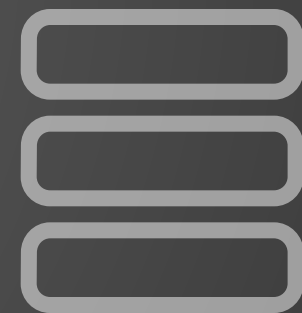
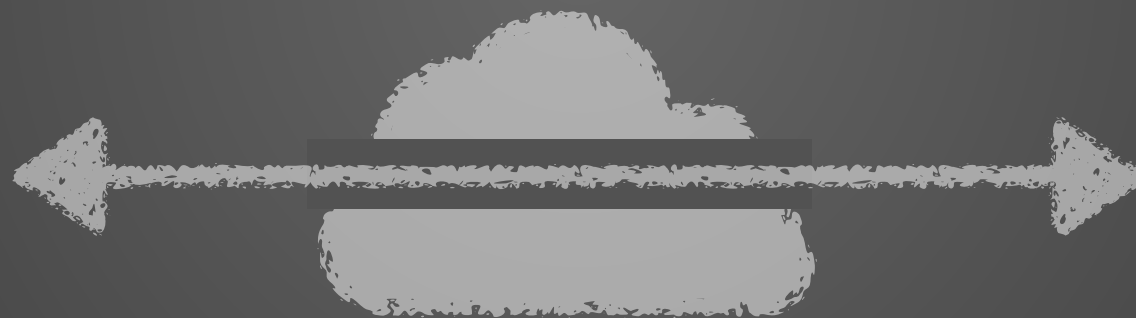
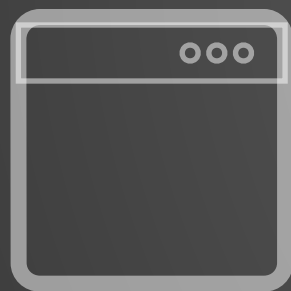


High Performance WebSocket



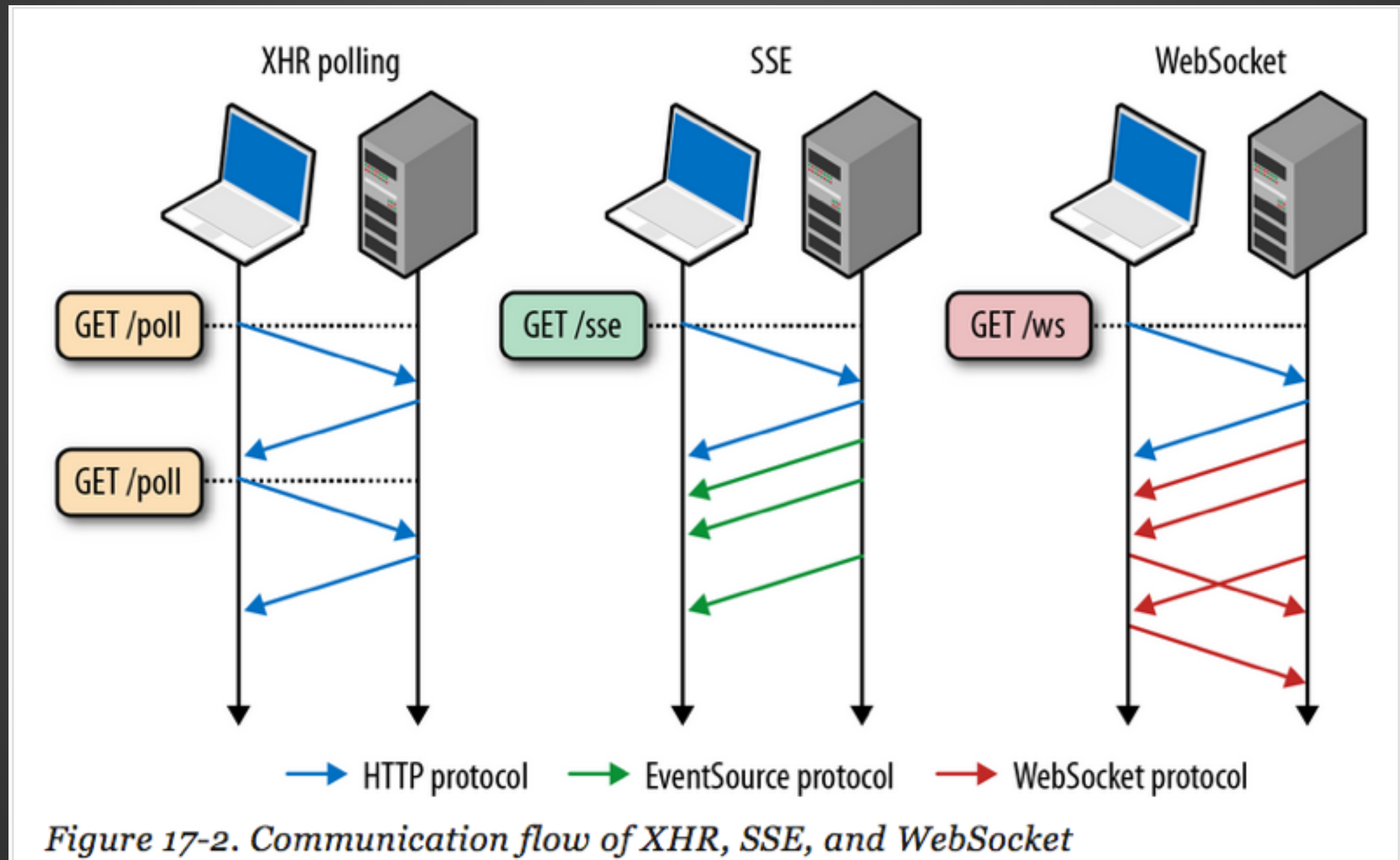
@wesleyhales

Real Time Enablers



In 2015, bidirectional client <-> server communication can only be achieved with AJAX and/or WebSocket

Client Server Connectivity



AJAX

When you make an AJAX request from the browser you don't have to think about TCP.

The underlying complexity is abstracted away.

WebSocket

When you send a message over
a WebSocket connection, you
have to think about everything.

Subprotocols, Fallbacks, Network

You might not need a WebSocket

Posted by justin on June 24th, 2014 — Filed under [Protocols](#), [WebSockets](#)
([Permalink](#))

Before I begin, I want to say that WebSockets are great. I've even implemented [RFC 6455](#) myself in [Zurl](#) and [Pushpin](#), which are used by the [Fanout.io](#) service. Fanout.io also supports WebSockets via its [XMPP-FTW interface](#) using [Primus](#).

However, after spending quite some time working on large distributed applications and gaining a greater appreciation of REST and messaging patterns, I feel that much of what typical web applications want to accomplish with WebSockets (or with socket-like abstractions) is perhaps better solved by other means.

HTTP streaming and Server Sent Events

WebSockets aren't the only game in town when it comes to efficiently pushing data to browsers. Consuming an HTTP response of indefinite length with

<http://blog.fanout.io/2014/06/24/you-might-not-need-a-websocket/>

Real Time?

- Rapid client <-> server communication
- Real time execution of events in browser
- Binary data delivery

WebSocket

The Promise Idea

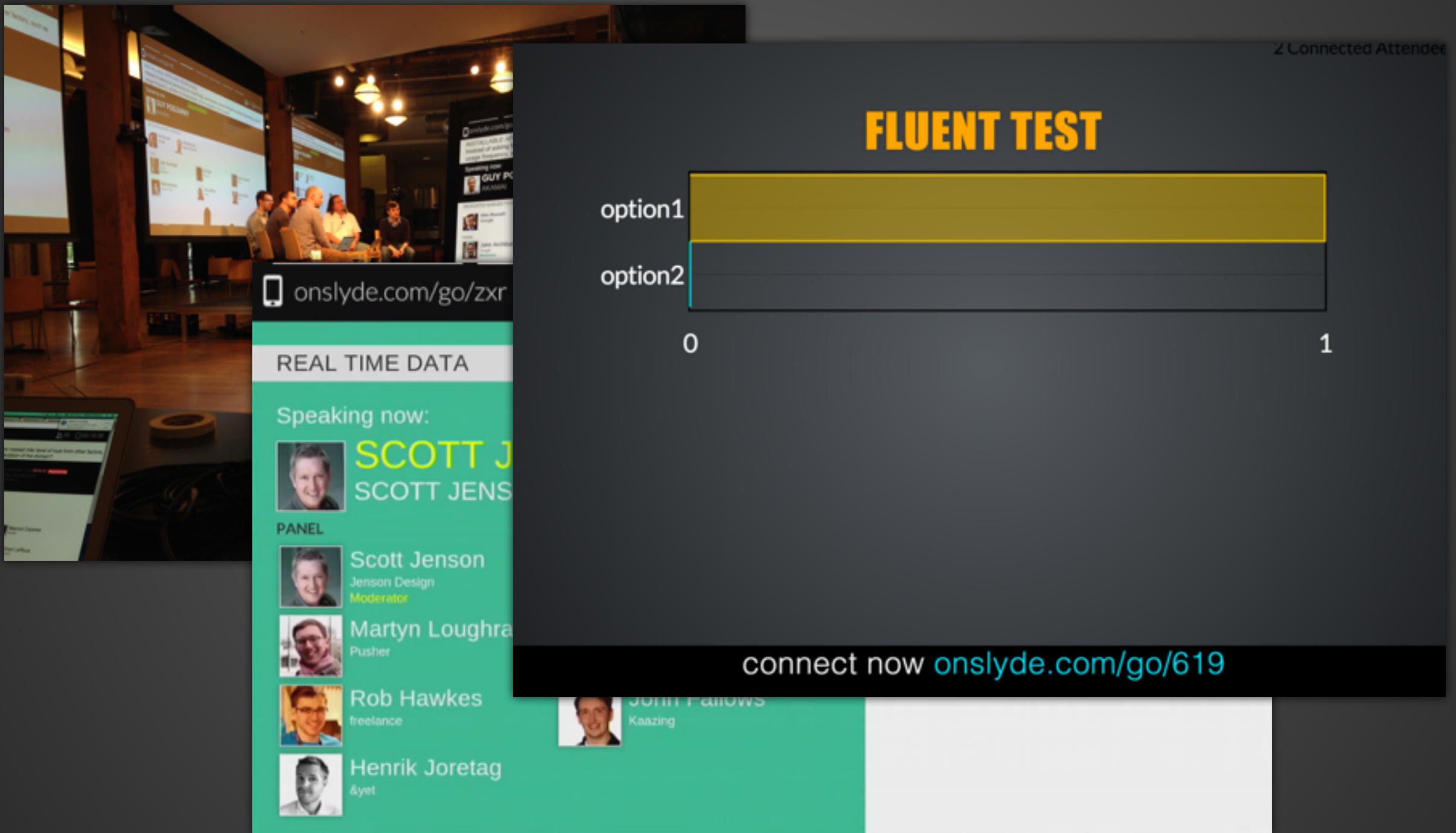
“facilitating live content and the creation of real-time games”



The screenshot shows the Wikipedia article for **WebSocket**. The page layout includes the Wikipedia logo and navigation links on the left, and article content on the right. A callout box highlights the following sentence in the article's introduction:

The WebSocket protocol makes more interaction between a browser and a website possible, facilitating live content and the creation of real-time games.

So let's give that a try...



2 Connected Attendees

FLUENT TEST

option1

option2

0 1

onslyde.com/go/zxr

REAL TIME DATA

Speaking now:

SCOTT J
SCOTT JENSON

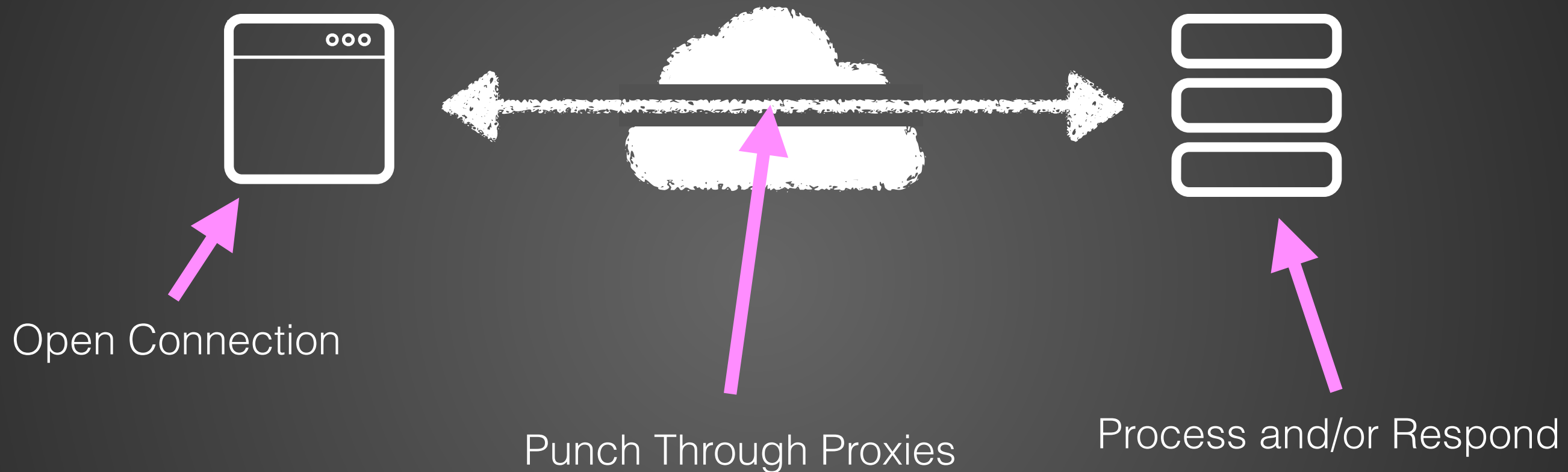
PANEL

- Scott Jenson
Jenson Design
Moderator
- Martyn Loughran
Pusher
- Rob Hawkes
freelance
- Henrik Jorettag
&yet

connect now onslyde.com/go/619

JOHN FALLOWS
Kaazing

High Performance WS



[Live Demo](#)

High Performance Hurdles

- Server Side Connections
- Networks and Proxies
- Subprotocols
- Time
- Services vs. Frameworks

Server Side Connections

caustik's blog

Node.js w/ 1M concurrent connections!
with 65 comments

How Disqus Went Realtime With 165K Messages Per Second And Less Than .2 Seconds Latency

MONDAY, APRIL 28, 2014 AT 9:21AM

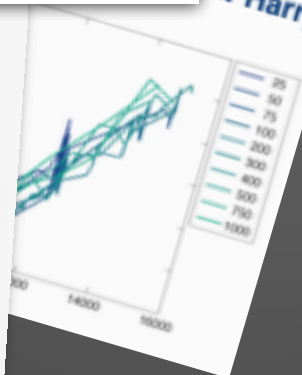
Benchmarks

By [simon](#) 21/09/2011 CometD

[Tweet](#)

Slightly more than one year has passed since the [last CometD 2 benchmarks](#), and more than three years since the [CometD 1 benchmark](#). During this year we have done a lot of work on [CometD](#), both by adding features and by continuously improving performance and stability to make it faster and more scalable.

With the upcoming CometD 2.4.0 release, one of the biggest changes is the



- Up to 1M concurrent connections
- Low Overhead
- Bidirectional

Networks and Proxies

Networks & Proxies

TL;DR Use TLS

99.9% Guaranteed Connection

Networks

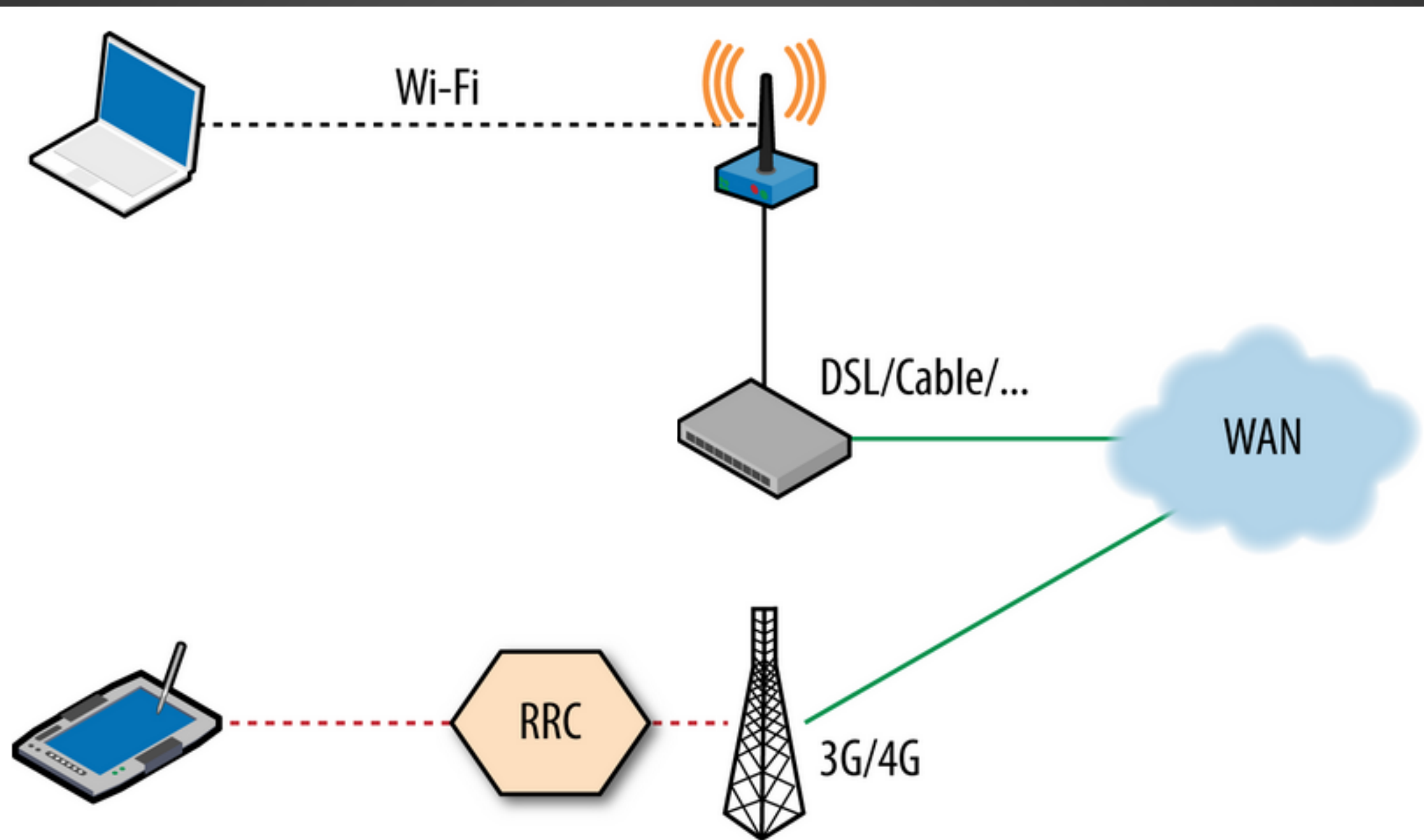


Figure 7-4. Radio Resource Controller

Networks

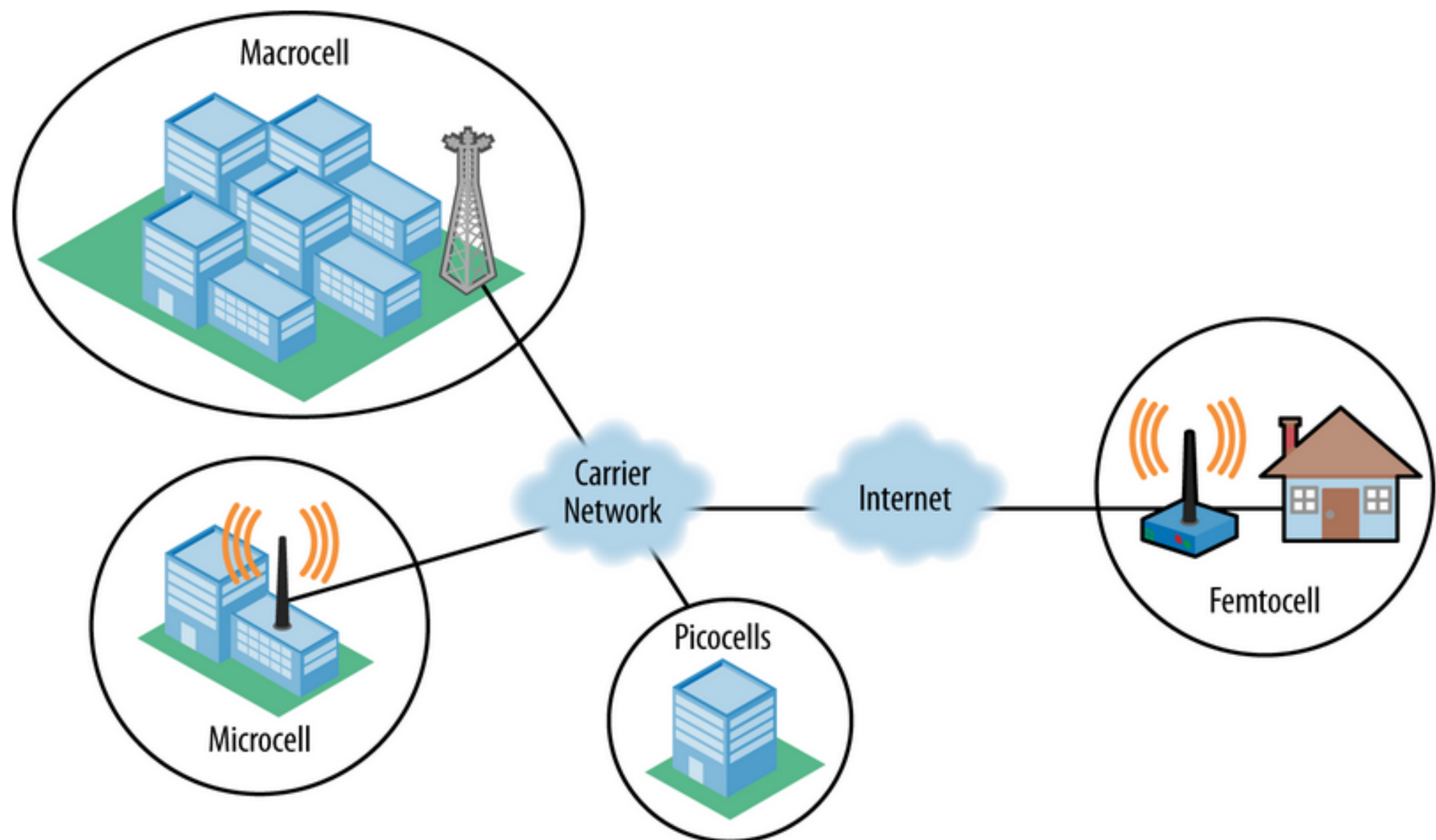


Figure 7-14. Heterogeneous network infographic (Ericsson)

Networks

Service Provider	Sessions
tw telecom holdings inc.	282
service provider corporation	52
(not set)	37
t-mobile usa inc.	13
cellco partnership dba verizon wireless	6
softlayer technologies inc.	5
sprint nextel corporation	5
comcast cable communications holdings inc	4
agile-inap	3
vodafone limited	2

Subprotocols

Because one isn't enough

Subprotocols

Meta data for the client <-> server transmission

```
var ws = new WebSocket('wss://example.com/socket',  
                        ['appProtocol', 'appProtocol-v2']);  
  
ws.onopen = function () {  
    if (ws.protocol == 'appProtocol-v2') {  
        ...  
    } else {  
        ...  
    }  
}
```

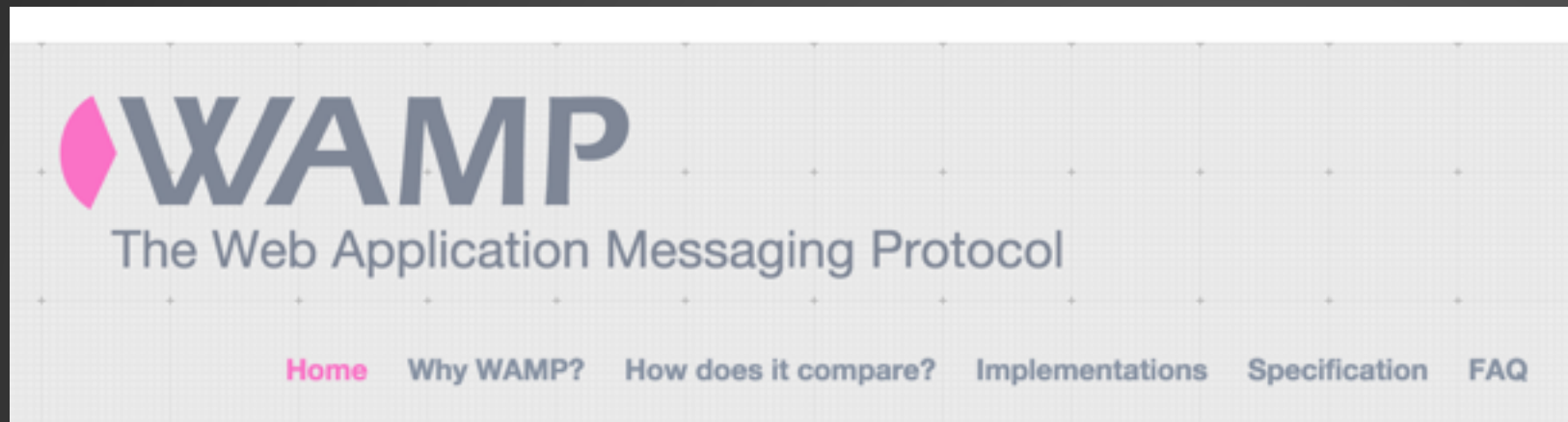
**No effect on the core WebSocket API*

Messaging Protocol



Subprotocols

You have to use an envelope e.g. JSON



WAMP is an open standard [WebSocket](#) subprotocol that provides two application messaging patterns in one unified protocol:

× Headers Frames Cookies		
Data	Length	Time
{"onslydeEvent":{"sessionId":"619","fire":function(){window.eventObjb = document.createEvent('Event');eventO...	244	10:17:04 AM
{"onslydeEvent":{"sessionId":"619","fire":function(){window.eventObjg = document.createEvent('Event');eventO...	317	10:17:04 AM
activeOptions:null,null,0:0	27	10:17:04 AM
Binary Frame (Opcode 2, mask)	0	10:17:04 AM
{"remoteMarkup":"%3Ch2 Binary Frame (Opcode 2, mask) 63EMy%20awesome%20presentation%3C%2Fh2%3E"}	86	10:17:04 AM
{"onslydeEvent":{"sessionId":"619","fire":function(){window.eventObjg = document.createEvent('Event');eventO...	358	10:17:03 AM
{"onslydeEvent":{"sessionId":"619","fire":function(){window.eventObjb = document.createEvent('Event');eventO...	244	10:17:03 AM
::connect::	11	10:17:03 AM

Time

Why should you care?

Time

Client Side Synchronization

- Headers vs. Content rewrite
- Resource Timing on AJAX

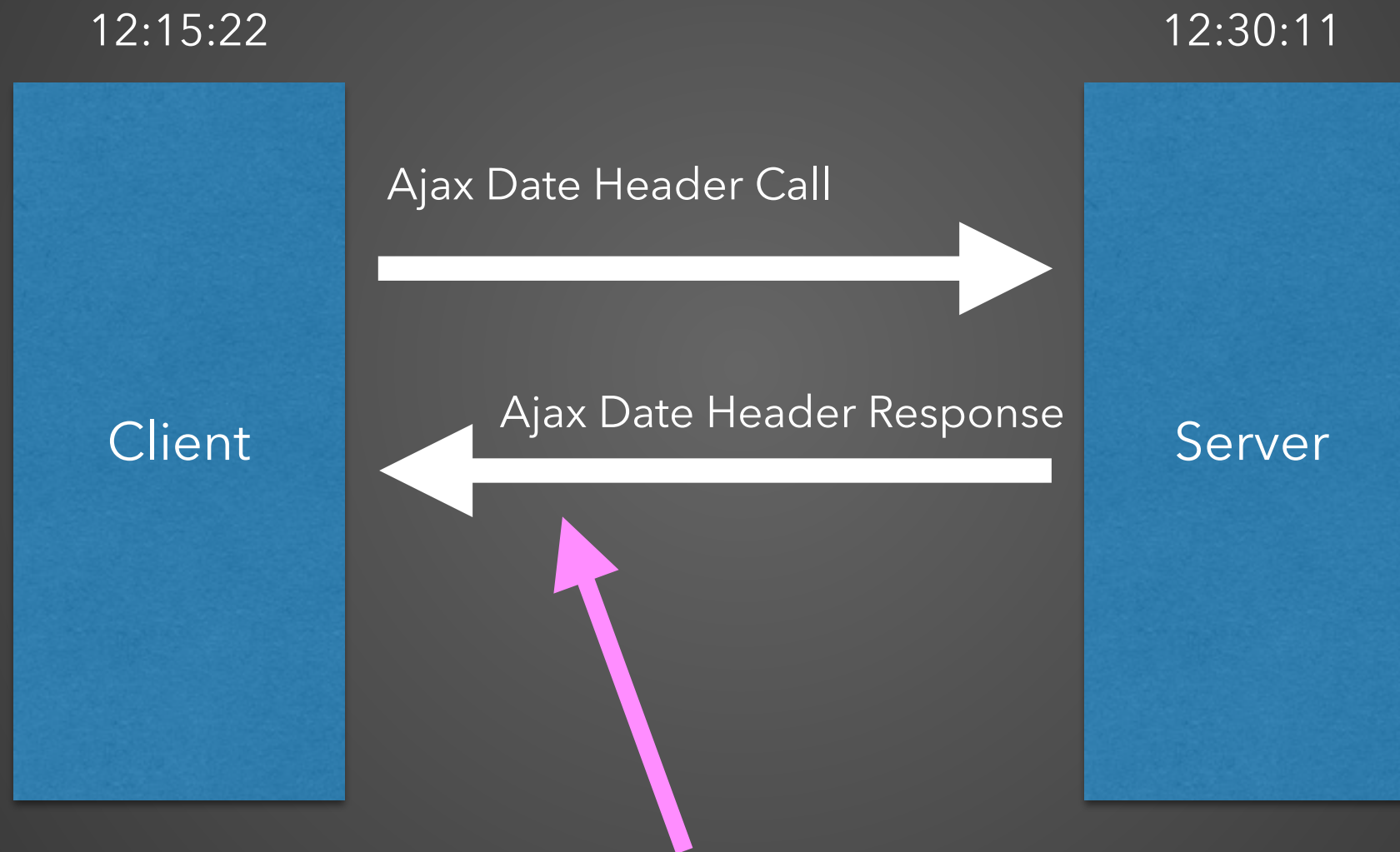
Time

Get the data from an AJAX request in 4 easy steps

```
var serverTimeMillisUTC;  
var oReq = new XMLHttpRequest();  
oReq.onload = function(){  
    var dateStr = oReq.getResponseHeader('Date');  
    serverTimeMillisUTC = new Date(Date.parse(dateStr)).getTime();  
};  
oReq.open("HEAD", "?foo123", false);  
oReq.send();  
  
function getServerTime() {  
  
    for(tvalue in window.performance.getEntriesByType("resource")){  
        tresource = window.performance.getEntriesByType("resource")[tvalue];  
        if(tresource.name.indexOf('foo123') > 0){  
            return serverTimeMillisUTC + (tresource.responseEnd -  
                                           tresource.responseStart);  
        }  
    }  
}
```

Time

Why would I do this?

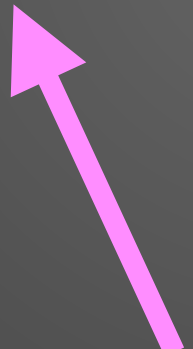


What is response time?

Time

1. Get the date header from the server

```
var oReq = new XMLHttpRequest();  
oReq.onload = function() {  
    var dateStr = oReq.getResponseHeader('Date') ;  
    serverTimeMillisUTC = new Date(Date.parse(dateStr)).getTime();  
};  
oReq.open("HEAD", "/?foo123", false);  
oReq.send();
```



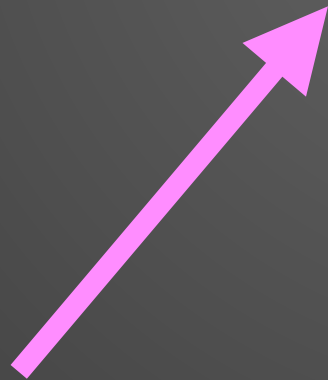
2. User random query param

Time

3. Filter on random query param



```
if (tresource.name.indexOf('foo123') > 0) {  
    return serverTimeMillisUTC +  
        (tresource.responseEnd -  
         tresource.responseStart);  
}
```



4. Calculate Server Time

What Time Is It?

Edge 4 (Data)

Edge 2 (Seek)

Services and Frameworks

Service or Homegrown?

Services

PubNub®

KAAZING >K®



Frameworks



socket.io

Servers

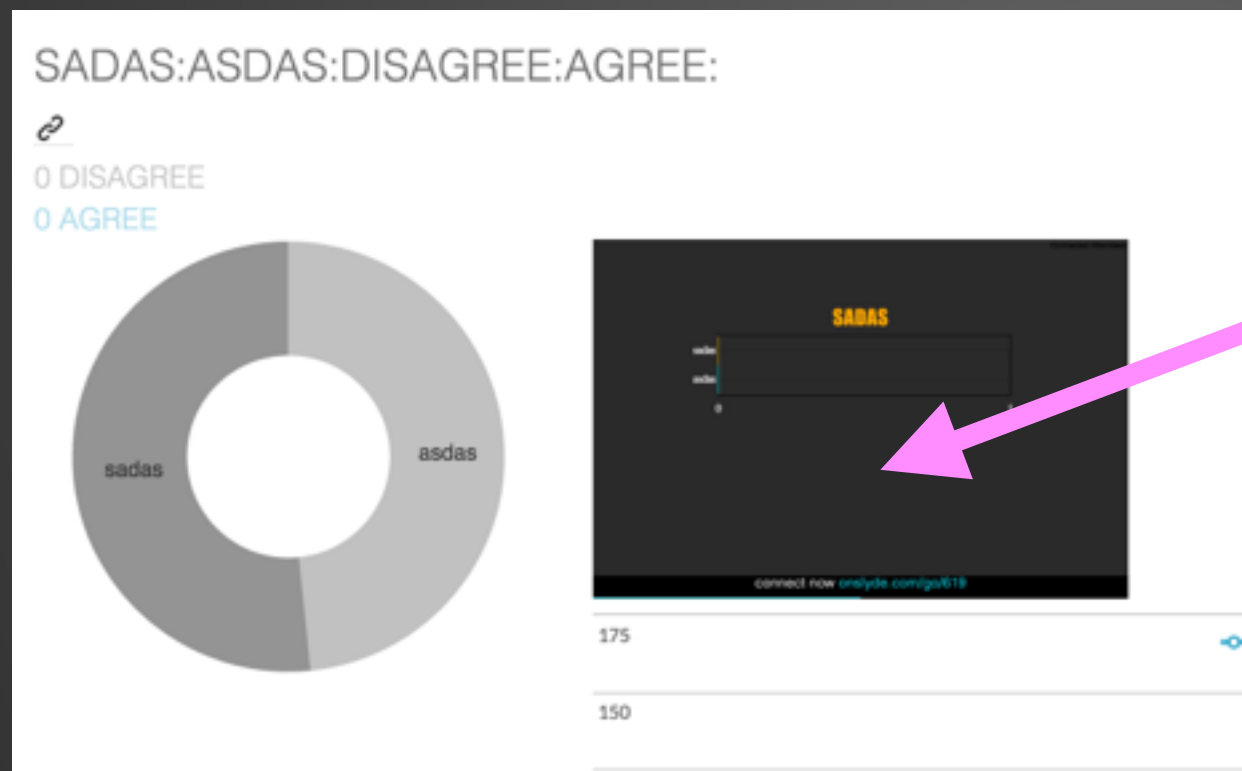


Tips

Head of Line Blocking

Large messages block frame delivery

Latency Sensitive Data? Split up the large messages



Screenshots are sent as binary data over wss:// in onslyde's dashboard

client side code on github

Compression

Extensions are managed by the browser
(spec says it's a "framework")

wss:// connection headers

```
Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits  
Sec-WebSocket-Key: /LqLDugB0+EK9Bnp6d3gqw==  
Sec-WebSocket-Version: 13
```

- 1) Chrome has implicit support (no config on client)
- 2) Compresses any message over 10 bytes

Make Sure You Need It

Do you need WebSocket?

HTTP2, SPDY, SSE, and WebSocket are all different. They could be combined.

Are you willing to polyfill your WebSocket Implementation?

Thanks!

Ben Vinegar
Book author



Jarrold Overson
Book author



Venus.js
where bugs go to die

Seth McLaughlin



Ariya Hidayat



PhantomJS



Wesley Hales
Book author

Michael Ficarra
CoffeeScript, JavaScript,
PLT, TC-39

Appendix

Browser Event Firing

All Events Fire onmessage

```
onmessage: function (m) {  
    var event = (m.data);  
    event = (new Function("return " + event))();  
    event.onslideEvent.fire();  
}
```

Remote Control listens

```
window.addEventListener('remoteMarkup', function(e) {  
    var markup = JSON.parse(e.markup);  
    document.getElementById('from-slide').innerHTML =  
        decodeURIComponent(markup.remoteMarkup);  
}, false);
```


HTTP/2

Demo 1

Demo 2