {
    alert("Received message: " + evt.data);
};
mySocket.onerror = function(evt) {
    alert("Error");
};
```



# Using the WebSocket API

## JavaScript

```
// Sending data  
mySocket.send("WebSocket Rocks!");  
  
// Close WebSocket  
mySocket.close();
```

Available ?	<code>window.WebSocket</code> or <code>Modernizr.websocket</code>
Events	<code>onopen</code> , <code>onerror</code> , <code>onmessage</code>
Functions	<code>send</code> , <code>close</code>
Attributes	<code>url</code> , <code>readyState</code> , <code>bufferedAmount</code> , <i>extensions</i> , <i>protocol</i>

<http://dev.w3.org/html5/websockets/>

*Italics: -08 and later*

## Native:

- Chrome 4+
- Safari 5+
- Firefox 4+
- Opera 10.7+
- Internet Explorer 10+

## Emulation:

- Kaazing WebSocket Gateway
- socket.io
- SockJS
- web-socket.js (Flash)

<http://caniuse.com/#search=WebSocket>



# WebSocket Servers Libraries

- Kaazing
- Socket.io (node.js)
- Apache-websocket
- Cramp
- Nowjs
- SockJS
- SuperWebSocket
- Jetty
- Atmosphere
- APE Project
- Xsockets
- Orbited
- Atmosphere
- Autobahn
- CouchDB
- Netty
- Misultin
- Cowboy
- YAWS
- Juggernaut
- PHP Websocket
- websocketify
- ActiveMQ
- HornetMQ
- phpwebsocket
- Protocol::WebSocket
- em-websocket
- Jwebsocket
- WaterSprout Server
- Pywebsocket
- And more...
- **Client Libraries**
- Web-socket-js (JavaScript)
- AS3 WebSocket (ActionScript)
- .NET WebSocket client (.NET)
- Anaida (.NET)
- WebSocket Sharp (.NET)
- Silverlight WebSocket client
- Java WebSocket Client
- Arduino C++ WebSocket client
- Ruby-web-socket
- ZTWebSocket (Objective-C)
- Libwebsockets (C)

- Kaazing WebSocket Gateway
  - <http://www.kaazing.com/download>
  - Makes WebSocket work in all browsers today (including I.E. 6)
- Flash WebSocket implementation
  - <http://github.com/gimite/web-socket-js>
  - Requires opening port on the server's firewall
- Socket.io
  - <http://socket.io>
  - Alternate API
  - Adds heartbeat, timeouts and disconnection
  - Uses Flash when available



# THE WEBSOCKET PROTOCOL

“draft-hixie-thewebsocketprotocol-xx” IETF Network Working Group

Version	Date	Details
-00	Jan. 9 2009	• Initial version
-52	Oct. 23 2009	• Subprotocol concept introduced
-76	May 6 2010	• Used in older browsers (FF4, etc.)

“draft-ietf-hybi-thewebsocketprotocol-xx” (IETF HyBi Working Group)

Version	Date	Details
-01	Aug. 31 2010	• Added binary format
-04	Jan. 11 2011	• Introduced data masking to address proxy server security issue • Introduced including protocol version number in handshake
-14	Sep. 8 2011	• Guidance on version number negotiation
RFC 6455	Dec. 2011	• Final version <a href="http://tools.ietf.org/html/rfc6455">http://tools.ietf.org/html/rfc6455</a>

# WebSocket Protocol History

Client wants  
*ws://example.com/chat*



Client

Server accepts



Server

## REQUIRED

**GET** /chat HTTP/1.1  
**Host:** *example.com*  
**Upgrade:** websocket  
**Connection:** Upgrade  
**Sec-WebSocket-Key:** 16-byte nonce, base64 encoded  
**Sec-WebSocket-Version:** 13

## OPTIONAL

**Origin:** *http://example.com*  
**Sec-WebSocket-Protocol:** *protocol [,protocol]\**  
**Sec-WebSocket-Extensions:** *extension [,extension]\**  
**Cookie:** *cookie content & other cookie-related headers*

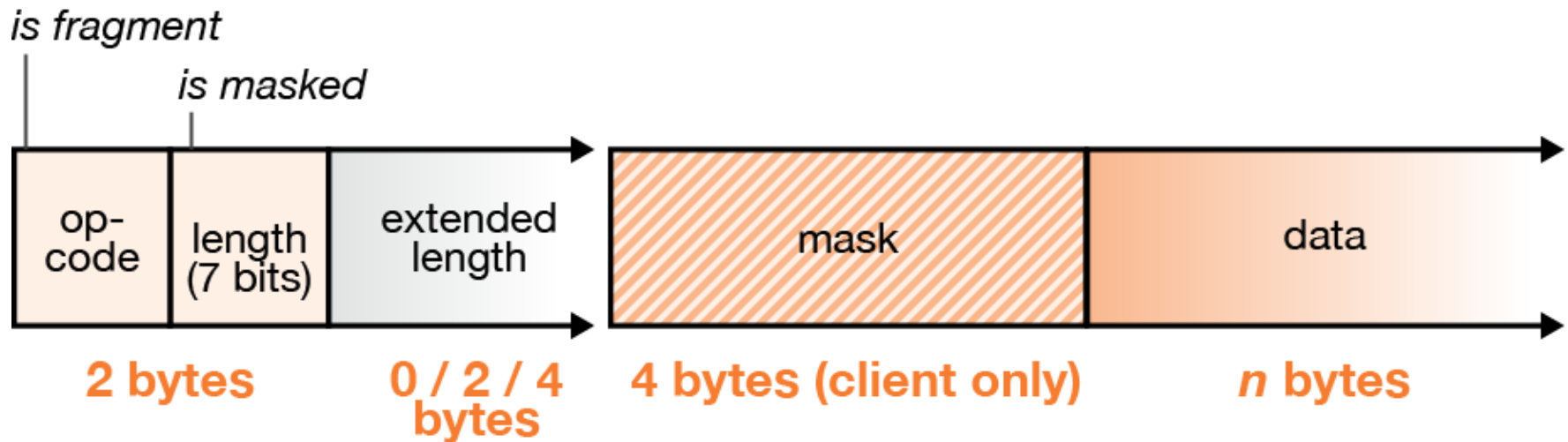
## REQUIRED

**HTTP/1.1** 101 "Switching Protocols" or other description  
**Upgrade:** websocket  
**Connection:** Upgrade  
**Sec-WebSocket-Accept:** 20-byte MD5 hash in base64

## OPTIONAL

**Sec-WebSocket-Protocol:** *protocol*  
**Sec-WebSocket-Extensions:** *extension [,extension]\**

# WebSocket Handshake



# WebSocket Frames

- Have a few header bytes

- Text or binary data

- Frames are masked from client to server

en1 - Wireshark

Filter: `ip.addr == 184.73.102.149` Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
38	23.676601	184.73.102.149	10.0.0.49	TCP	teradataordbms > 55516 [PSH, ACK] Seq=1 Ack=205 Win=6912 Len=34 TSV=29615904
39	23.676726	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [ACK] Seq=205 Ack=35 Win=524280 Len=0 TSV=51134816 TS
40	23.704288	184.73.102.149	10.0.0.49	TCP	teradataordbms > 55516 [PSH, ACK] Seq=35 Ack=205 Win=6912 Len=94 TSV=2961590
41	23.704425	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [ACK] Seq=205 Ack=129 Win=524280 Len=0 TSV=51134817 T
51	34.370630	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [PSH, ACK] Seq=205 Ack=129 Win=524280 Len=11 TSV=5113
52	34.427061	184.73.102.149	10.0.0.49	TCP	teradataordbms > 55516 [PSH, ACK] Seq=129 Ack=216 Win=6912 Len=7 TSV=2961601
53	34.427189	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [ACK] Seq=216 Ack=136 Win=524280 Len=0 TSV=51134924 T

Frame 51: 77 bytes on wire (616 bits), 77 bytes captured (616 bits)

Ethernet II, Src: Apple\_Of:13:00 (e0:f8:47:0f:13:00), Dst: Nomadix\_01:55:d3 (00:50:e8:01:55:d3)

Internet Protocol, Src: 10.0.0.49 (10.0.0.49), Dst: 184.73.102.149 (184.73.102.149)

Transmission Control Protocol, Src Port: 55516 (55516), Dst Port: teradataordbms (8002), Seq: 205, Ack: 129, Len: 11

Source port: 55516 (55516)

Destination port: teradataordbms (8002)

[Stream index: 2]

Sequence number: 205 (relative sequence number)

[Next sequence number: 216 (relative sequence number)]

Acknowledgement number: 129 (relative ack number)

Header length: 32 bytes

Flags: 0x18 (PSH, ACK)

Window size: 524280 (scaled)

Checksum: 0xb31a [validation disabled]

Options: (12 bytes)

[SEQ/ACK analysis]

Data (11 bytes)

Data: 8185cb6e9f8e830bf3e2a4

[Length: 11]

0000 00 50 e8 01 55 d3 e0 f8 47 0f 13 00 08 00 45 00 .P..U... G....E.

0010 00 3f f9 a5 40 00 00 06 18 04 0a 00 00 31 b8 49 .?..@. ....l.I

0020 66 95 d8 dc 1f 42 06 a1 b6 09 90 c7 ef b3 80 18 f....B. ....

0030 ff ff b3 1a 00 00 01 01 08 0a 03 0c 41 cb 11 a7 .....A...

0040 07 4b 81 85 cb 6e 9f 8e 83 0b f3 e2 a4 .K...n. ....

FIN bit, MASK bit, RSV bits, payload length, Op-Code

mask

payload

81 85 CB 6E 9F 8E 83 0B F3 E2 A4

83 0B F3 E2 A4

XOR CB 6E 9F 8E CB

-- -- -- -- --

48 65 6C 6C 6F

H e l l o

File: "/var/folders/xZ/xZZ+..." Packets: 53 Displayed: 15 Marked: 0 Dropped: 0 Profile: Default

# WebSocket Frames

## Wireshark Trace of WebSocket Basics Lab Message (Server to Client)

## Wireshark Trace of WebSocket Basics Lab Message (Client to Server)

Filter: `ip.addr == 184.73.102.149` Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
38	23.676601	184.73.102.149	10.0.0.49	TCP	teradataordbms > 55516 [PSH, ACK] Seq=1 Ack=205 Win=6912 Len=34 TSV=29615904
39	23.676726	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [ACK] Seq=205 Ack=35 Win=524280 Len=0 TSV=51134816 TS
40	23.704288	184.73.102.149	10.0.0.49	TCP	teradataordbms > 55516 [PSH, ACK] Seq=35 Ack=205 Win=6912 Len=94 TSV=2961590
41	23.704425	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [ACK] Seq=205 Ack=129 Win=524280 Len=0 TSV=51134817 T
51	34.370630	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [PSH, ACK] Seq=205 Ack=129 Win=524280 Len=11 TSV=5113
52	34.427061	184.73.102.149	10.0.0.49	TCP	teradataordbms > 55516 [PSH, ACK] Seq=129 Ack=216 Win=6912 Len=7 TSV=2961601
53	34.427189	10.0.0.49	184.73.102.149	TCP	55516 > teradataordbms [ACK] Seq=216 Ack=136 Win=524280 Len=0 TSV=51134924 T

Frame 52: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)

Ethernet II, Src: Nomadix\_01:55:d3 (00:50:e8:01:55:d3), Dst: Apple\_Of:13:00 (e0:f8:47:0f:13:00)

Internet Protocol, Src: 184.73.102.149 (184.73.102.149), Dst: 10.0.0.49 (10.0.0.49)

Transmission Control Protocol, Src Port: teradataordbms (8002), Dst Port: 55516 (55516), Seq: 129, Ack: 216, Len: 7

Source port: teradataordbms (8002)  
Destination port: 55516 (55516)  
[Stream index: 2]  
Sequence number: 129 (relative sequence number)  
[Next sequence number: 136 (relative sequence number)]  
Acknowledgement number: 216 (relative ack number)  
Header length: 32 bytes

Flags: 0x18 (PSH, ACK)  
Window size: 6912 (scaled)  
Checksum: 0x1112 [validation disabled]  
Options: (12 bytes)  
[SEQ/ACK analysis]

Data (7 bytes)  
Data: 810548656c6c6f  
[Length: 7]

0000 e0 f8 47 0f 13 00 00 50 e8 01 55 d3 08 00 45 20 ..G....P ..U...E  
0010 00 3b e8 7e 40 00 2f 06 3a 0f b8 49 66 95 0a 00 ;.-@./..If...  
0020 00 31 1f 42 d8 dc 90 c7 ef b3 06 a1 b6 14 80 18 .1.B... ..  
0030 00 6c 11 12 00 00 01 01 08 0a 11 a7 0b 7b 03 0c .l.....{..  
0040 41 cb 81 05 48 65 6c 6f 6f A...Hell o

FIN bit, RSV bits, Op-Code    MASK bit, payload length    payload

81	05	48	65	6C	6C	6F	
			H	e	l	l	o

File: "/var/folders/xZ/xZ7+... Packets: 53 Displayed: 15 Marked: 0 Dropped: 0 Profile: Default



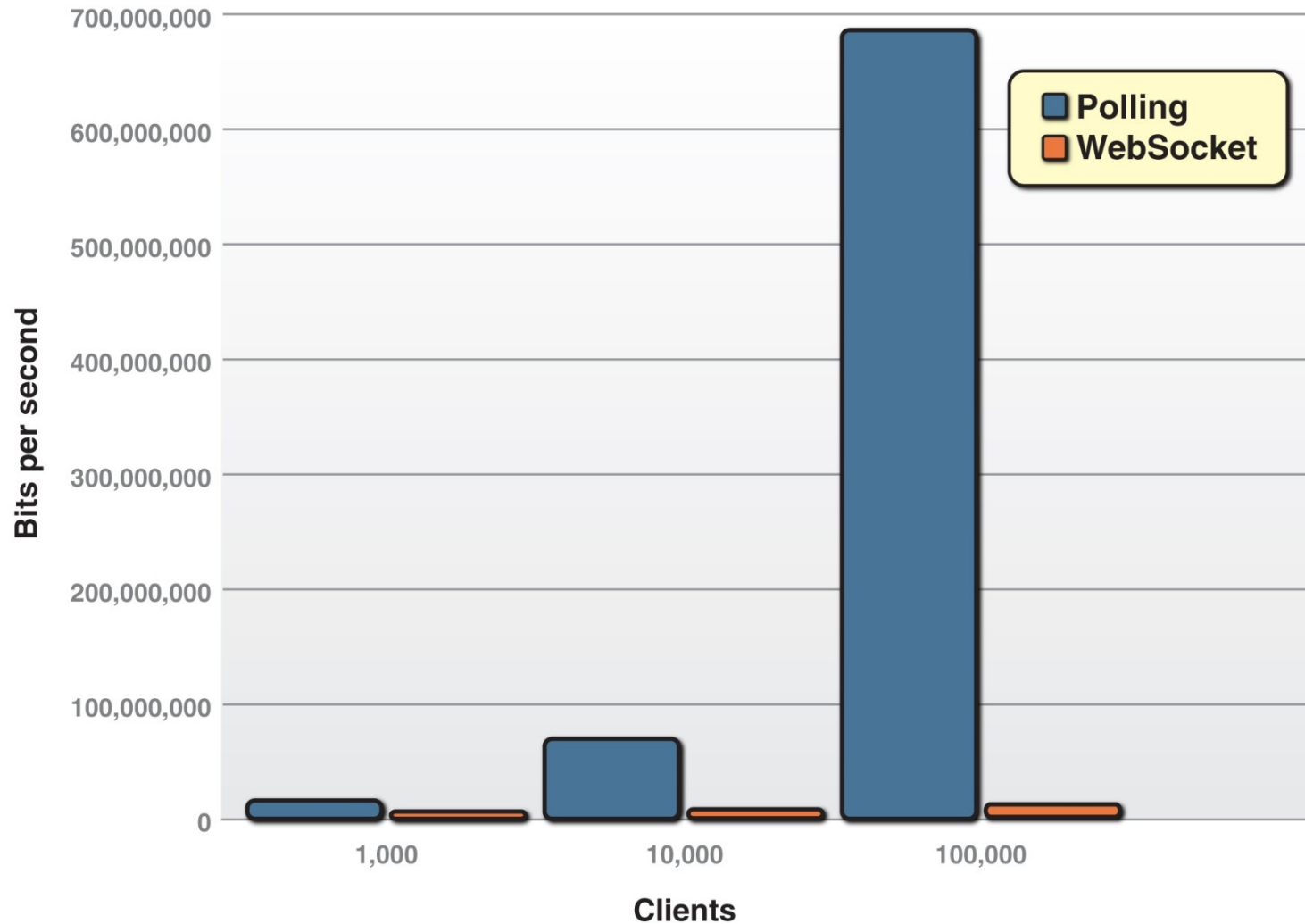
	HTTP	WebSocket
Overhead	100s of bytes	2-6 bytes (typical)
Latency	New connection each time	None: Use existing connection
Latency (polling)	Wait for next interval	No waiting
Latency (long polling)	None, if request sent earlier + time to set up next request	No waiting

# WebSocket Framing Analysis

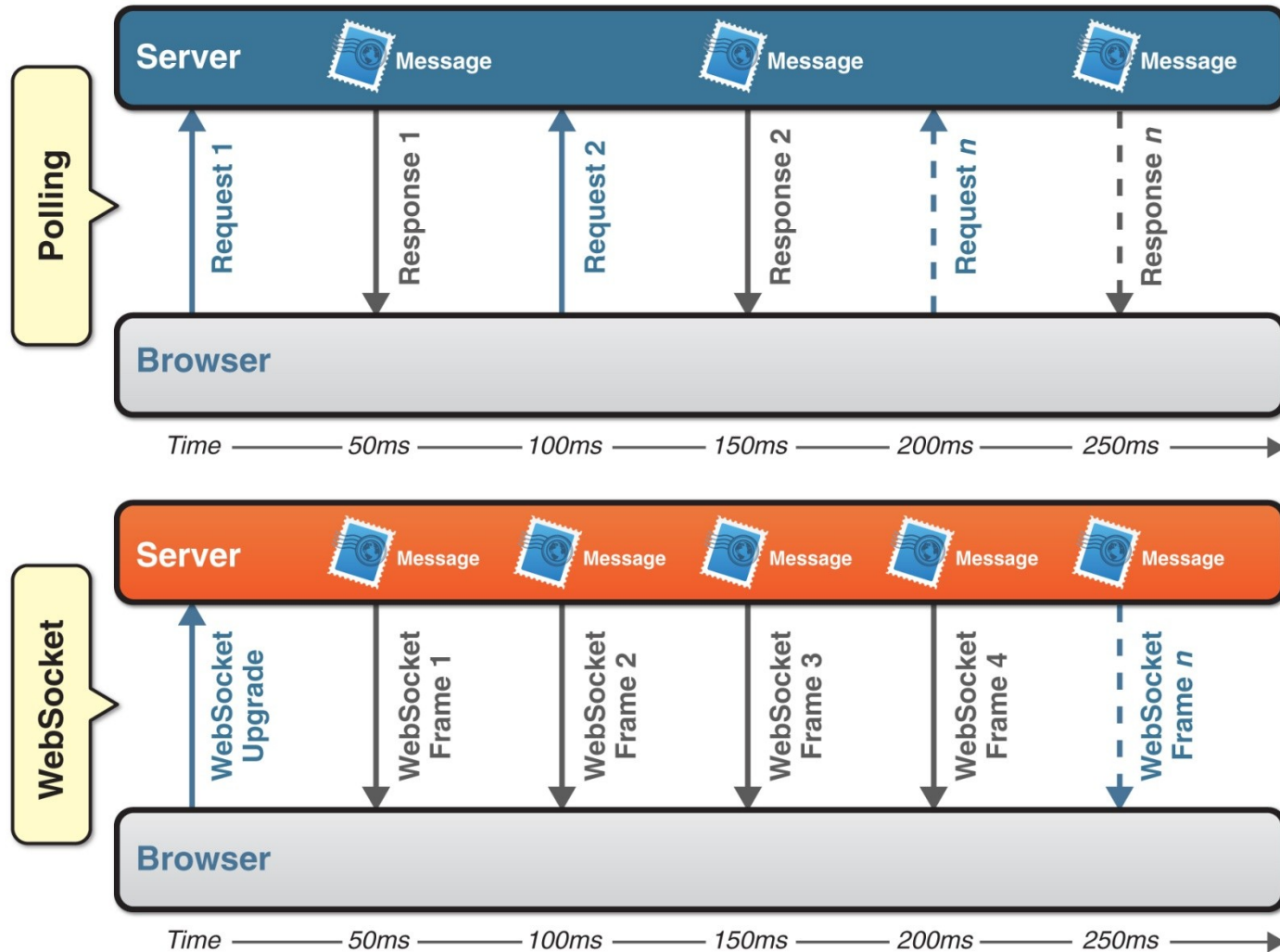
Client	Overhead Bytes	Overhead Mbps
1,000	2,000	~0.015*
10,000	20,000	~0.153
100,000	200,000	~1.526

\* 2,000 bytes = 16,000 bits (~0.015 Mbps)

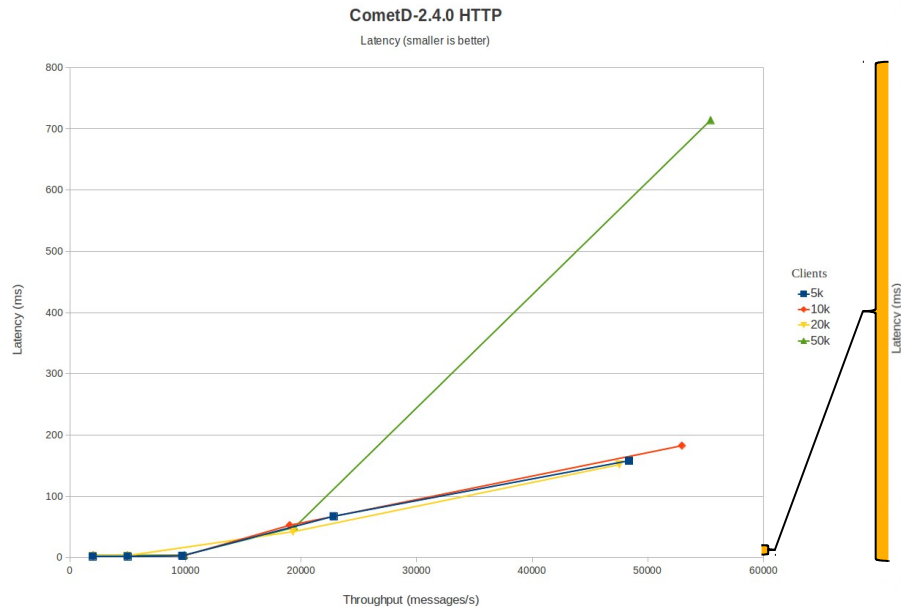
# Polling vs. WebSocket



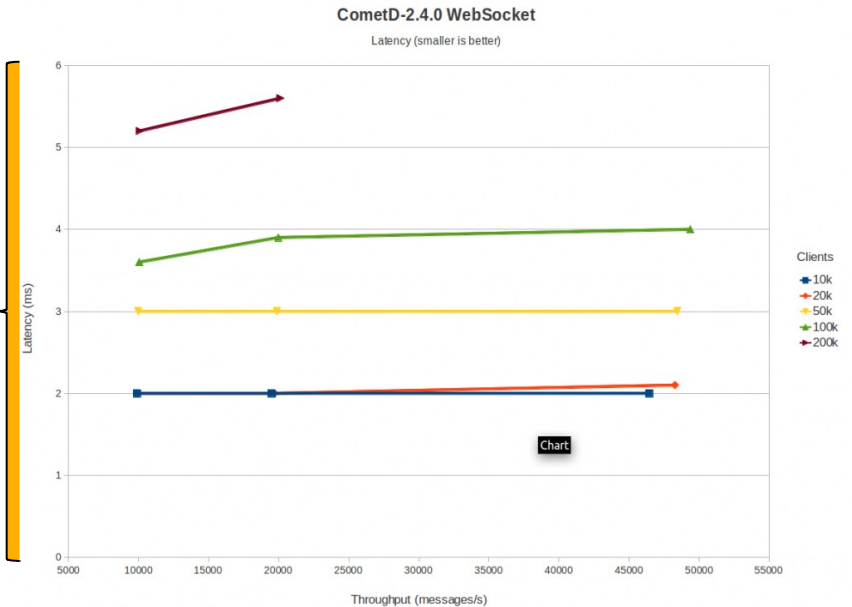
# Latency Reduction



## Using Comet



## Using WebSocket



<http://webtide.intalio.com/2011/09/cometd-2-4-0-websocket-benchmarks/>

- Extends network applications across the web
  - Desktop apps
  - Browser-based apps
  - Mobile apps
- *Far* more efficient than HTTP
- Part of the HTML5 Standard
- Older browsers can play too



- Extends legacy systems to the web
  - Message brokers, databases, etc.
- Extends client-server protocols to the web:
  - XMPP, Jabber
  - Pub/Sub (Stomp/AMQP)
  - Gaming protocols
  - Any TCP-based protocol
  - RFB/VNC
- Your browser becomes a first-class network citizen

- Better responsiveness
- Better scalability
  - Less traffic on the wire
  - Less work for the server
- Easier back-end development
  - Custom commands = better match to your needs
- Easier migration of existing systems
  - Just a new UI

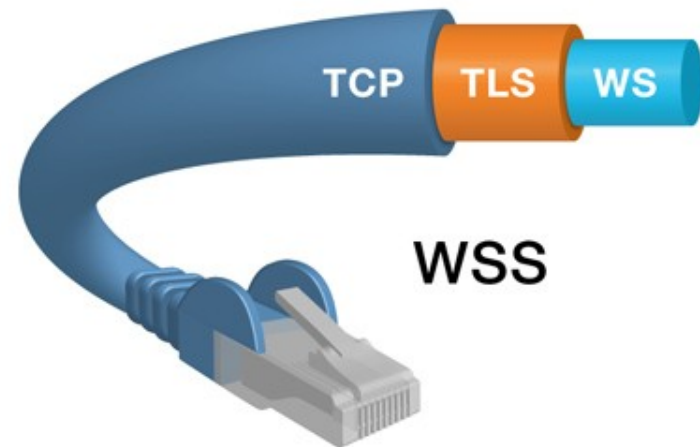
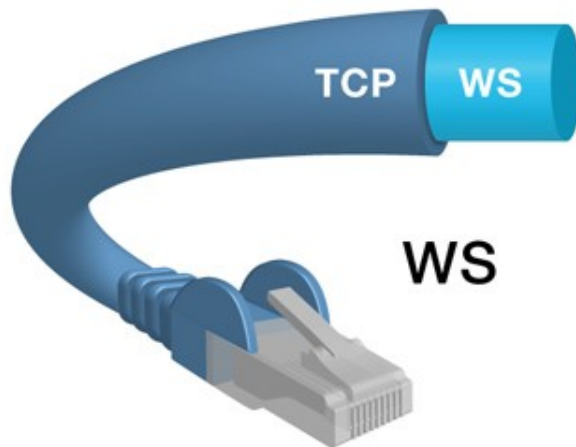
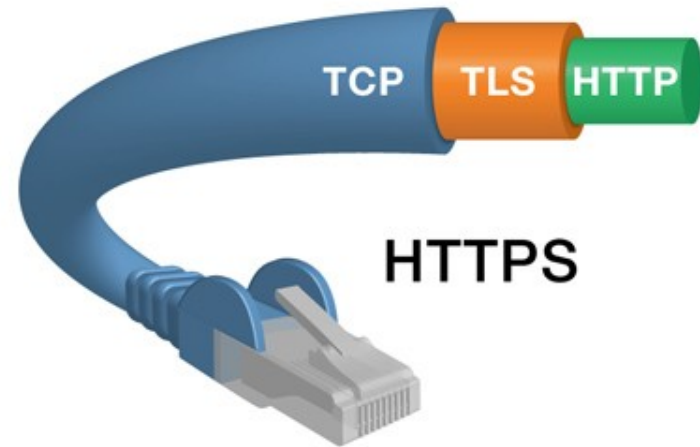
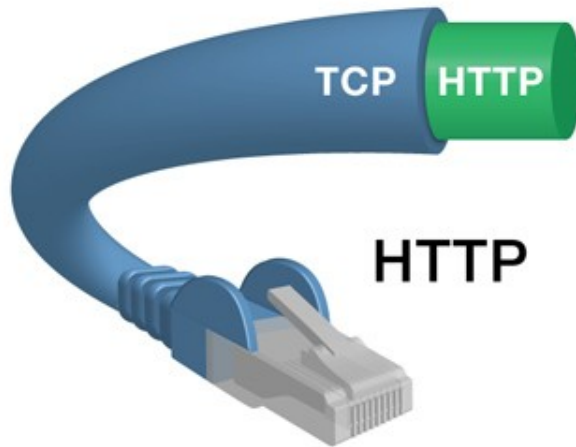
# *WEBSOCKETS IN THE REAL WORLD*

- Components of secure communication :
  1. Transport Layer
  2. Authentication
  3. Authorization
  4. Origin-Based Security and CORS

# Transport Layer Security (TLS)

- Also known as SSL (Secure Socket Layer) support
- HTTP over TLS is called HTTPS
  - Default port is 443
  - HTTPS is not a separate protocol
  - An HTTPS connection is established after a successful TLS handshake (using public and private key certificates)

# ws:// and wss:// schemes





- Mechanism by which systems identify users and check whether users really are who they represent themselves to be
- Authentication process
  - Step 1) Server issues a challenge using the HTTP 401 Authorization Required code
  - Step 2) Client responds by providing the requested authentication information if it can

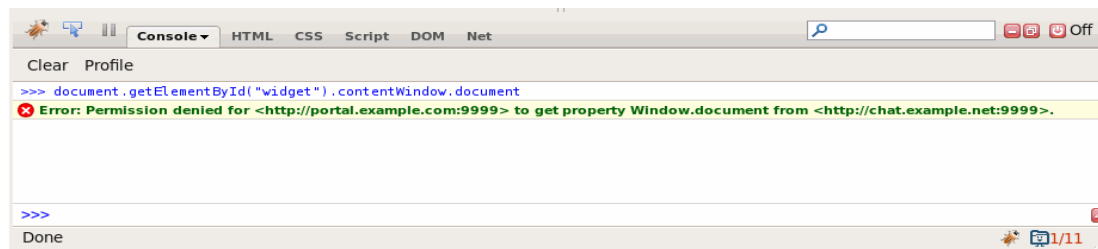
- Mechanism by which a system determines users' level of access
  - For example, a web page can have viewer, moderator, and administrator privileges
- Access rights are typically stored in the policy store that is associated with the application

- Web Origin Concept RFC 6454:  
<http://www.ietf.org/rfc/rfc6454.txt>
- An Origin is a subset of an address used for modeling trust relationships on the web
- Origins consist of a scheme, a host, and a port:
  - **Scheme:** http:, https:, ws:, wss:
  - **Host:** www.example.com, img.example.com, 192.0.2.10
  - **Port:** 80, 443

*“Generally speaking, documents retrieved from distinct origins are isolated from each other” – W3C*

[http://www.w3.org/Security/wiki/Same\\_Origin\\_Policy](http://www.w3.org/Security/wiki/Same_Origin_Policy)

- Browsers prevent a script or document loaded from one origin from communicating with a document loaded from another origin
- Original security model for HTML
  - Introduced in Netscape Navigator 2.0
  - Too limiting in this age of client-side web apps



Which URLs have the same Origin?

1. `http://www.example.com/index.html`
2. `https://www.example.com/index.html`
3. `http://img.example.com/html5.png`
4. `http://www.example.com:8080/page2.html`
5. `http://192.0.2.10:80/index.html*`
6. `http://www.example.com/about.html`

\* Where `192.0.2.10` resolves to `www.example.com`

## enable cross-origin resource sharing

### What is this about?

Cross-Origin Resource Sharing (CORS) is a [specification](#) that enables a truly open access across domain-boundaries. With this site we want to support the adoption of CORS. [\[more...\]](#)

If you have **public content** that doesn't use require cookie or session based authentication to see, then please consider opening it up for universal JavaScript/browser access. [\[more...\]](#)

### Why is CORS important?

[enable-cors.org](http://enable-cors.org)

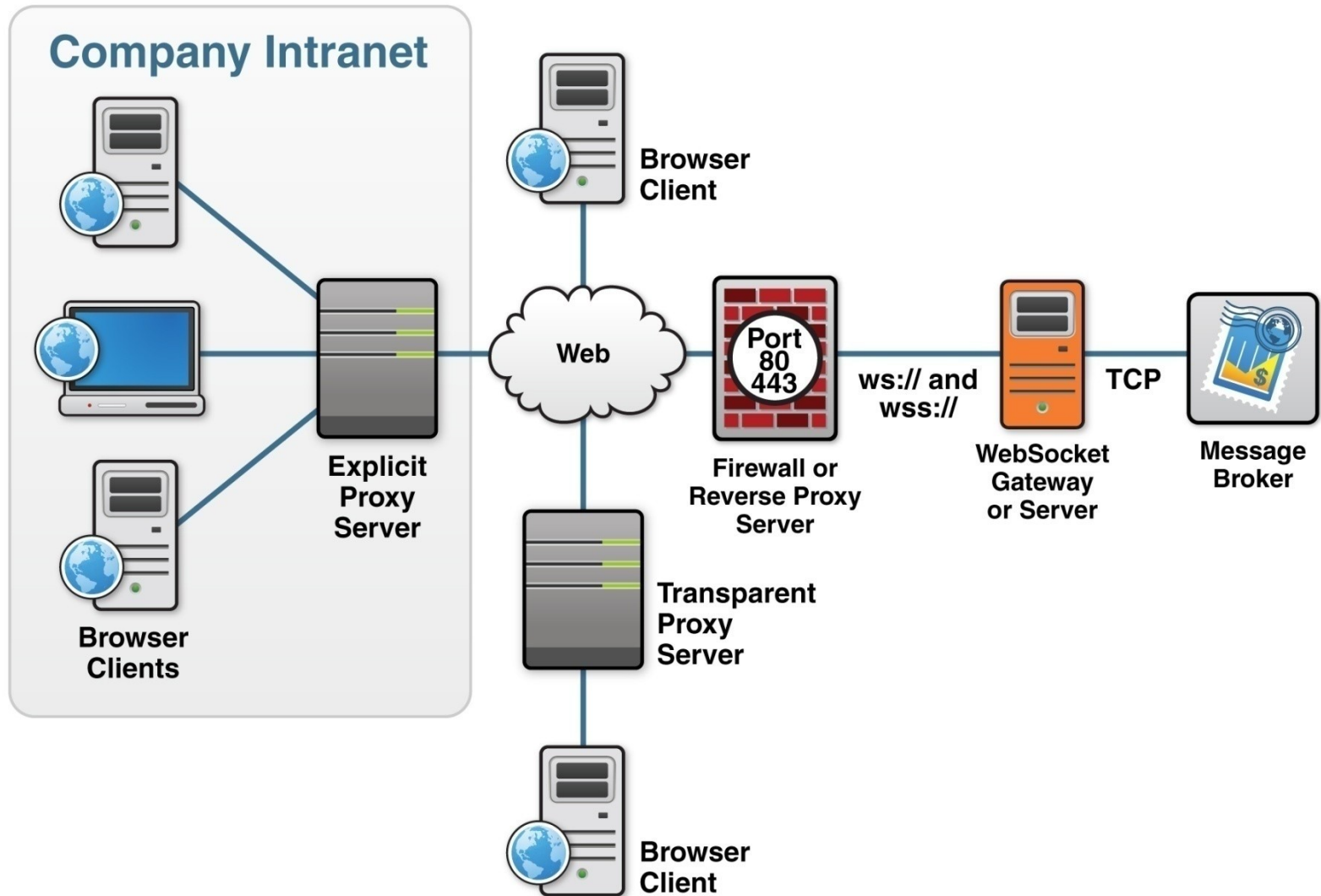


- Have an **Origin** header
  - Contains the request's origin
  - Produced by the browser
  - Cannot be changed by application code
  - Differs from **referer** [*sic*]: **referer** is a complete URL (can include full path)
- Originating page's server must approve (**Access-Control-Allow-\* headers**)



# *Intermediaries*

# Types of Proxy Servers

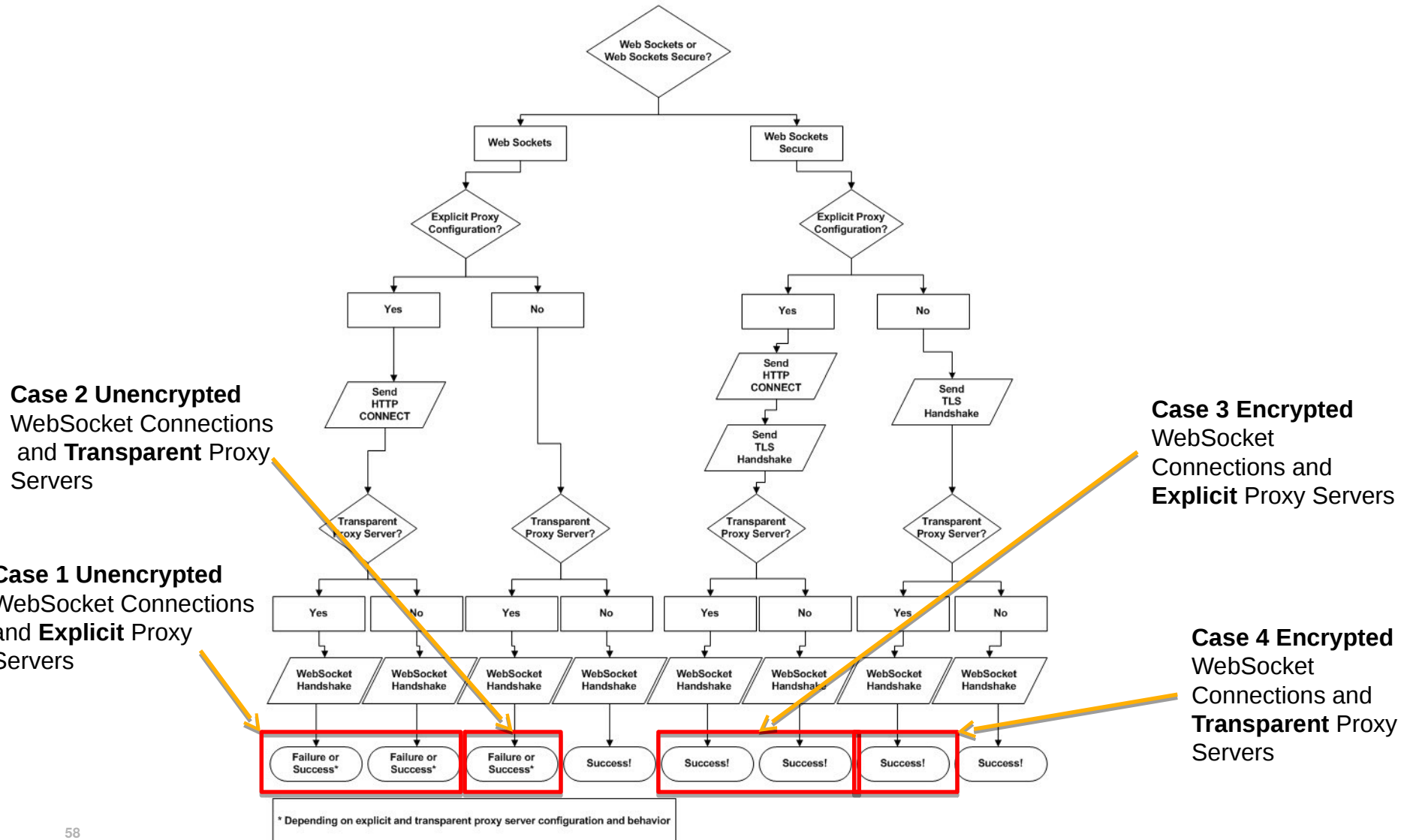


<http://www.infoq.com/articles/Web-Sockets-Proxy-Servers>

- WebSocket protocol is unaware of proxy servers and firewalls
  - HTTP-based handshake

Proxy type	Connection	Outcome
Explicit	Unencrypted	1. HTTP CONNECT 2. WebSocket connection flows to destination
Explicit	Encrypted	
Transparent	Unencrypted	Proxy strips extra headers, connection fails
Transparent	Encrypted	Connection tunnels past proxy

# Proxy Traversal Tree





1. **TCP** (layer-4) work well with WebSockets
2. **HTTP** (Layer 7) expect HTTP traffic, can get confused by WebSocket upgrade traffic.  
May need to be configured to be explicitly aware of WebSocket traffic.

- Usually no specific WebSocket concerns
- Stateful packet inspection may need to know about WS protocol (or use WSS)