Real-time web

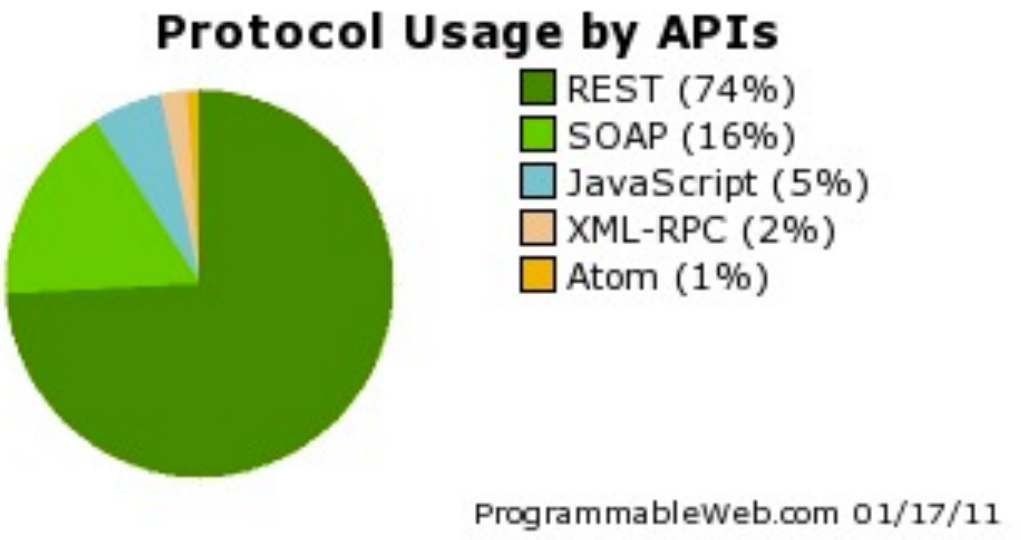
Cadec 2011

Niklas Gustavsson | niklas.gustavsson@callistaenterprise.se | @protocol7

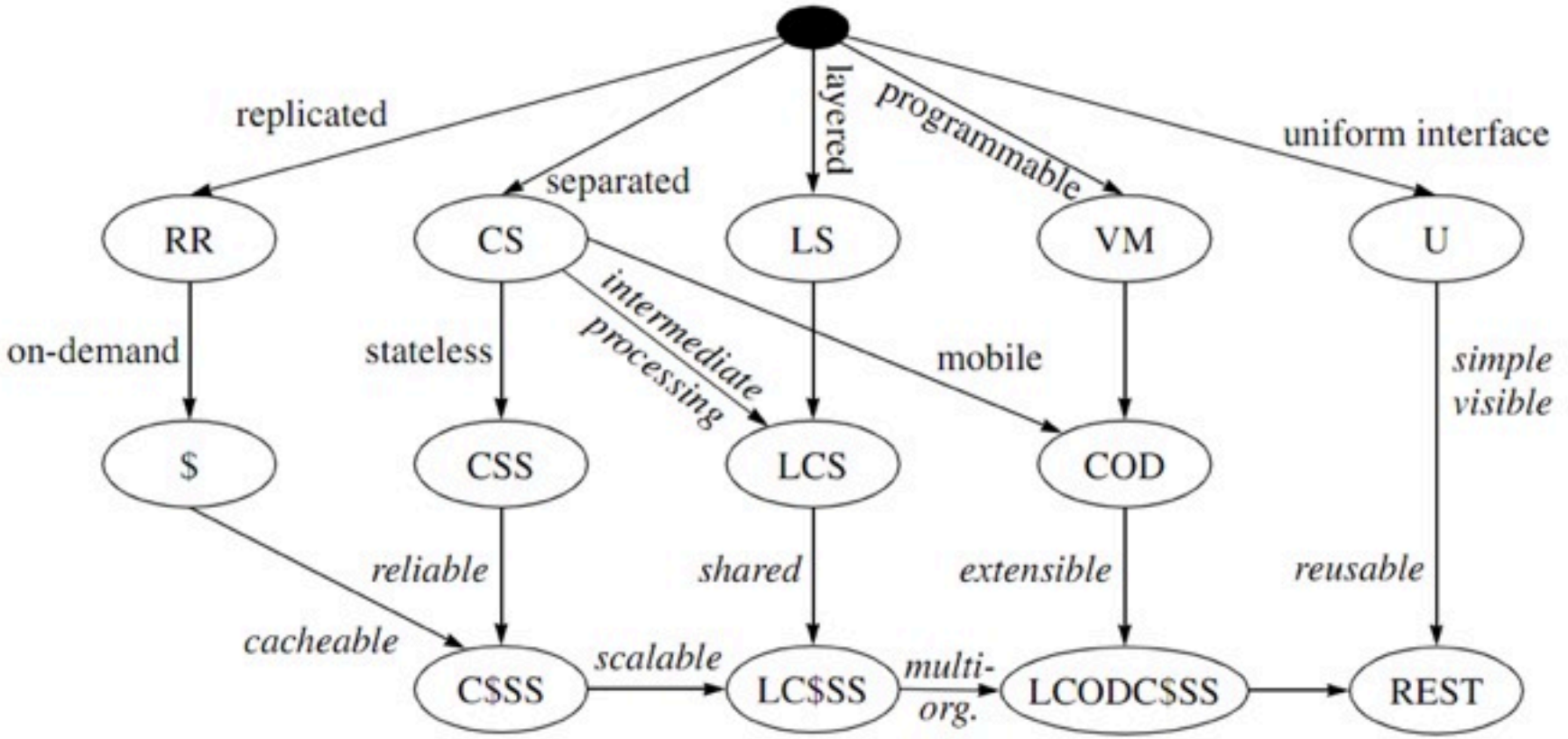
CALLISTA
Enterprise

REST won

(at least outside the firewall)

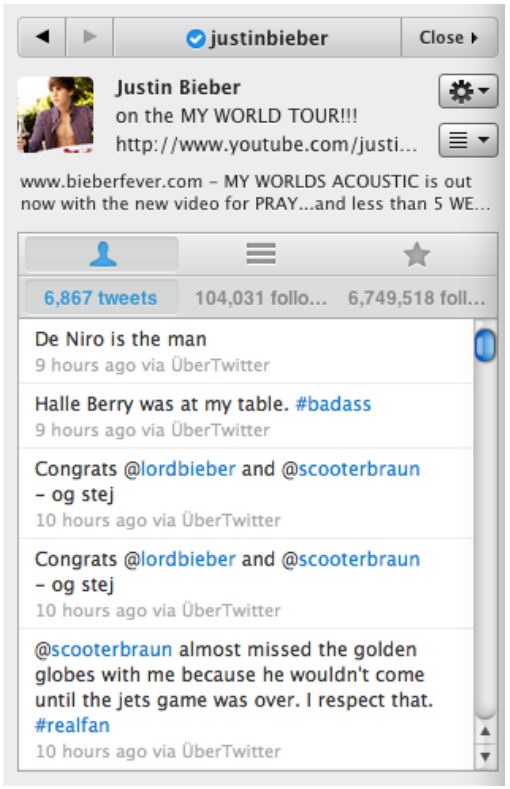


Out of 2724 public APIs

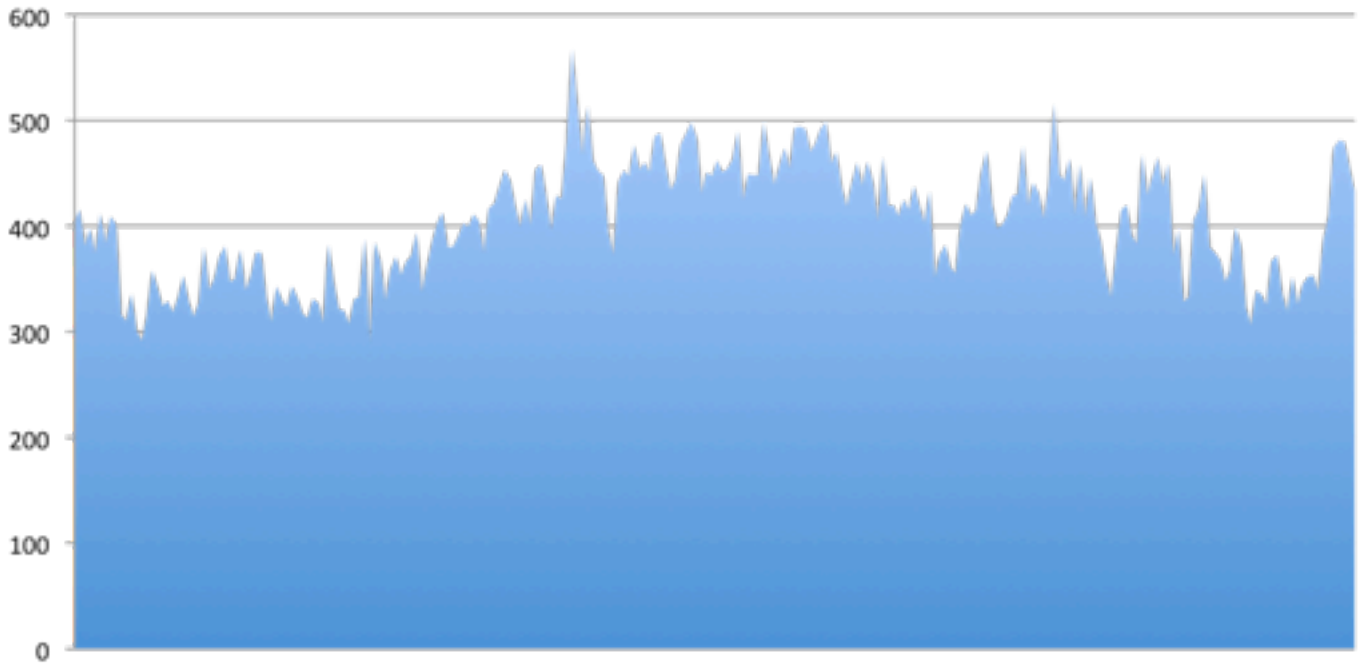


Client-server
Short lived requests

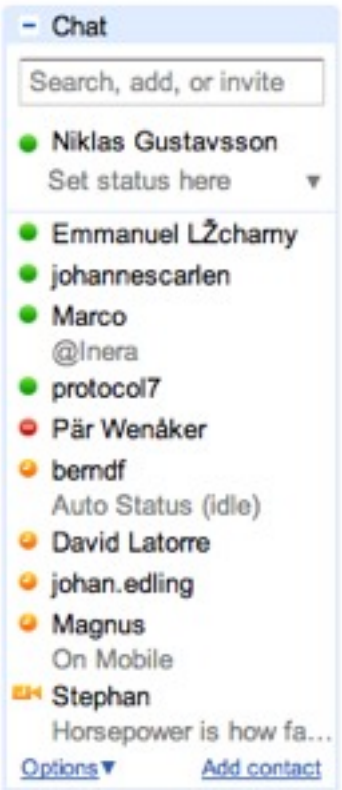
The Justin Bieber use case



Polling (sometimes) sucks



Or, it really sucks

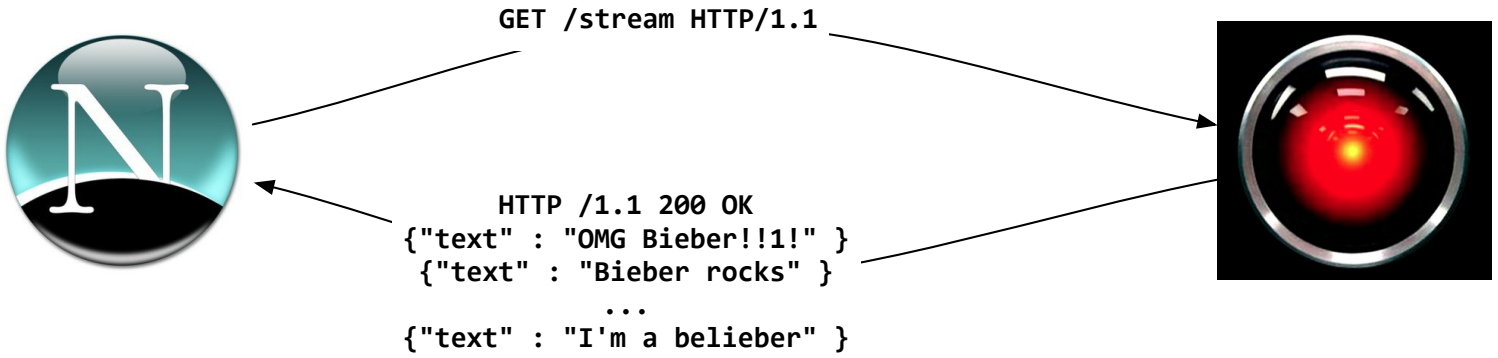


Enter real time web

What about messaging, WS-Notification and all that great stuff?

Streaming

(Comet, Twitter, websockets)



Websockets

Stream Content

```
GET /pubsub-1.0-SNAPSHOT/ws?topic=http%3A%2F%2Fstream.twitter.com%2F1%2Fstatuses%2Ffilter.json%23bieber HTTP/1.1
Upgrade: WebSocket
Connection: Upgrade
Host: localhost:8080
Origin: http://localhost:8080
Sec-WebSocket-Key1: 8X Q Nn0 FOu4@55 9    4 X5 0 d-
Sec-WebSocket-Key2: 3 =1k8y5981858

.......`HTTP/1.1 101 WebSocket Protocol Handshake
Upgrade: WebSocket
Connection: Upgrade
Sec-WebSocket-Origin: http://localhost:8080
Sec-WebSocket-Location: ws://localhost:8080/pubsub-1.0-SNAPSHOT/ws?topic=http%3A%2F%2Fstream.twitter.com%2F1%2Fstatuses%
2Ffilter.json%23bieber

nu.....E.{..L.![
```

Transparent Proxies: Threat or Menace?

Lin-Shung Huang, Eric Y. Chen, Adam Barth, Eric Rescorla, and Collin Jackson

Abstract—**Browsers limit how web sites can access the network. Historically, the web platform has limited web sites to HTTP, but HTTP is inefficient for a number of applications—including chat and multiplayer games—for which raw socket access is more appropriate. Java, Flash Player, and HTML5 provide socket APIs to web sites, but we discover and experimentally verify attacks that exploit the interaction between these APIs and transparent proxies. Our attacks poison the proxy's cache, causing all clients of the proxy to receive malicious content supplied by the attacker. We then propose a revised version of the HTML5 WebSocket handshake that resists these (and other) attacks.**

I. Introduction

Browsers restrict how web applications can interact with the network by enforcing a number of security invariants on browsers, the protocols themselves have seen only modest amounts of security analysis. Recently, these protocols were shown to be vulnerable to DNS rebinding attacks [1], whereby the consent was scoped to a host name rather than an IP address, letting the attacker transfer his or her consent to another network endpoint.

We show that both of these consent protocols are vulnerable to attack in some network configurations. In particular, consider a network scenario in which the user connects to the Internet via a transparent proxy and a firewall, as is common in enterprise networks. The transparent proxy intercepts outbound HTTP requests, perhaps to monitor employee network access or to enforce a security policy, and the firewall prevents miscreants on the Internet from accessing the internal network.

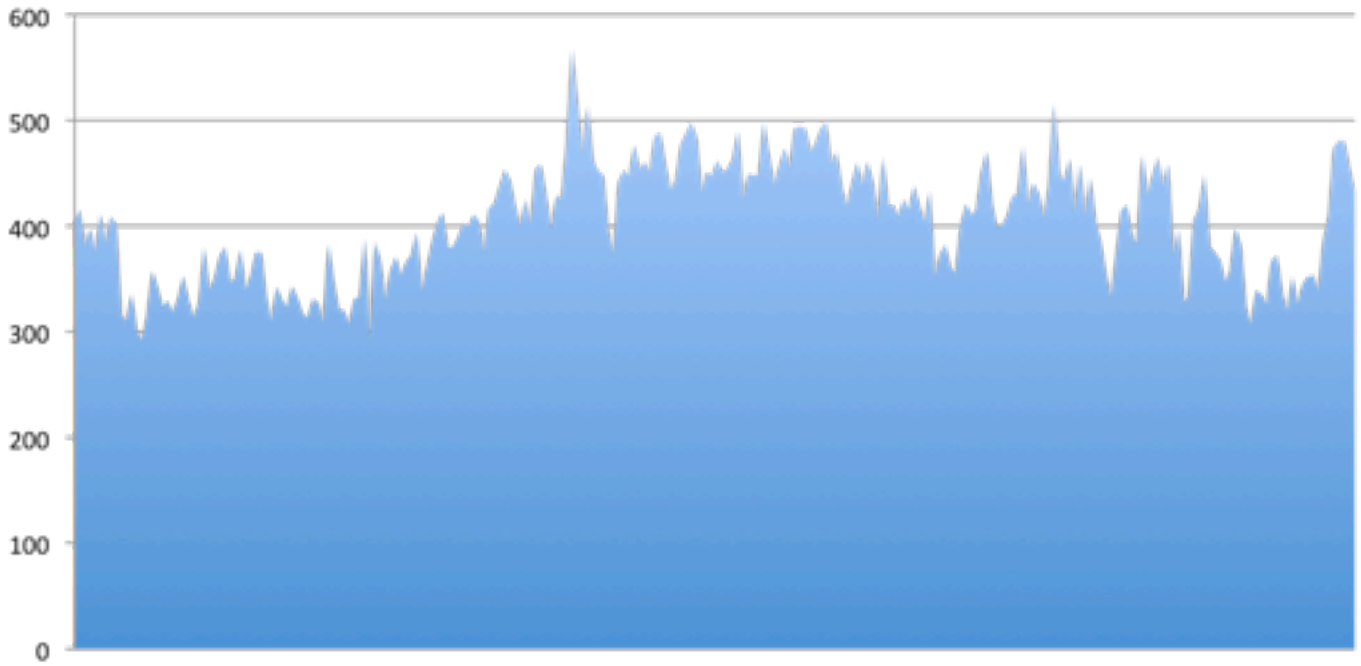
http://www.adambarth.com/experimental/websocket.pdf

Webhooks

The Hollywood principle

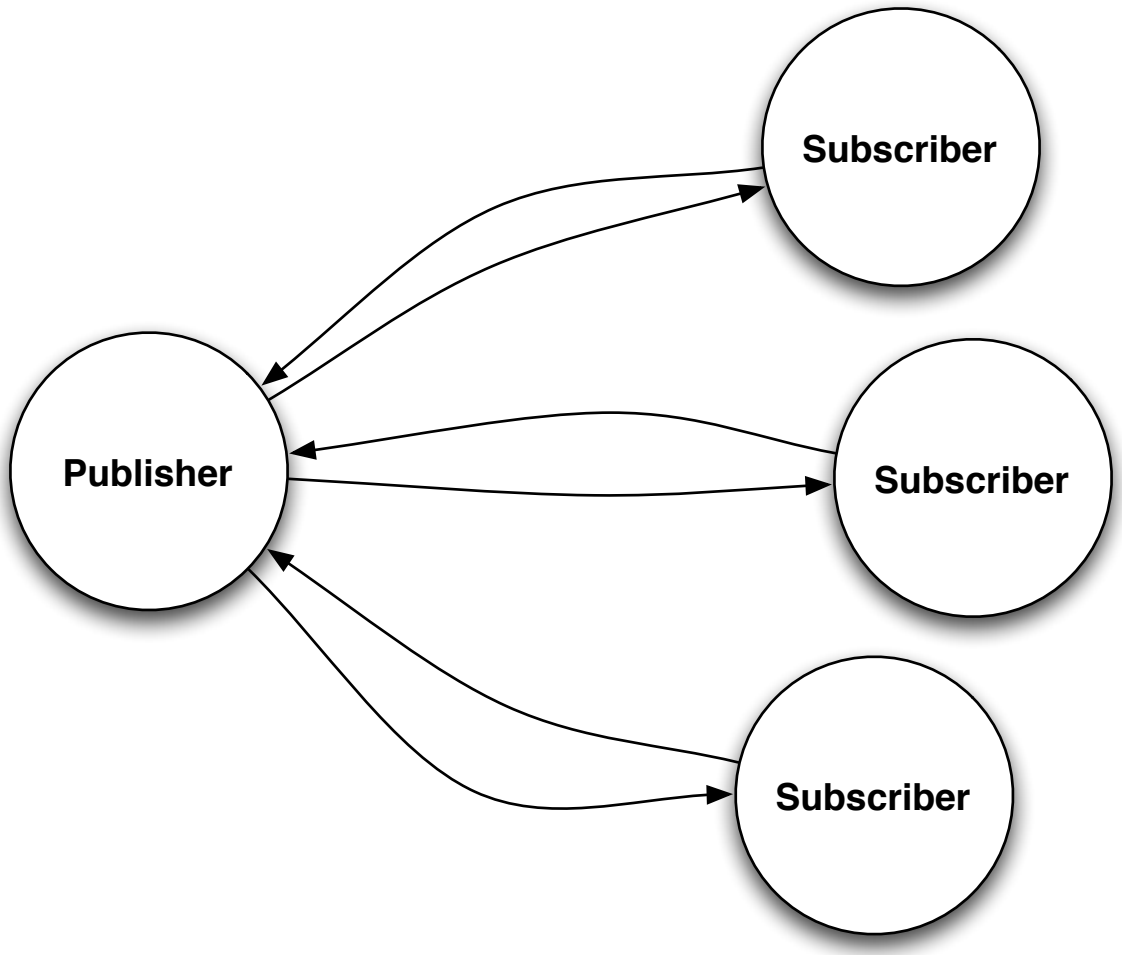
```
POST /questions HTTP/1.1
{ "question" : "Meaning of life...",
"callback" : "http://whitemicehq.com/answers" }
```

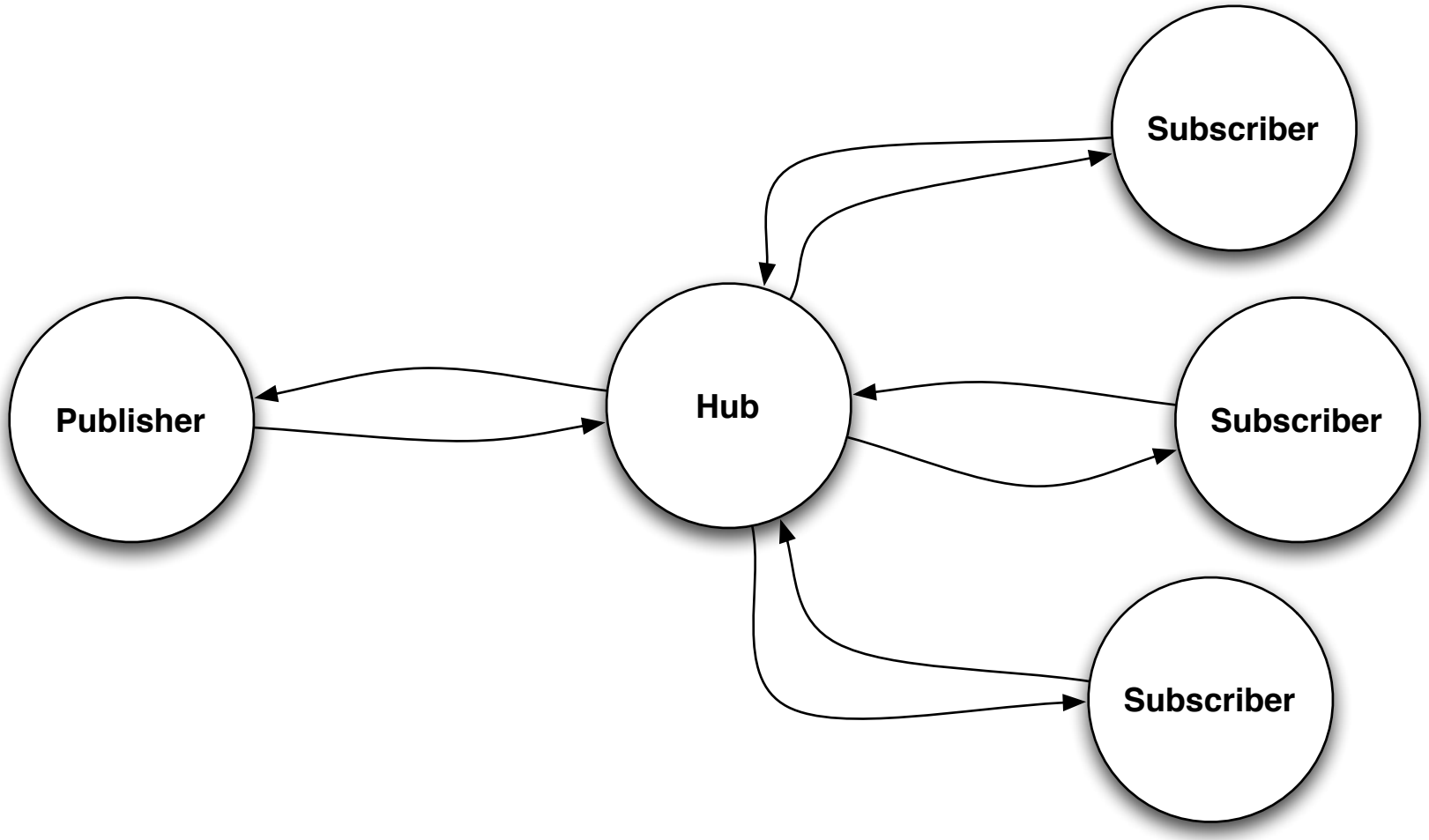
```
POST /answers HTTP/1.1
{ "answer" : 42 }
```

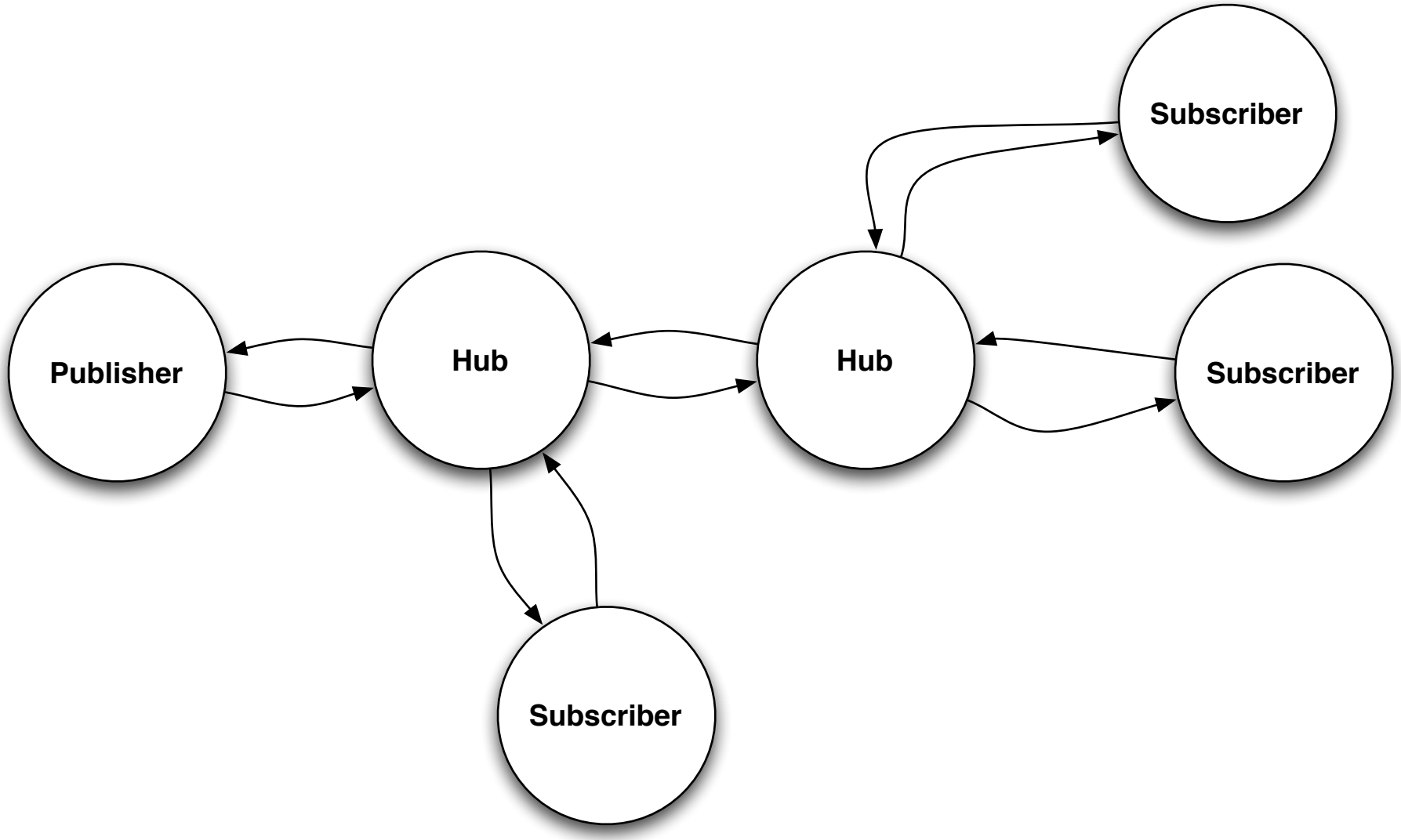


PubSubHubbub

PubSubHubbub



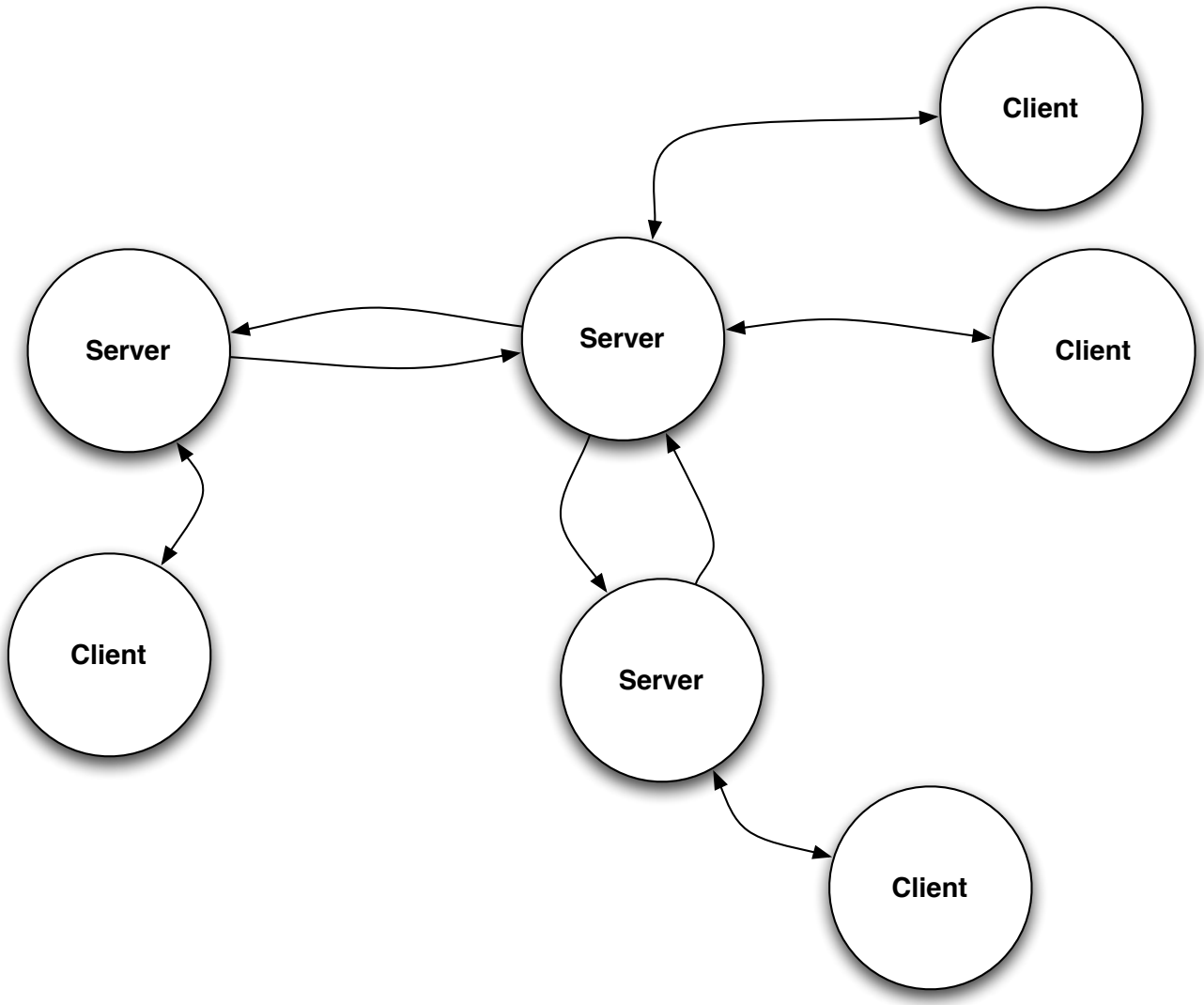


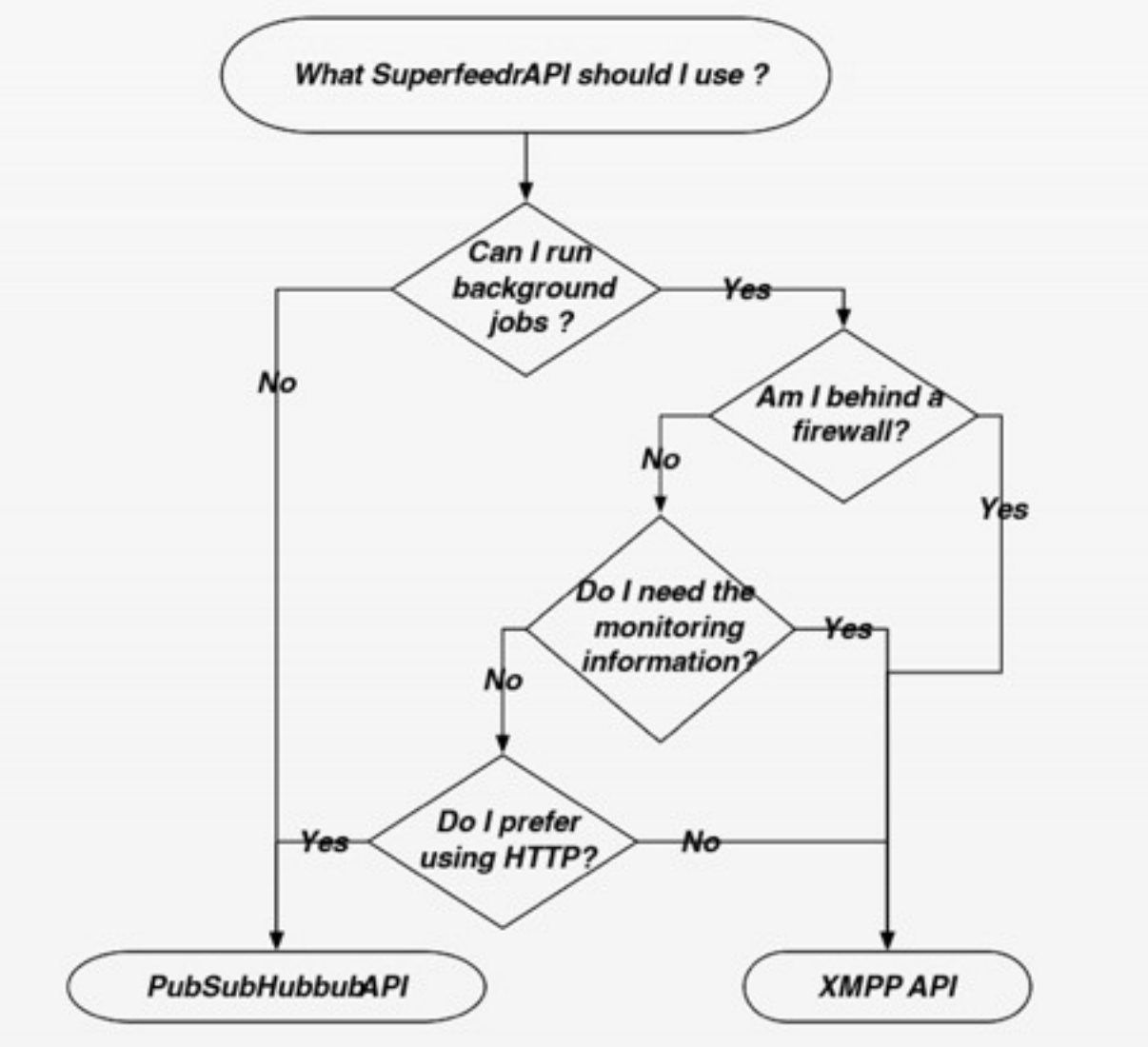


Hub <link>

XMPP

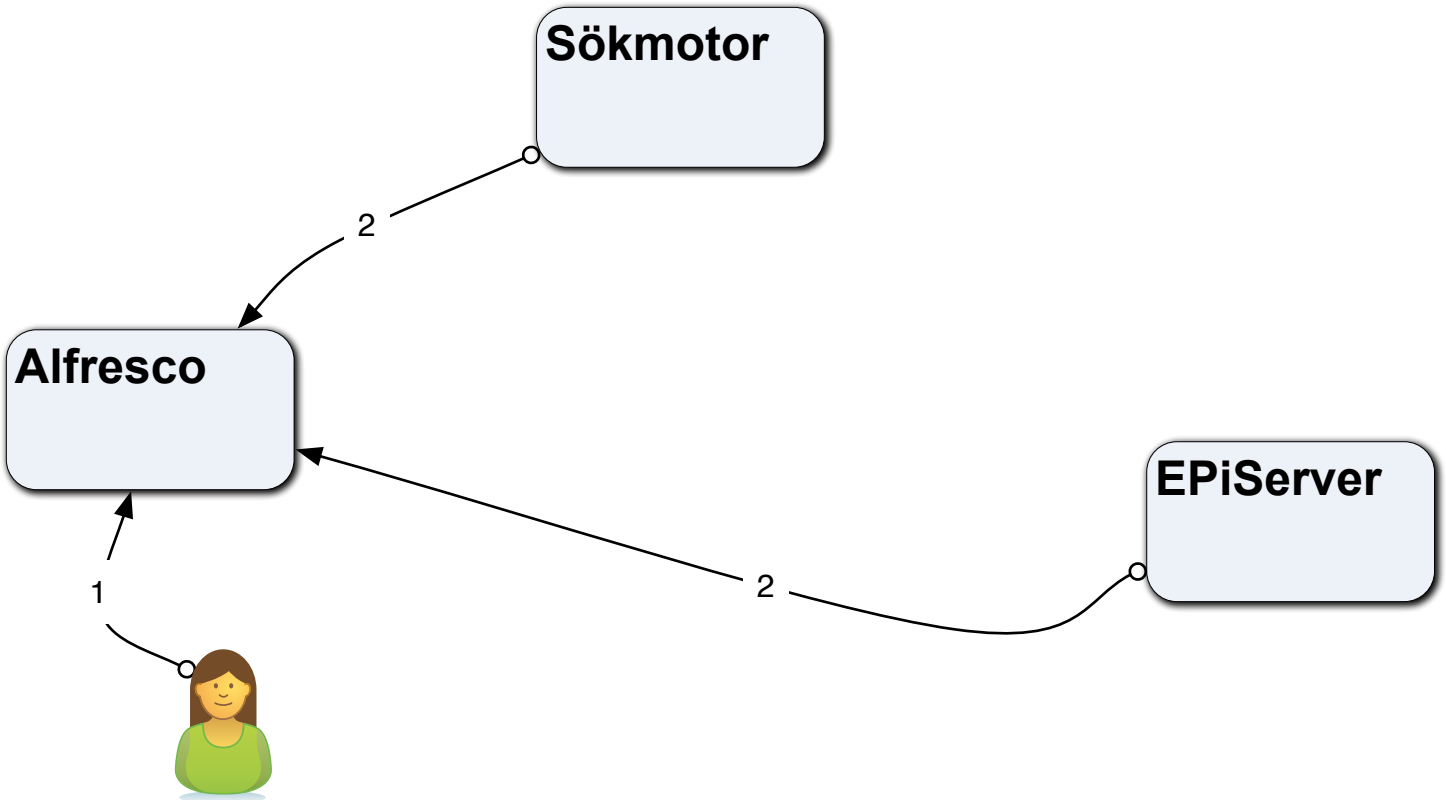
(aka Jabber)

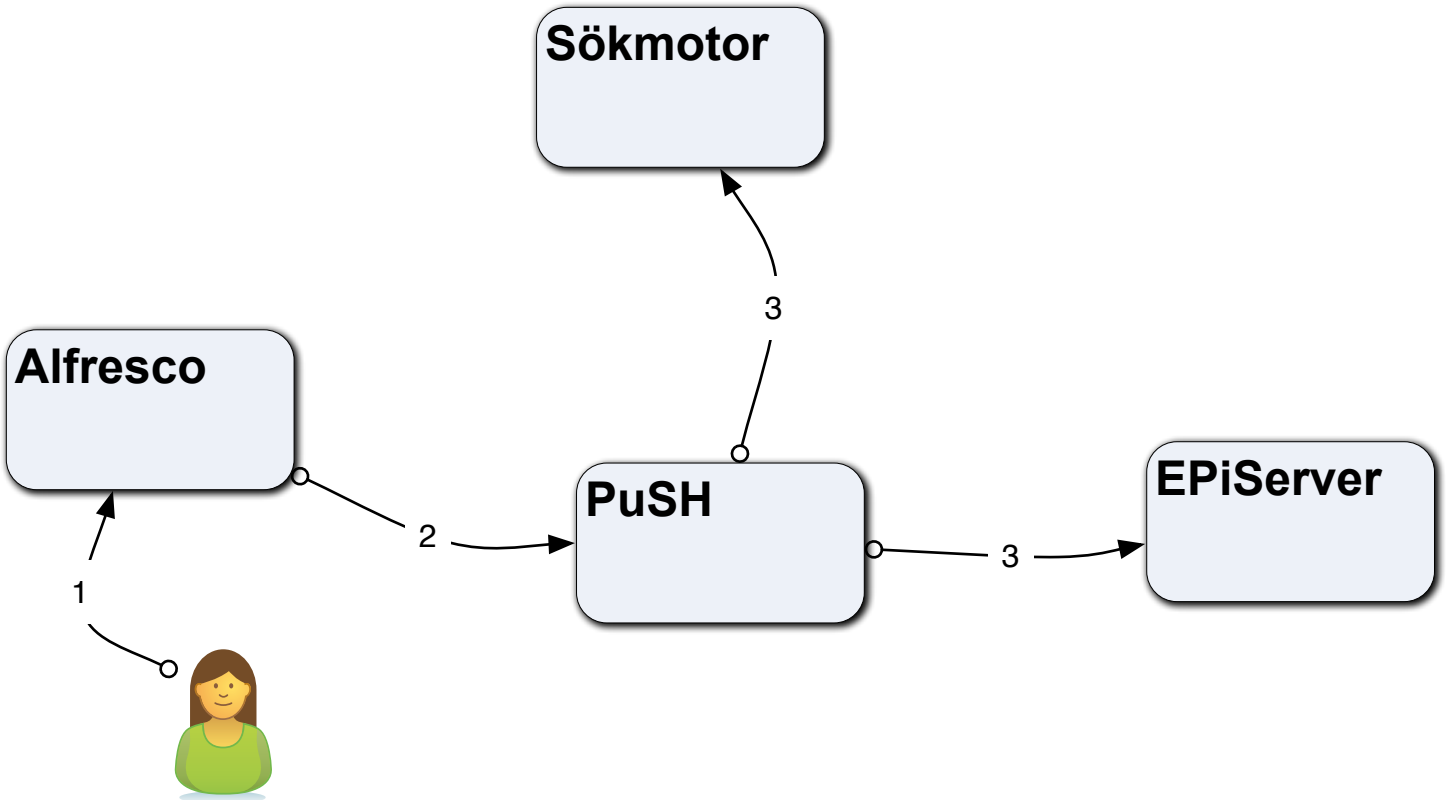


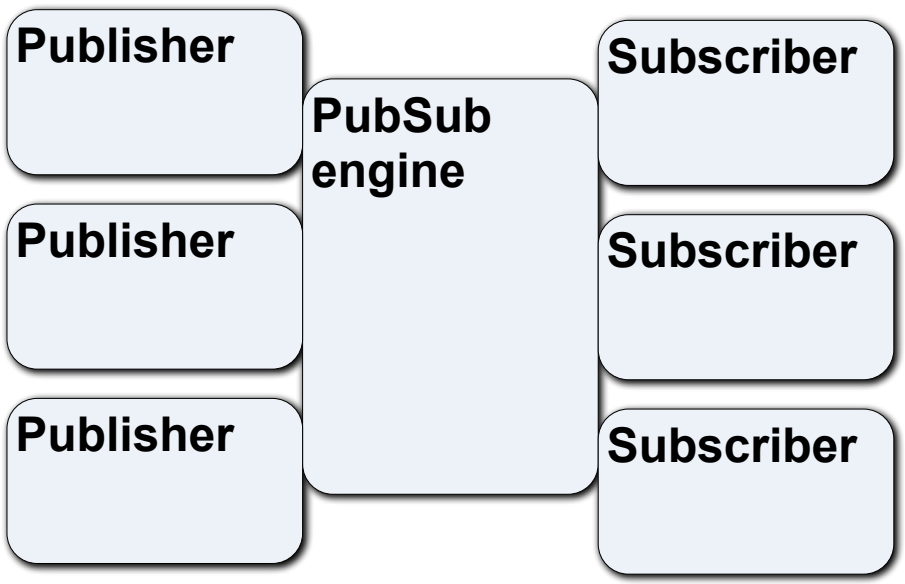


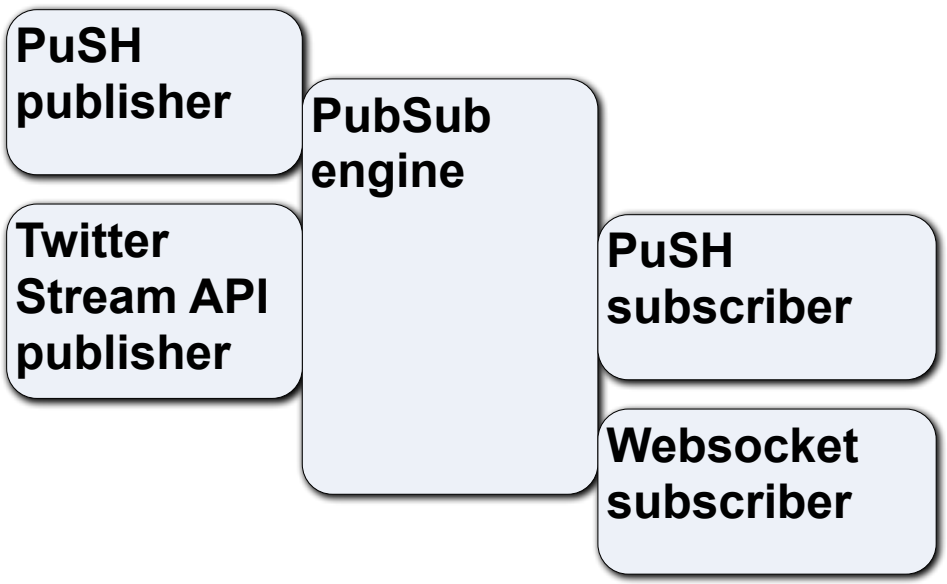
http://superfeedr.com/documentation

Case: distributing content









Open source

http://code.google.com/p/oppna-program-pubsub-service

Demo

The players

#cadec

http://dev.twitter.com/pages/streaming_api

http://wiki.webhooks.org/

More?

http://xmpp.org/

http://s4.io/

http://code.google.com/p/pubsubhubbub/

?