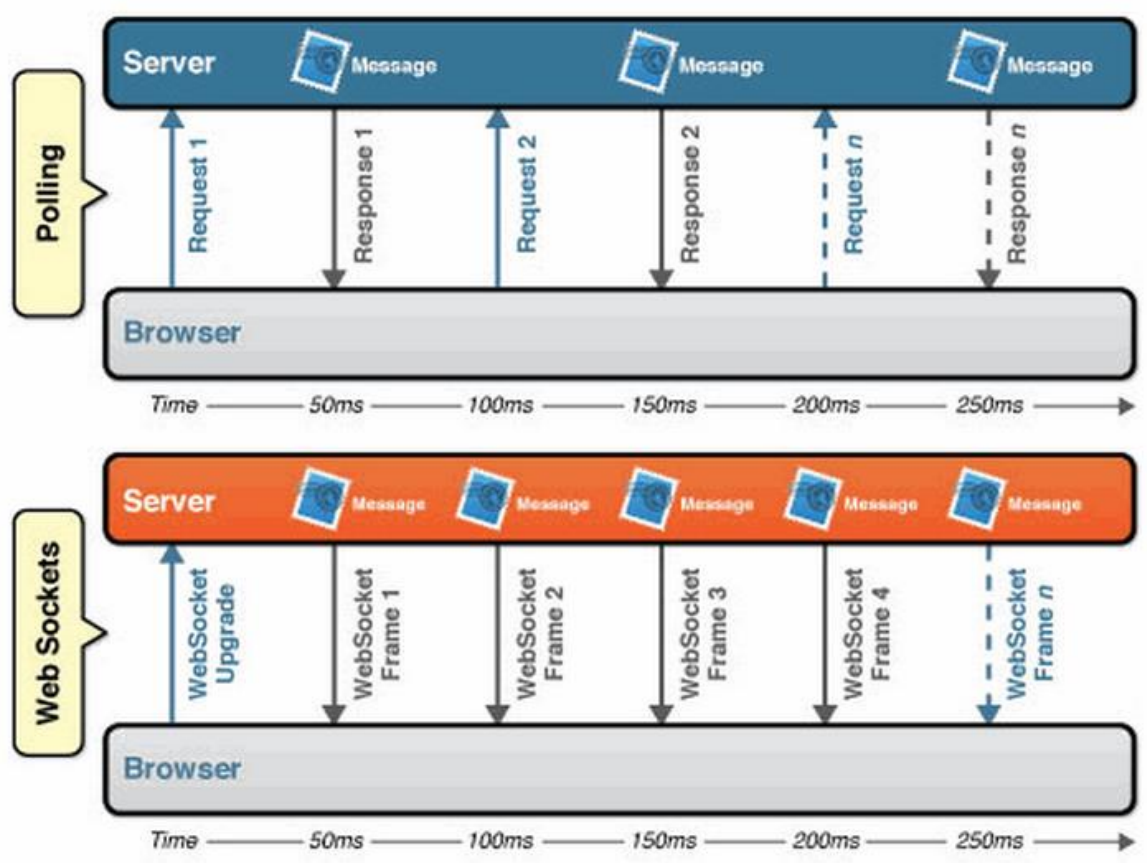
WebSocket

- References:
  - RFC 6455
  - The WebSocket API - W3C draft specification of the API
  - The WebSocket protocol - Internet-Draft published by the IETF HyBi Working Group

Background

- In Brower, AJAX is still inefficient due to polling.

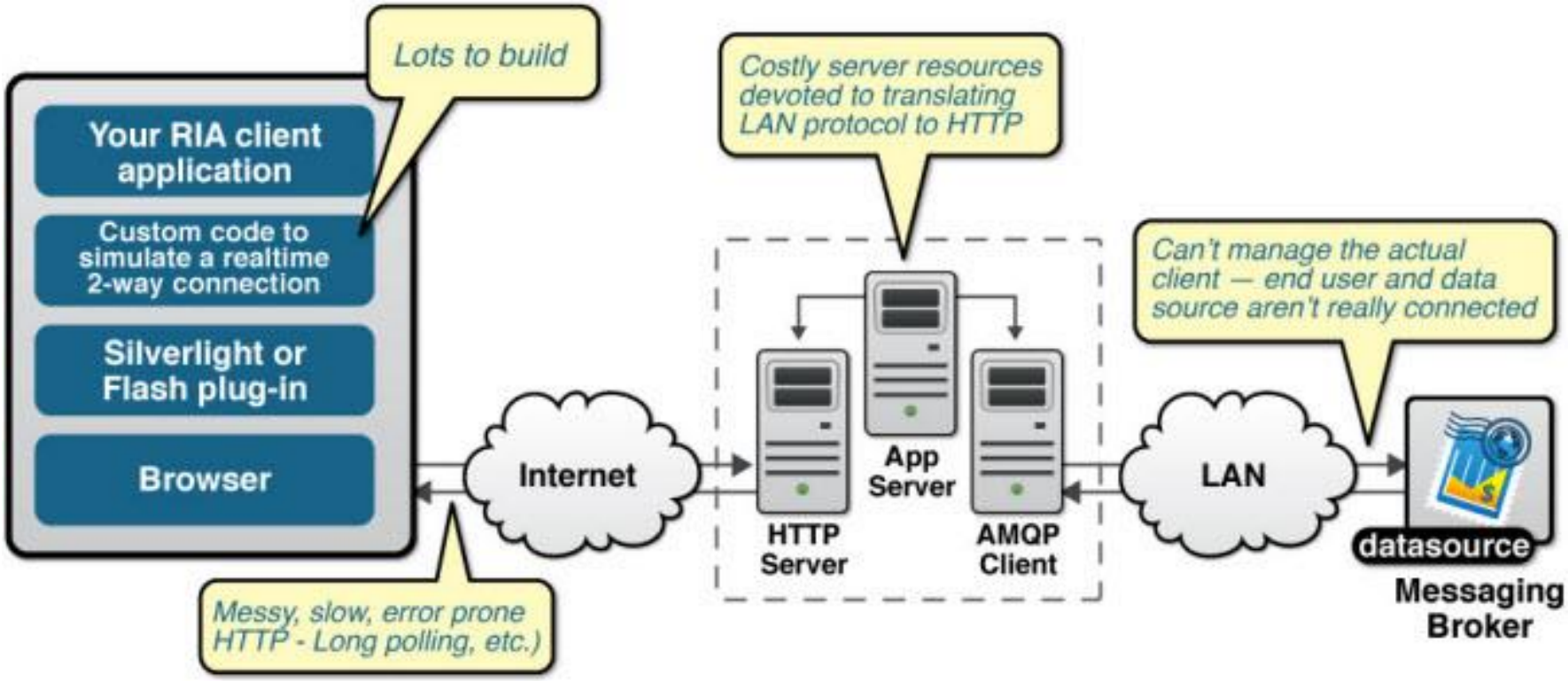


← Polling by AJAX

← Real-time response for WebSocket

[WebSocket.org]

Problem


Lots to build
Your RIA client application
Custom code to simulate a realtime 2-way connection
Silverlight or Flash plug-in
Browser
Messy, slow, error prone HTTP - Long polling, etc.)
Internet
Costly server resources devoted to translating LAN protocol to HTTP
HTTP Server
App Server
AMQP Client
Can't manage the actual client — end user and data source aren't really connected
LAN
datasource
Messaging Broker

Motivation

- Solution of the problem:
  - Support socket for browsers.
- New problem:
  - Security Issue
    - E.g., scan your intranet.
- Solution of Java Applet:
  - Connect back to the original server.
- Problem of Applet Solution:
  - The original server becomes bottleneck.

Goal of WebSocket

- Current Situation:
  - Ordinary TCP connections to port numbers other than 80 are frequently blocked by administrators outside of home environments.
- Goal:
  - A way to overcome these restrictions and provide similar functionality with some additional protocol overhead while multiplexing several WebSocket services over a single TCP port.
  - A technology providing for bi-directional, full-duplex communications channels, over a single TCP socket.

Standardization

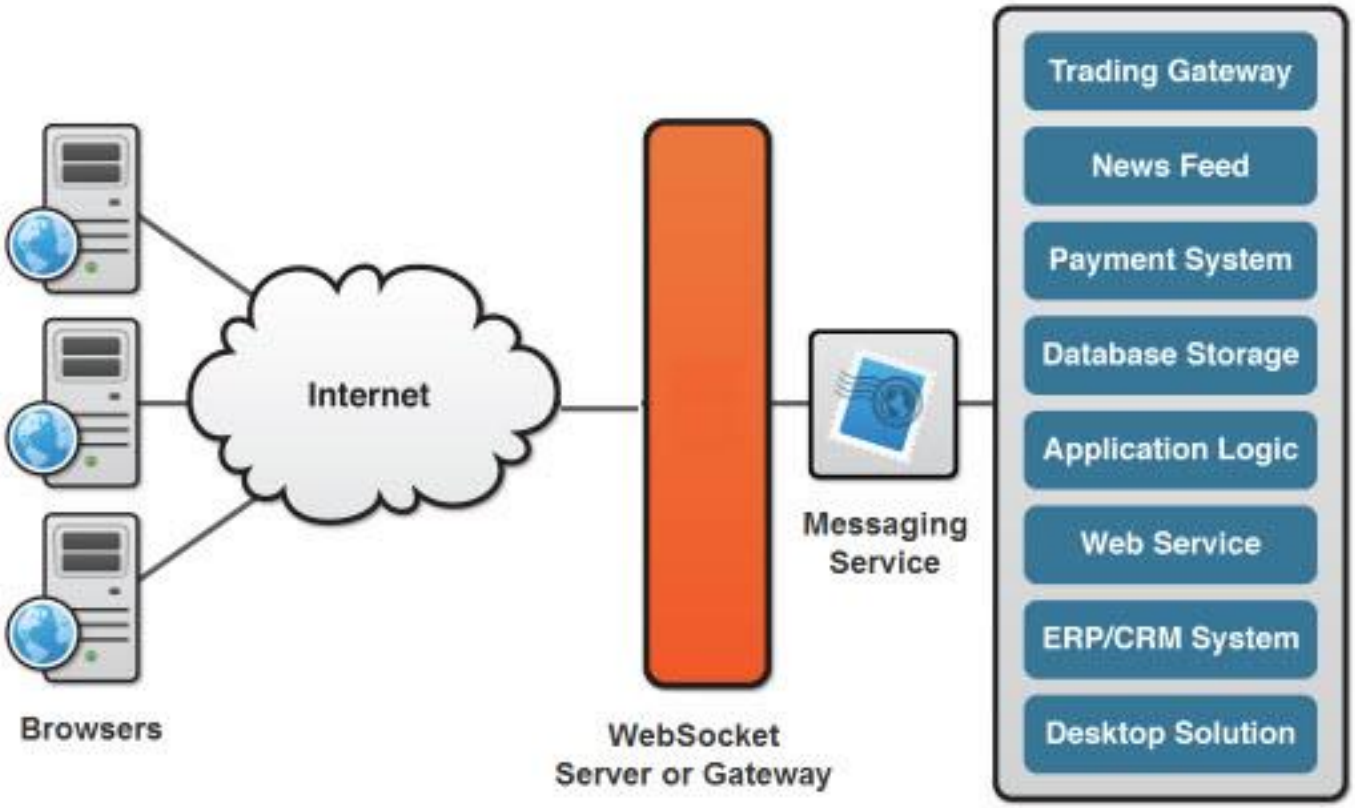
- Standardization:
  - The WebSocket API
    - Standardized by the W3C,
  - WebSocket protocol
    - Standardized by the IETF as RFC 6455.

Browser Support

http://caniuse.com/#feat=websockets

Show all versions	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	IE Mobile
								2.1		
						3.2		2.2		
						4.0-4.1		2.3		
						4.2-4.3		3.0		
	8.0	22.0				5.0-5.1		4.0		
	9.0	23.0	28.0	5.1		6.0-6.1		4.1	7.0	
Current	10.0	24.0	29.0	6.0	16.0	7.0	5.0-7.0	4.2	10.0	10.0
Near future	11.0	25.0	30.0	7.0	17.0					
Farther future			31.0							

Architecture



WebSocket Protocol

- Browser:

```
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
```

- Server:

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

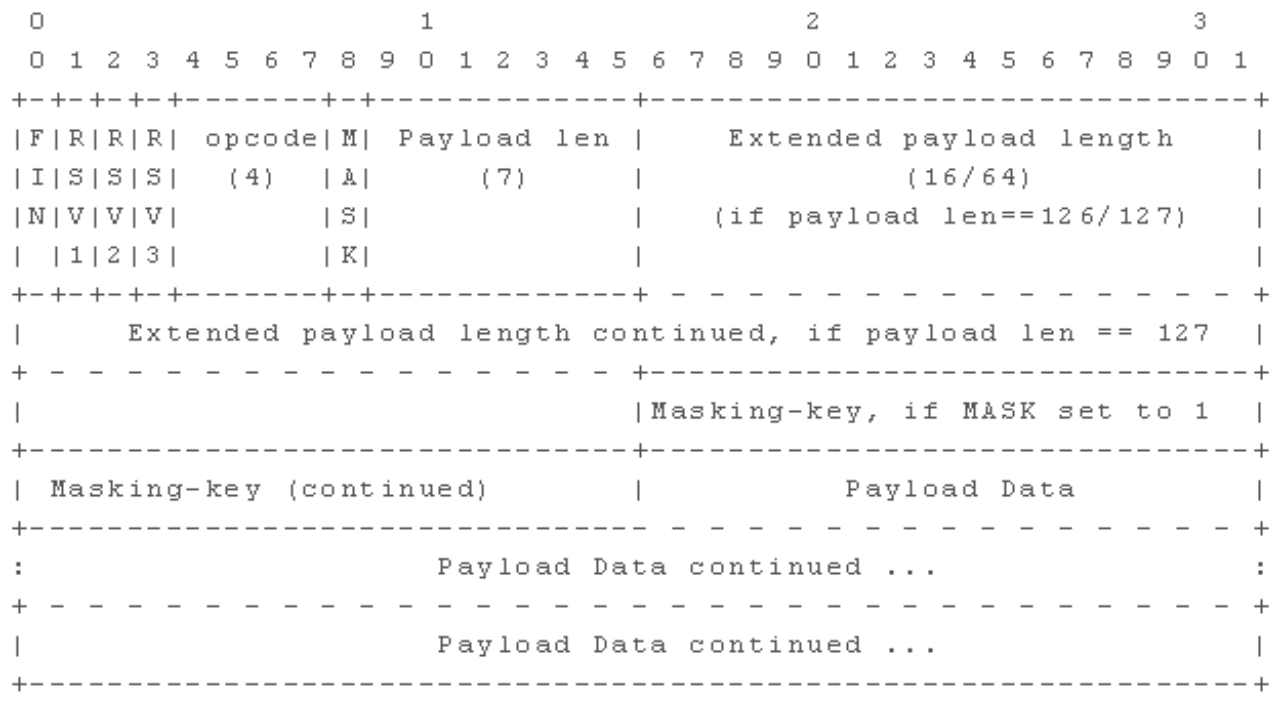
Handshake

- client request
  - GET / HTTP/1.1
  - Upgrade: websocket
  - Connection: Upgrade
  - Host: example.com
  - Origin: null
  - Sec-WebSocket-Key: sN9cRrP/n9NdMgdcy2VJFQ==
  - Sec-WebSocket-Version: 13
- server response
  - HTTP/1.1 101 Switching Protocols
  - Upgrade: websocket
  - Connection: Upgrade
  - Sec-WebSocket-Accept: fFBooB7FAkLlXgRSz0BT3v4hq5s=
  - Sec-WebSocket-Origin: null
  - Sec-WebSocket-Location: ws://example.com/

Sec-WebSocket-Accept = BASE-64(SHA-1(Sec-WebSocket-Key+Magic-Key))
Magic-Key is fixed to be 258EAFA5-E914-47DA-95CA-C5AB0DC85B11

RFC 6455 Frame Header

- After handshake, all messages should add lightweight headers (not HTTP).
  - Browsers in clients will encode/decode headers automatically.
  - Server should implement the encoding/decoding of headers.



Using the HTML5 WebSocket API

- Create a web socket.

```
var myWebSocket = new WebSocket("ws://www.websockets.org");
```

- Set up event handlers for receiving messages.

```
myWebSocket.onopen = function(evt) { alert("Connection open ..."); };
myWebSocket.onmessage = function(evt) { alert( "Received Message: " + evt.data); };
myWebSocket.onclose = function(evt) { alert("Connection closed."); };
```

- Send messages.

```
myWebSocket.send("Hello Web Sockets!");
myWebSocket.close();
```

Resources

- C++ sha1
- C++ Base64
- php webSocket Server(select implement)
- IE10 support webSocket spec