HTML5 Websockets with Rails and Pusher

Mark Johnson
mark@menumill.com
@logicspin

who?

Mark Johnson
kailua, 1 wife, 2 kids, 2 dogs
ski, surf, travel

developing rails apps since 2005
co-founder at Menumill.com

websocket examples

see: http://pusher.com/examples

demo:

- chat: http://pusher-chat.heroku.com
- gaug.es screencast: http://www.screenr.com/xqo

websocket - what?

WebSocket is a technology providing for:

- bi-directional,
- full-duplex communications channels,
- over a single Transmission Control Protocol (TCP) socket.

HTTP-compatible handshake - default HTTP and HTTPS ports (80 and 443)

The WebSocket protocol defines a ws:// and wss:// prefix to indicate a WebSocket and a WebSocket Secure connection, respectively

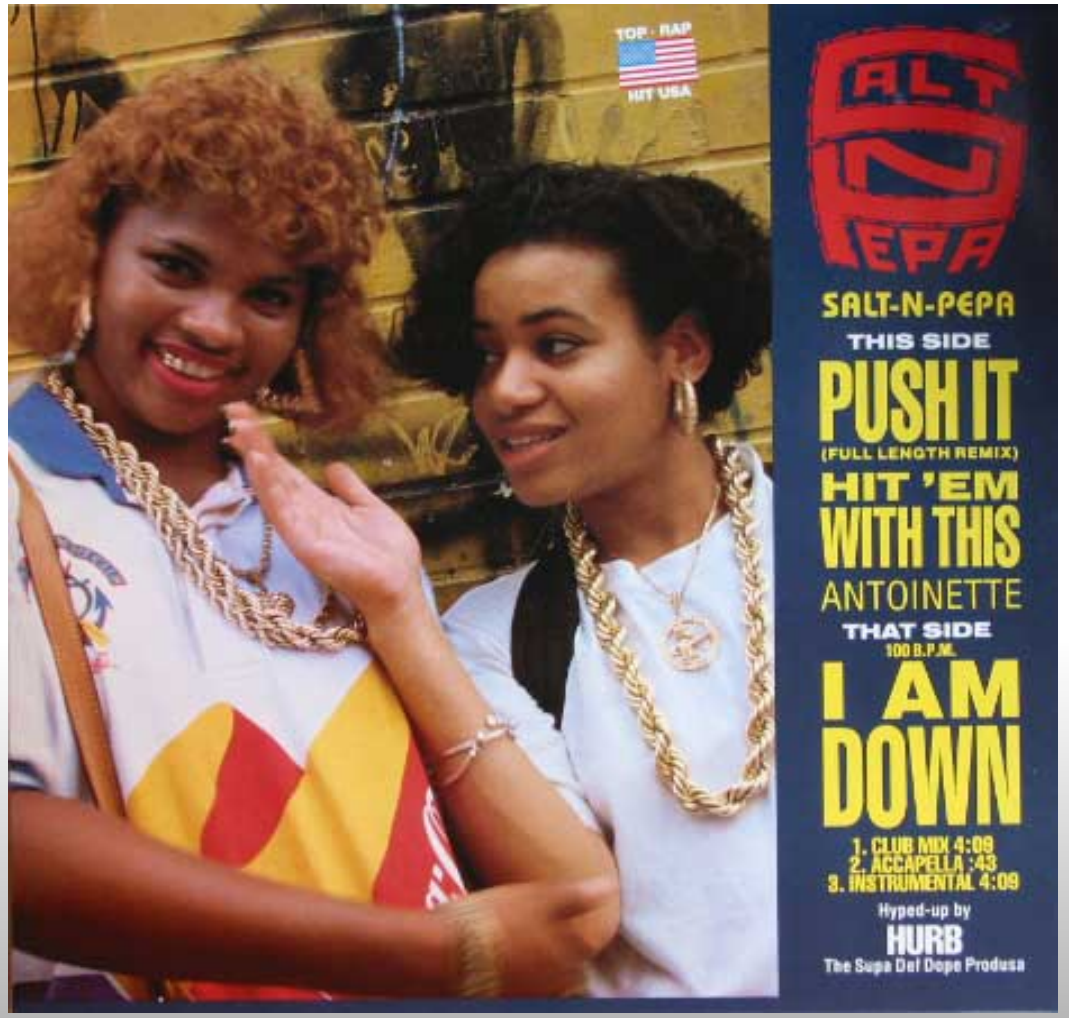
source: *http://en.wikipedia.org/wiki/WebSockets*

why?

HTML5 Web Sockets can provide:

- up to 1000:1 reduction in unnecessary HTTP header traffic
- 3:1 reduction in latency

scale!
avoid polling!
save $!
get real(time)



source: http://websocket.org/quantum.html

polling

Use case A:
1,000 clients polling every second: Network throughput is (871 x 1,000) = 871,000 bytes
= 6,968,000 bits per second (6.6 Mbps)

Use case B:
10,000 clients polling every second: Network throughput is (871 x 10,000) = 8,710,000 bytes
= 69,680,000 bits per second (66 Mbps)

Use case C:
100,000 clients polling every 1 second: Network throughput is (871 x 100,000) = 87,100,000 bytes
= 696,800,000 bits per second (665 Mbps)

source: http://websocket.org/quantum.html

websockets

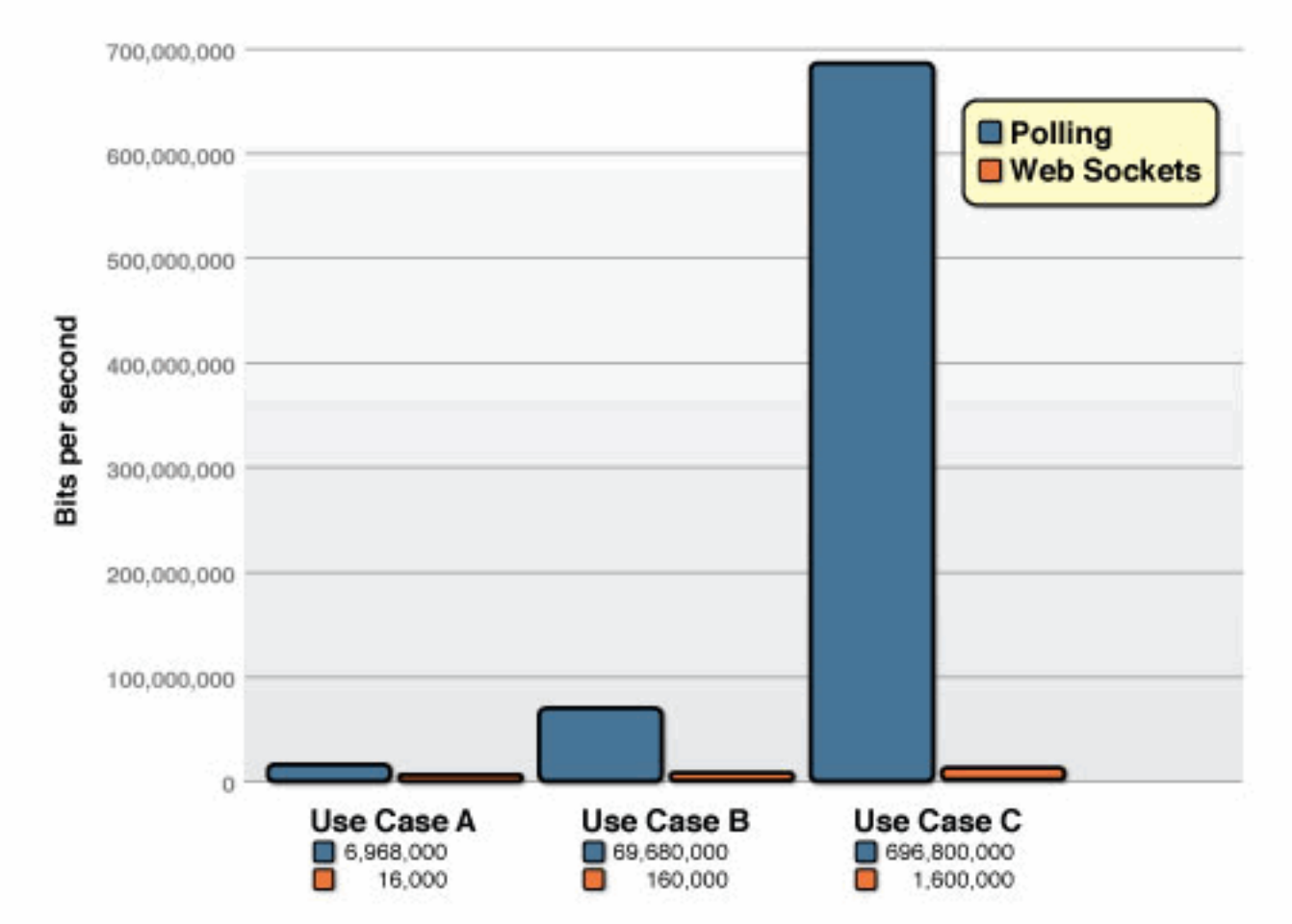
Use case A:
1,000 clients receive 1 message per second: Network throughput is (2 x 1,000) = 2,000 bytes
= 16,000 bits per second (0.015 Mbps)

Use case B:
10,000 clients receive 1 message per second: Network throughput is (2 x 10,000)
= 20,000 bytes = 160,000 bits per second (0.153 Kbps)

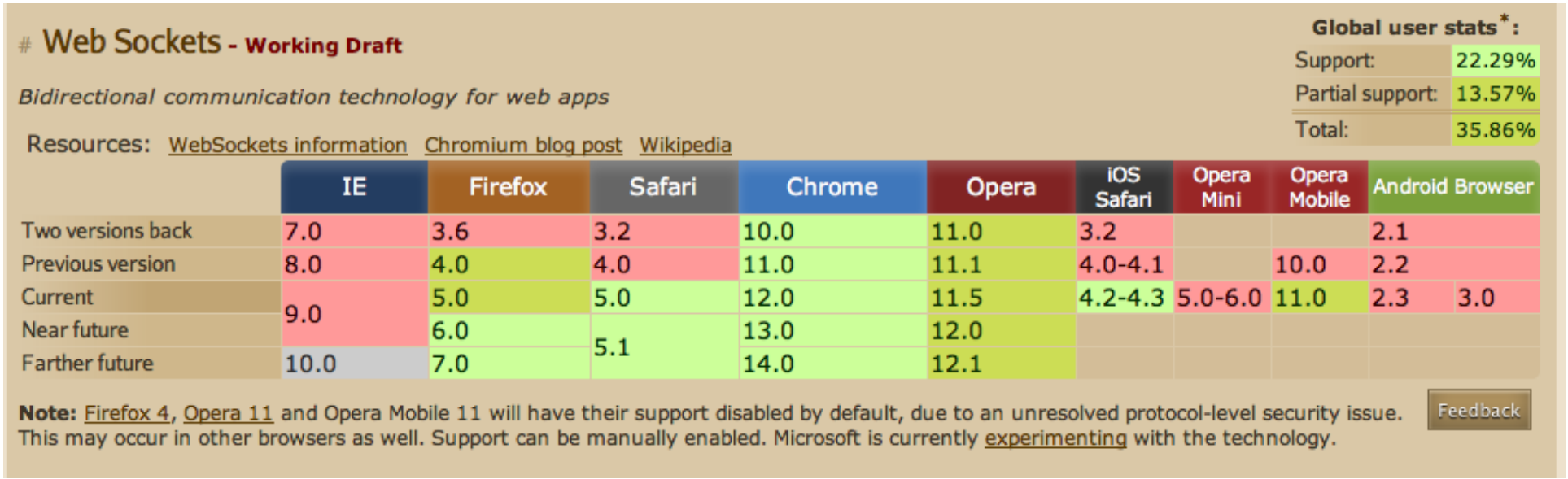
Use case C:
100,000 clients receive 1 message per second: Network throughput is (2 x 100,000) = 200,000 bytes
= 1,600,000 bits per second (1.526 Kbps)

source: http://websocket.org/quantum.html

any questions?



browser support / drawbacks



Web Sockets - Working Draft

Bidirectional communication technology for web apps

Resources: WebSockets information | Chromium blog post | Wikipedia

Global user stats*:

Support:	22.29%
Partial support:	13.57%
Total:	35.86%

	IE	Firefox	Safari	Chrome	Opera	iOS Safari	Opera Mini	Opera Mobile	Android Browser	
Two versions back	7.0	3.6	3.2	10.0	11.0	3.2			2.1	
Previous version	8.0	4.0	4.0	11.0	11.1	4.0-4.1		10.0	2.2	
Current	9.0	5.0	5.0	12.0	11.5	4.2-4.3	5.0-6.0	11.0	2.3	3.0
Near future		6.0	5.1	13.0	12.0					
Farther future	10.0	7.0		14.0	12.1					

Note: Firefox 4, Opera 11 and Opera Mobile 11 will have their support disabled by default, due to an unresolved protocol-level security issue. This may occur in other browsers as well. Support can be manually enabled. Microsoft is currently experimenting with the technology.

Feedback

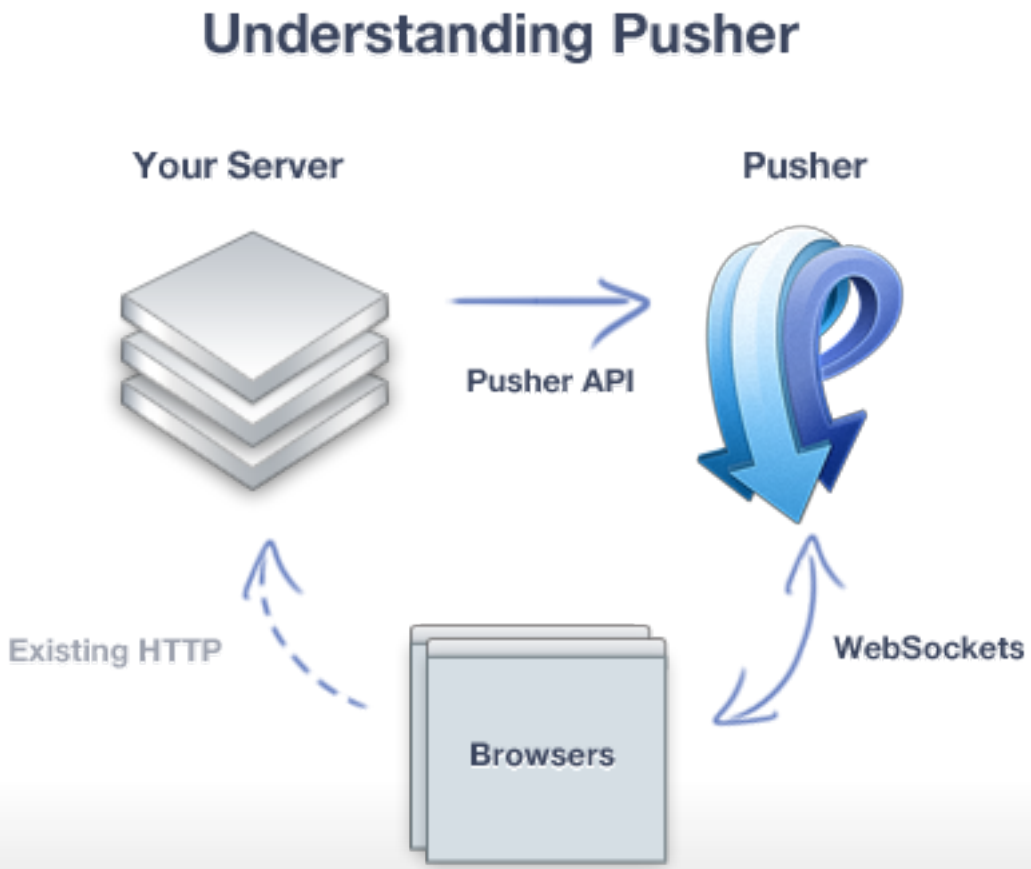
from: http://caniuse.com/#search=websocket

more: http://pusher.com/docs/browser_compatibility

enter pusher

Pusher is a hosted API for quickly, easily and securely adding scalable realtime functionality via WebSockets to web and mobile apps.

how pusher works



pusher API

Client Libraries - js, ruby, objective-c, .NET, etc
REST API
Client API
Publisher API

Client API

Overview

- Connection
- Channels
- Events
- Presence Events

Connection

Pusher connection is bi-directional and can send and receive messages from the server.

```
var pusher = new Pusher(applicationKey, options);
```

Channels

Each application has a number of channels, and every client can choose which channels it connects to.

var channel = pusher.subscribe(channelName);

channel types:

- public
- private
- presence

Events

Events are the primary method of packaging messages in the Pusher system. They form the basis of all communication.

```
var pusher = new Pusher('API_KEY');
var channel = pusher.subscribe('APPL');
channel.bind('new-price',
  function(data) {
    // add new price into the APPL widget
  }
);
```

Presence events

Presence channels have a number of pre-defined events that can be bound to in order to notify a connected client about users joining or leaving the channel.

- pusher:subscription_succeeded
- pusher:member_added
- pusher:member_removed

Publisher API

These are run on your server, and generally interact with the Pusher REST API.

```
require 'pusher'
Pusher.app_id = 'APP_ID'
Pusher.key = 'API_KEY'
Pusher.secret = 'SECRET_KEY'

class ThingsController < ApplicationController
  def create
    @thing = Thing.new(params[:thing])

    if @thing.save
      Pusher['things'].trigger('thing-create', @thing.attributes)
    end
  end
end
```

lets build one!

demo - build Rails 3.1 app with pusher enabled admin features

we likey

menumill.com demo

use case: digital signage / digital menu boards

other options

- socket.io
- pubnub - handles more messages daily than twitter
- faye - http://faye.jcoglan.com/

Questions/Thanks!

mark@menumill.com
@logicspin