



UNIVERSIDAD
COMPLUTENSE
MADRID

NP II

Websockets

Facultad de Informática

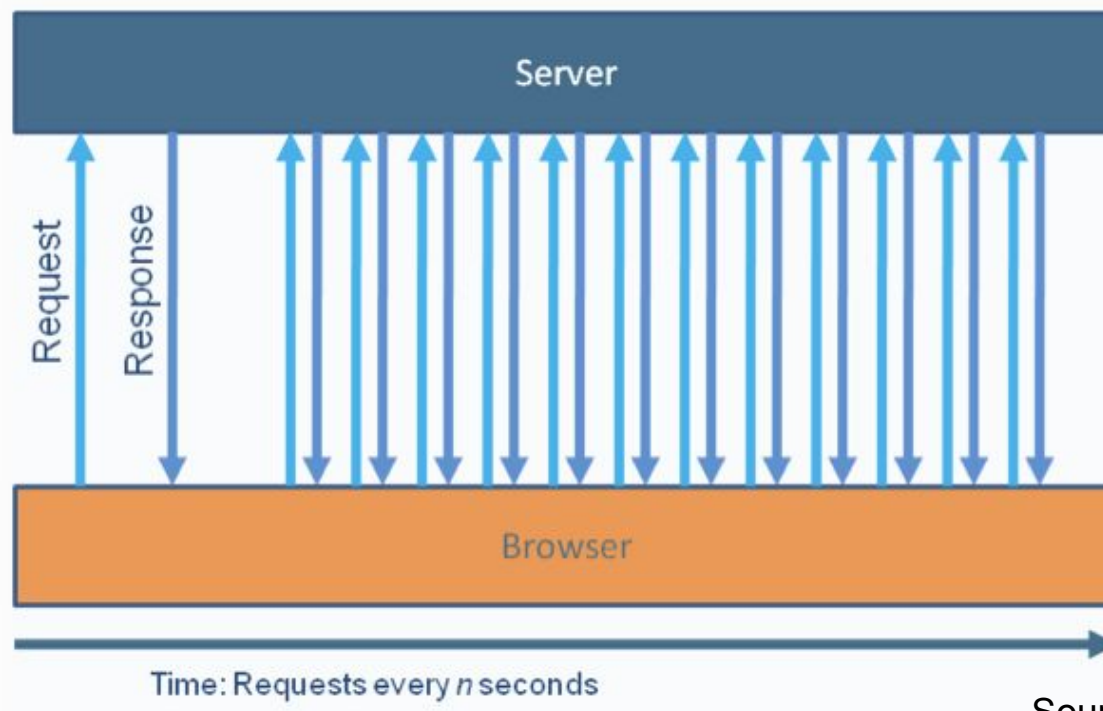
- Websockets
 - Protocol originally proposed by HTML5 for bidirectional full-duplex communication in Web applications
 - Defined by IETF ([RFC 6455](#), 12/2011)
 - API defined by W3C ([HTML living standard](#))

- HTTP/1.1
 - Half-duplex
 - Unidirectional
 - Client requests, server responds
 - Designed for data transfer and other static resources
 - *Stateless*
 - Complex, inefficient

- Full-duplex emulation with HTTP
 - AJAX (Asynchronous JavaScript + XML)
 - Contents can change without changing all webpage
 - Low-latency perception for the user
 - COMET
 - Technique for push on the server side
 - No standard, too complex

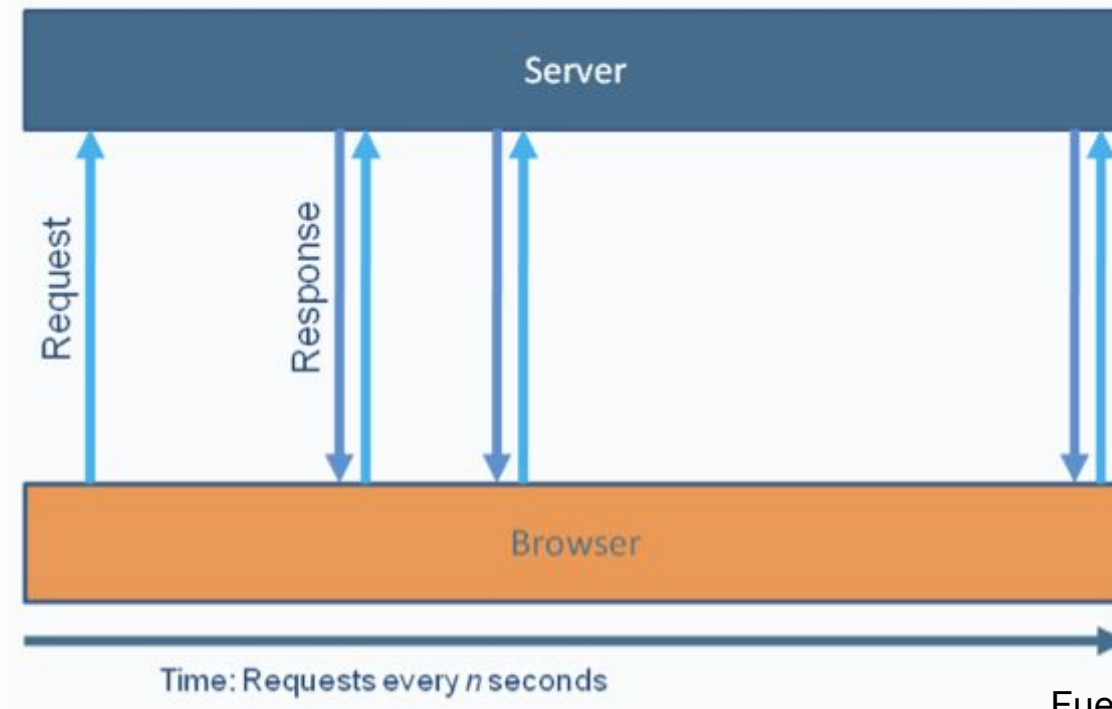


- *Polling*
 - Used in AJAX to simulate real-time
 - Client sends request at regular intervals

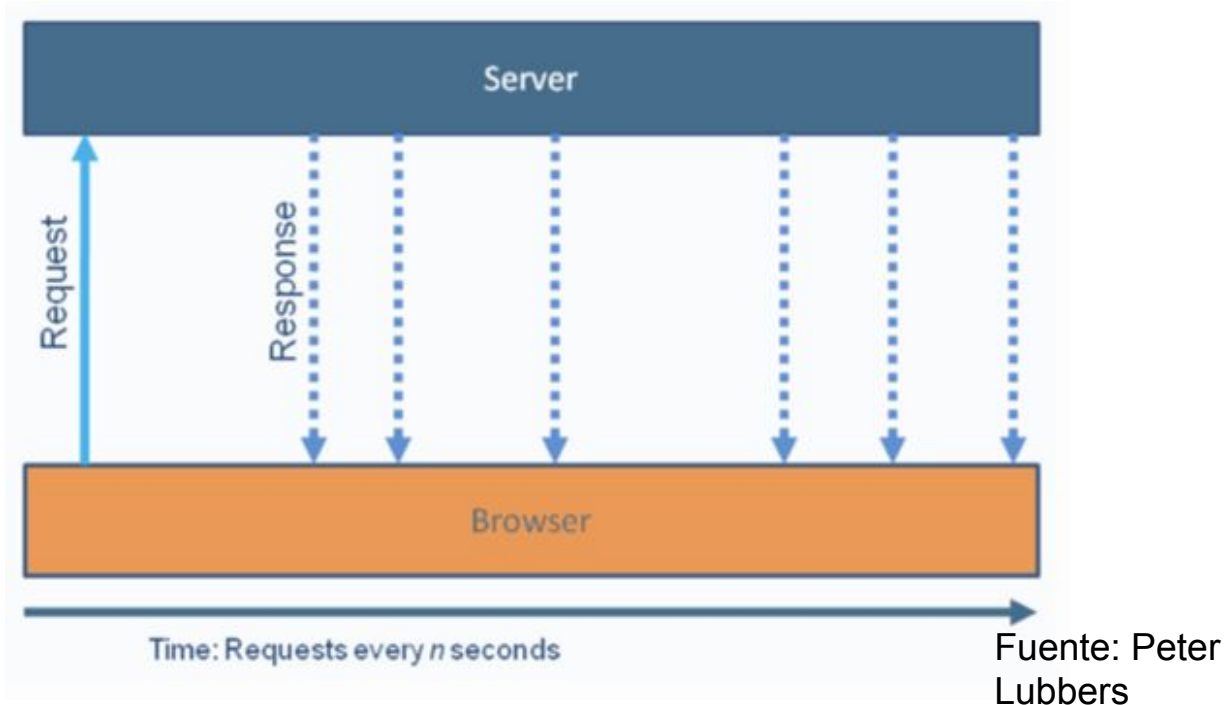


Source: Peter
Lubbers

- *Long Polling (aka Asynchronous polling)*
 - Client sends one request and server retains it open for a while



- *Streaming*
 - More efficient, but some problems
 - *Proxies and firewalls*
 - Response accumulation and necessity of flushes



- Ej. Headers HTTP (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;
```

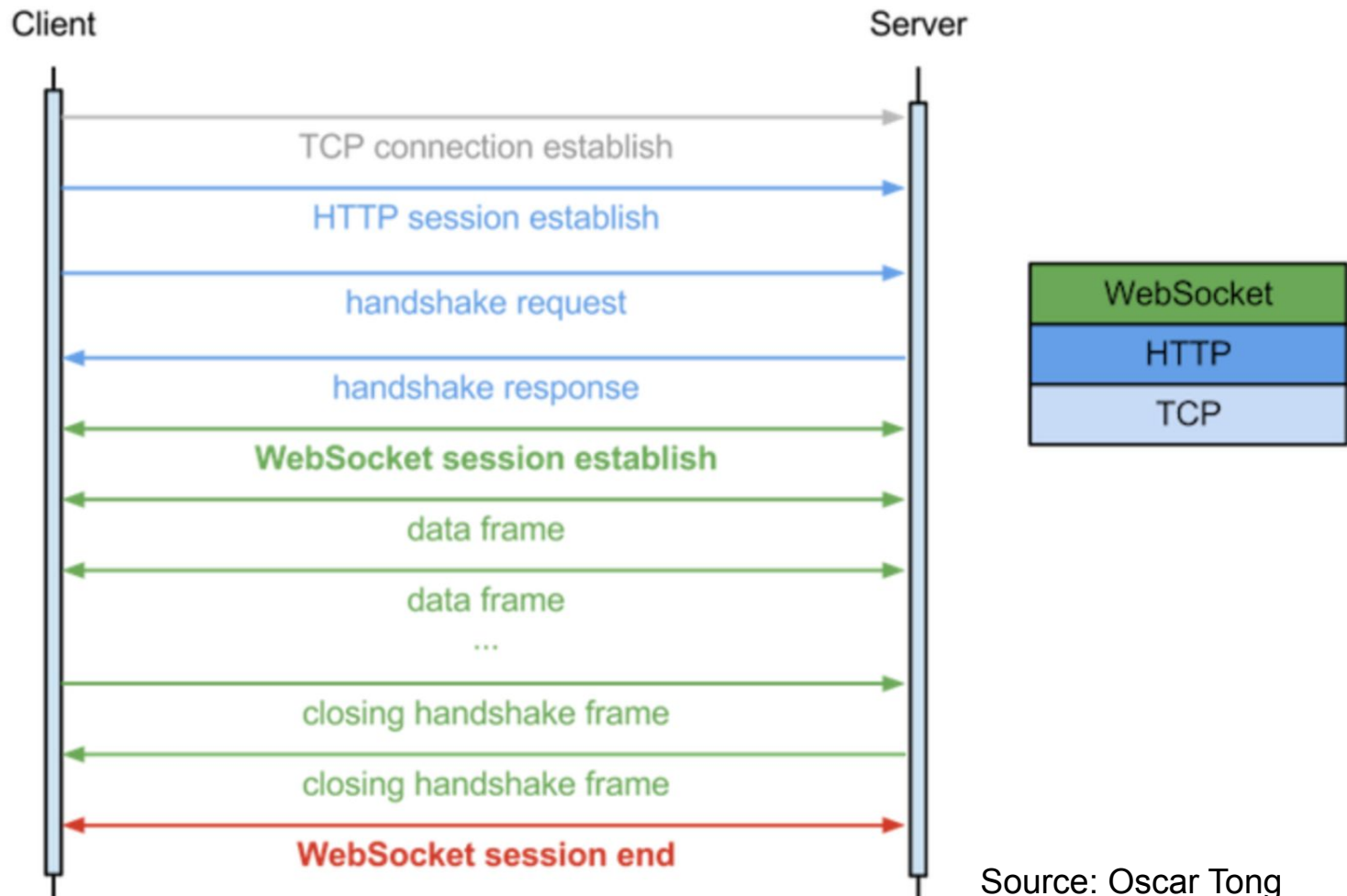

- Ej. Headers HTTP (server)

```
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
```

- HTTP Header Overhead
 - Between 800 and 2000 bytes per *request/reply*
 - In the example, 871 bytes

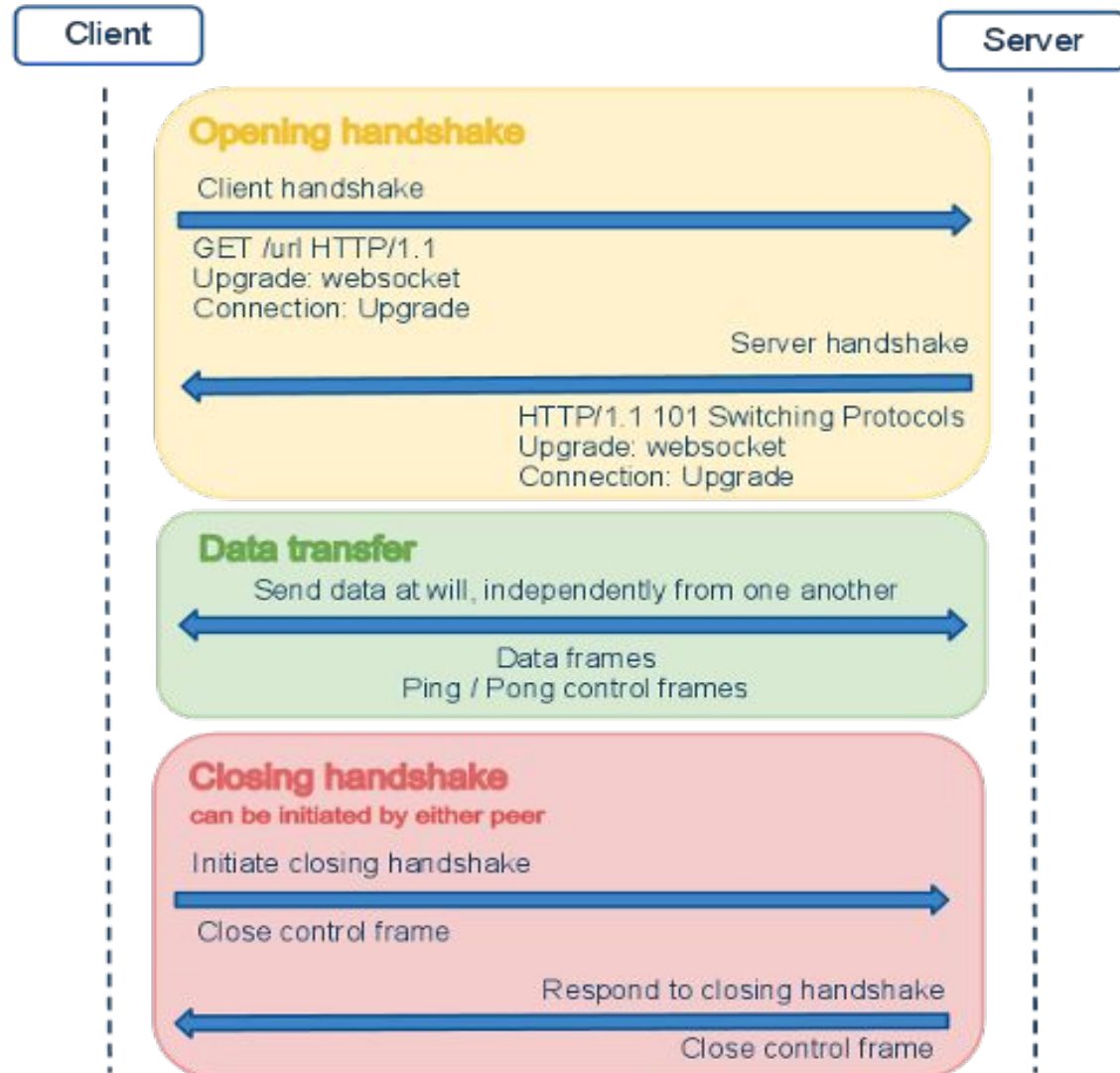
- Features
 - Only one TCP socket
 - Communication full-duplex
 - Shares ports with HTTP/S (80/443)
 - Through *firewalls* and *proxies*
 - Small header size
 - 2 bytes best case!
 - Used as a transport protocol for web apps
 - Layer-7 OSI

- Communication flow

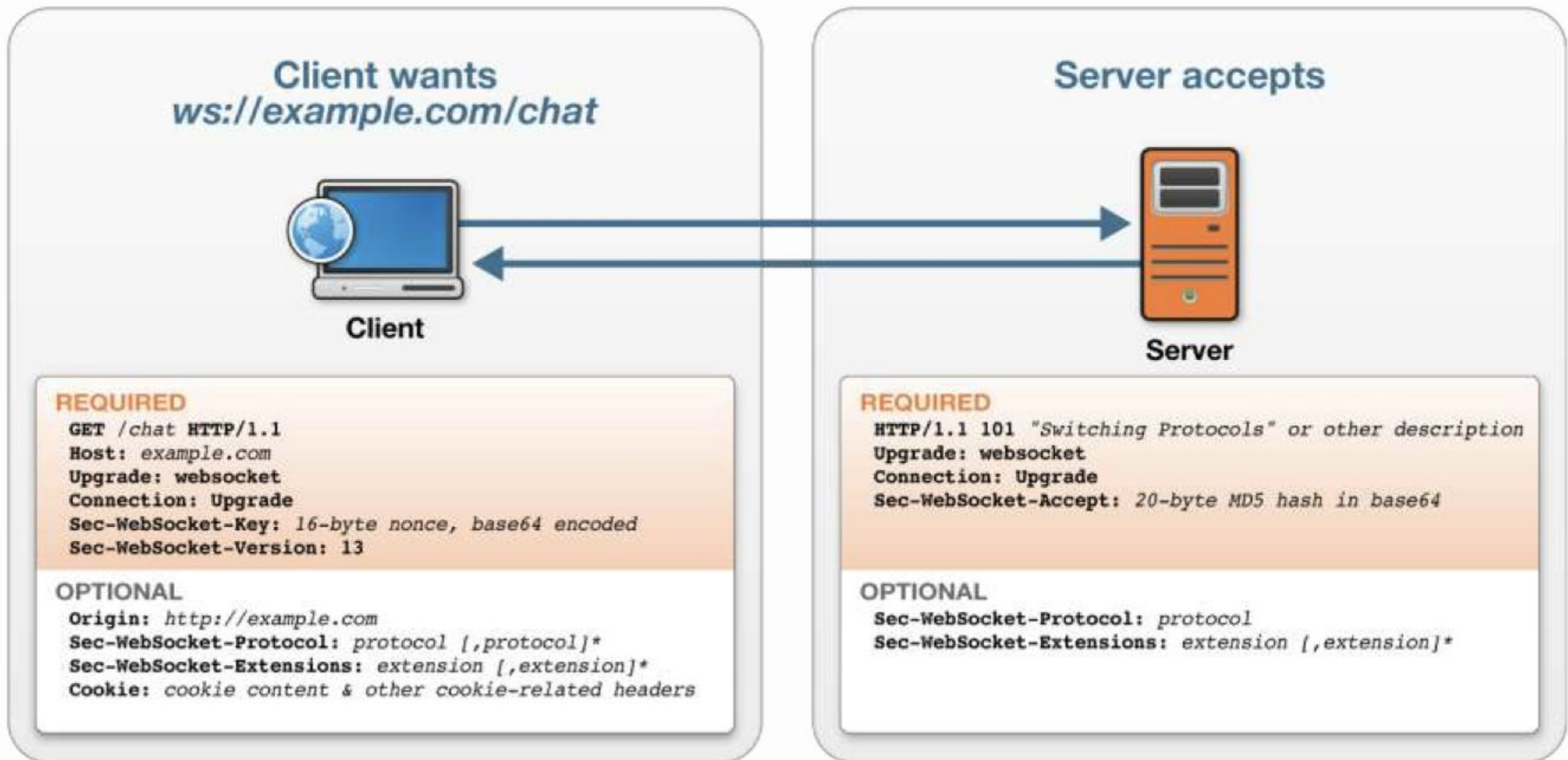


Source: Oscar Tong

- 3 stages



- *Handshake*



- *Handshake challenge*

```
GET /chat HTTP/1.1 Host: www.example.com
```

```
Origin: http://www.example.com
```

```
Upgrade: websocket
```

```
Connection: Upgrade
```

```
Sec-WebSocket-Key: dGhIHnhbXBsZSBub25jZQ==
```

```
Sec-WebSocket-Protocol: chat, superchat
```

```
Sec-WebSocket-Version: 13
```

```
HTTP/1.1 101 Switching Protocols
```

```
Upgrade: websocket
```

```
Connection: Upgrade
```

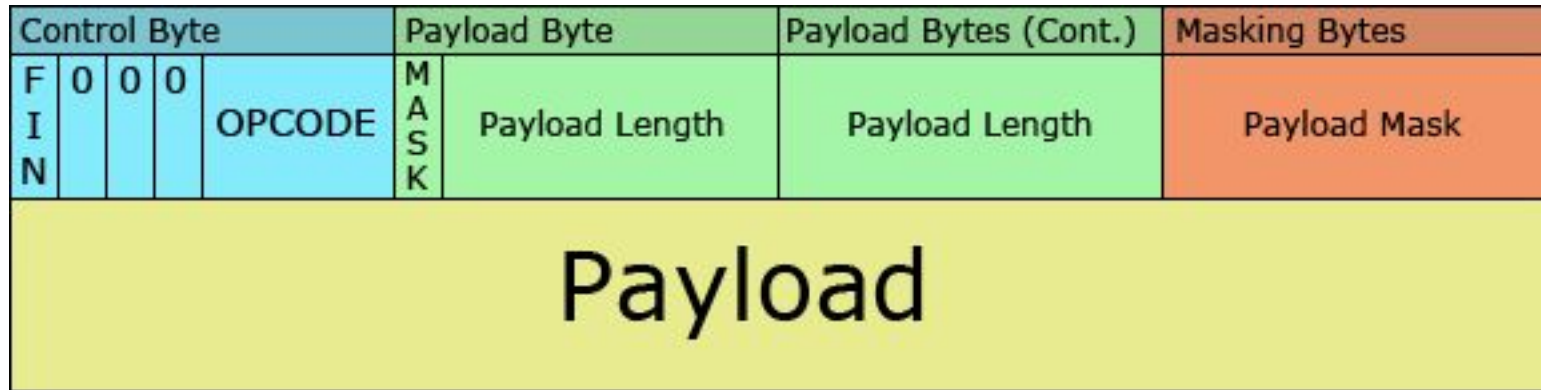
```
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
```

```
Sec-WebSocket-Protocol: chat
```

```
GUID = '258EAF5-E914-47DA-95CA-C5AB0DC85B11';
```

```
Sec-WebSocket-Accept = base64( sha1( Sec-WebSocketKey + GUID ) );
```

- Frames

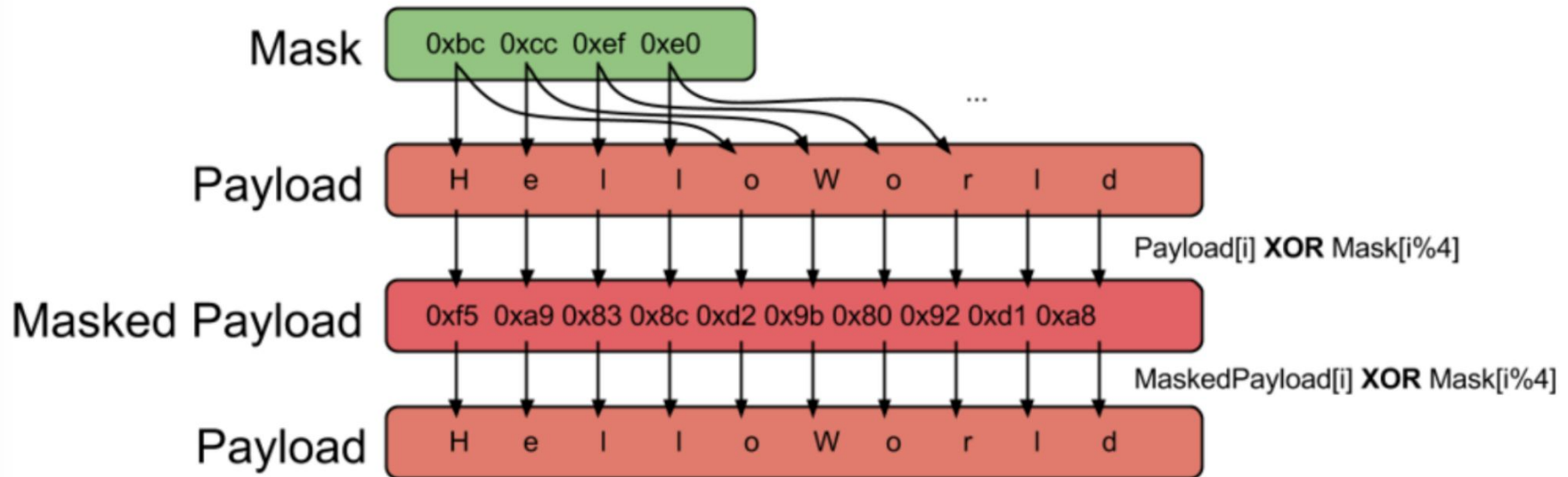


- Small header (≥ 2 bytes)
- *Payload* can be text or binary

- Opcode

Opcode	Tipo de frame
0x0	cont
0x1	text
0x2	binary
0x8	close
0x9	ping
0xA	pong

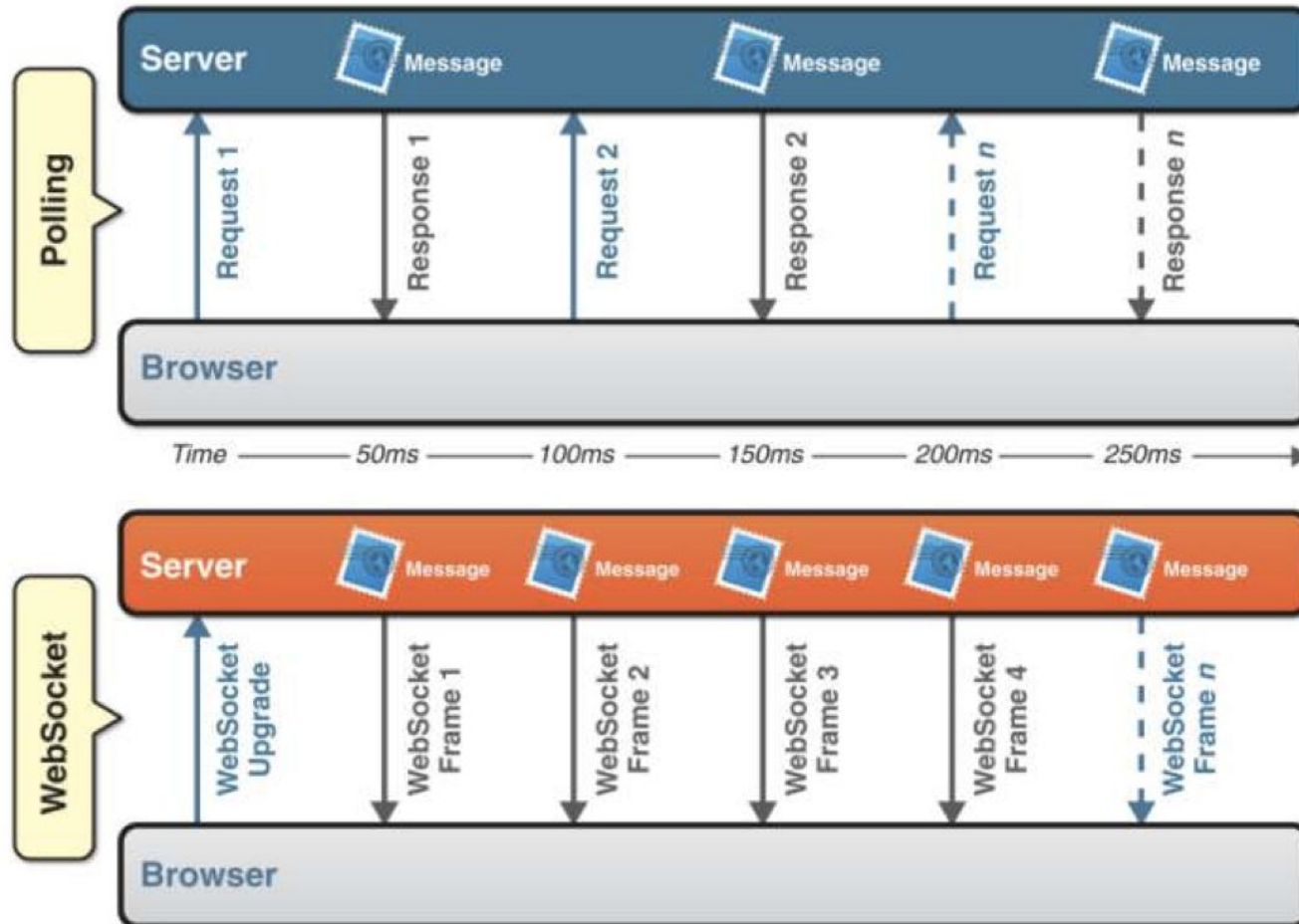
- Masking data (client)



- HTTP vs. Websockets

	HTTP	Websockets
Overhead	800-2K bytes	2-6 bytes (typical)
Latency	New connection every time	Reuse connection
Latency (polling)	Wait to next interval	No wait
Latency (long polling)	None if previous request; time to next request in other case	No wait

- HTTP vs Websockets



- API defined W3C
 - Consult [HTML living standard](#)

Constructor	WebSocket()
Events	onopen, onerror, onmessage, onclose
Methods	send(), close()
Attributes	url, readyState, bufferedAmount, binaryType, <i>extensions</i> , <i>protocol</i>

- Websocket creation

```
var myWebSocket= new WebSocket(url, [protocol] );
```

- URL: ws:// o wss:// (SSL)

- **Attributes**

`myWebSocket.readyState`

- 0: Not ready yet
- 1: establish is done
- 2: Closing now
- 3: closed

`myWebSocket.bufferedAmount`

- Buffer size, used by `send()`

- Event handlers

```
myWebSocket.onopen  
myWebSocket.onmessage  
myWebSocket.onerror  
myWebSocket.onclose
```

- Funciones to invoke when event is triggered

- Example

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

- Soporte navegador
 - <https://caniuse.com/#search=web%20socket>

- Server libraries
 - Kaazing
 - Socket.io (node.js)
 - Pywebsocket
 - Apache-websocket
 - Netty
 - Node.js
 -

- Client libraries
 - Web-socket-js (JavaScript)
 - Java WebSocket Client (Java)
 - Arduino WebSocket client (C++)
 - Libwebsockets (C)
 - ...

- Examples:
 - <http://www.websocket.org/echo.html>
 - <http://demos.kaazing.com/echo>

Kaazing WebSocket Echo Demo

This demo uses the WebSocket API to send text messages to the Kaazing Gateway Echo service, which echoes back the messages.

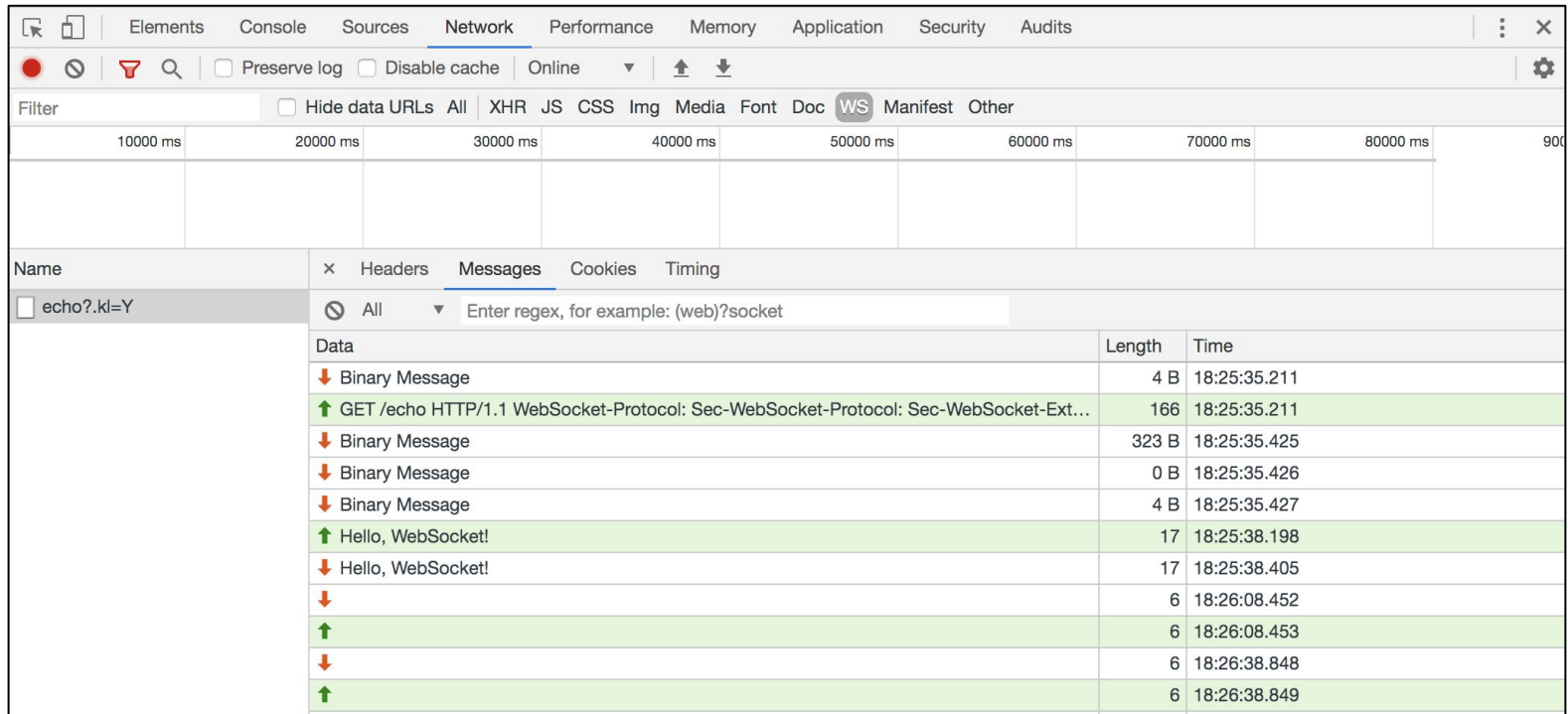
Location

Message

Log messages

```
CONNECT: wss://demos.kaazing.com/echo
CONNECTED
SEND TEXT: Hello, WebSocket!
RECEIVED TEXT: Hello, WebSocket!
CLOSE
CLOSED: (1005)
```

- In Chrome (DevTools)
 - Analyze source code
 - Visualize headers and messages



The screenshot shows the Chrome DevTools Network tab with the 'Messages' sub-tab selected. The filter is set to 'echo?.kl=Y'. The table below displays the captured messages:

Name	Length	Time
Binary Message	4 B	18:25:35.211
GET /echo HTTP/1.1 WebSocket-Protocol: Sec-WebSocket-Protocol: Sec-WebSocket-Ext...	166	18:25:35.211
Binary Message	323 B	18:25:35.425
Binary Message	0 B	18:25:35.426
Binary Message	4 B	18:25:35.427
Hello, WebSocket!	17	18:25:38.198
Hello, WebSocket!	17	18:25:38.405
	6	18:26:08.452
	6	18:26:08.453
	6	18:26:38.848
	6	18:26:38.849

- In Wireshark

Wireshark · Packet 289 · wireshark_en0_20191106215720_nm7uwt

- ▶ Frame 289: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface 0
- ▶ Ethernet II, Src: Apple_0c:dc:e4 (8c:85:90:0c:dc:e4), Dst: Amper_46:20:b2 (8c:0c:a3:46:20:b2)
- ▶ Internet Protocol Version 4, Src: 192.168.1.161, Dst: 34.209.17.111
- ▶ Transmission Control Protocol, Src Port: 49446, Dst Port: 80, Seq: 810, Ack: 588, Len: 11
- ▼ WebSocket
 - 1... = Fin: True
 - .000 = Reserved: 0x0
 - 0001 = Opcode: Text (1)
 - 1... = Mask: True
 - .000 0101 = Payload length: 5
 - Masking-Key: 996ecb65
 - Masked payload
 - Payload
 - ▼ Line-based text data
 - Hello

0000	8c 0c a3 46 20 b2 8c 85 90 0c dc e4 08 00 45 02	...FE.
0010	00 3f 00 00 40 00 40 06 44 2e c0 a8 01 a1 22 d1	.?..@.@. D.....".
0020	11 6f c1 26 00 50 83 48 54 99 dc 3a be 85 80 18	.o.&.P.H d.:....
0030	10 19 18 53 00 00 01 01 08 0a 38 5a 08 09 60 7f	...S.... ..8Z..`.
0040	1e 49 81 85 99 6e cb 65 d1 0b a7 09 f6	.I...n.e

48 65 6c 6c 6f (Hello)

Frame (77 bytes) | Unmasked data (5 bytes)

No.: 289 · Time: 35.136586 · Source: 192.168.1.161 · Destination: 34.209.17.111 · Protocol: WebSocket · Length: 77 · Info: WebSocket Text [FIN] [MASKED]

Help Close

- www.websocket.org/
- tools.ietf.org/html/rfc6455
- html.spec.whatwg.org/multipage/web-sockets.html
- enterprisewebbook.com/ch8_websockets.html
- github.com/kaazing/tutorials/tree/develop/mqtt
- www.tornadoweb.org/en/stable/websocket.html
- kaazing.com