



HTML5 WebRTC and SIP Over WebSockets

Thomas Quintana - TeleStax, Inc
April 2013, SIPNOC

HTML5

WebRTC

State of the Art

What is WebRTC ?

- Web Real Time Communications (WebRTC) is aimed to enable Real Time Communications among web browsers in a peer to peer fashion.
 - No License or any other Fees
 - Standard Web APIs Interoperable between browsers
 - no proprietary plugins (like Flash)
 - no downloads or installation (Skype)
 - Cross Platform (mobile + desktop + TV + ...) through HTML5

WebRTC Possibilities

- Add communications as a feature of web applications
- Value Add for operators.
- Current WebRTC Innovative startups or operators
- 3 WebRTC Acquisitions in the past year


WebRTC Uptake

- Major Players Overview :
 - Google, Mozilla, Opera, Microsoft, Apple
- Codec War : VP8 vs H264
- Nascent Fast Rising Technology
- Expand on OTT : Increase Threat vs Help Operators to fight OTT.

HTML5 WebRTC

Client Side

Building HTML5 Web SIP Phone

Mobicents HTML5 WebRTC Client, By 

REGISTRATION

Display Name:

Domain:

User Name:

SECURITY

ADVANCED SETTINGS

[Register](#)

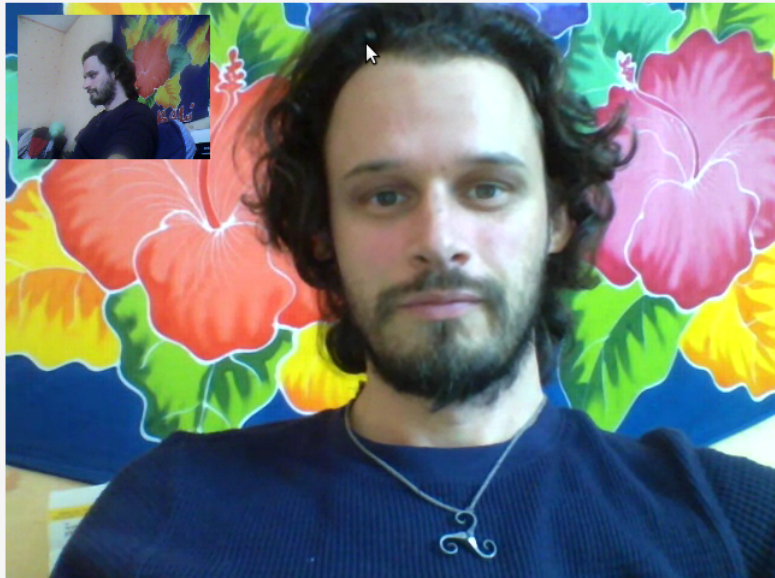
[UnRegister](#)

COMMUNICATE

Contact To Call:

[Call](#)

[Bye](#)

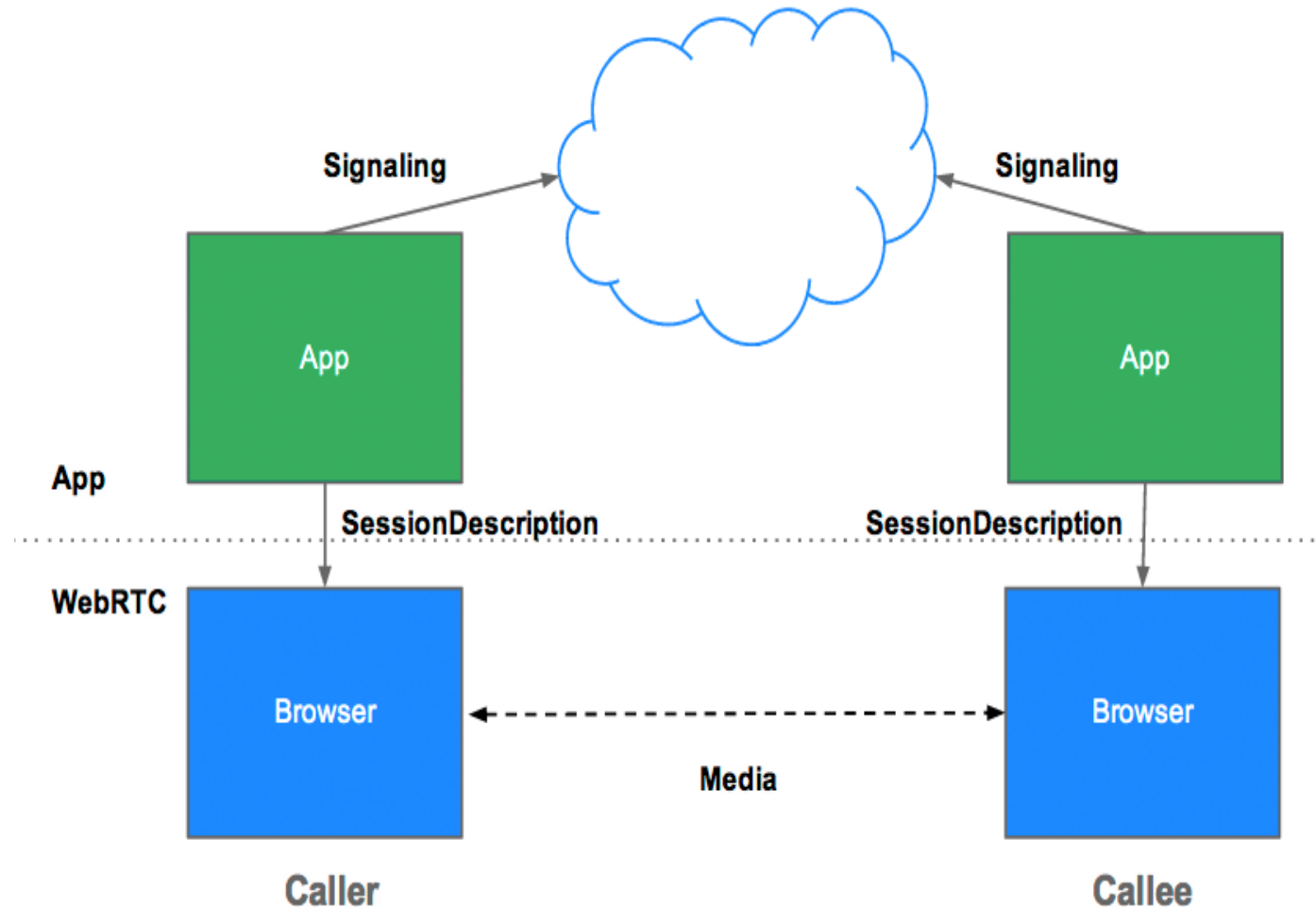


Double-Click the video to enter Full Screen mode

HTML5 WebRTC

Signaling and Media

- WebRTC is independent of WebSockets
- Can use anything for signalling including Ajax, server push or plain HTTP
- Media is peer to peer and can handle both audio and video



Handling NAT

- For SIP Over WebSockets since SIP is over the traditional TCP Sockets, it is not affected by NAT as pure SIP is.
- Contact has .invalid IP Address so Browser Application would not be reachable from the core SIP network. Usage of SIP Outbound enables the TCP Connection Flow to be stored at REGISTRAR and reused for inbound SIP traffic.
- The RTP is the most important NAT problem, but it is browser responsibility to fix this
 - STUN/ICE is a built-in and mandatory
 - Chrome to Chrome interop is guaranteed, Chrome Firefox interop is now working.

Security

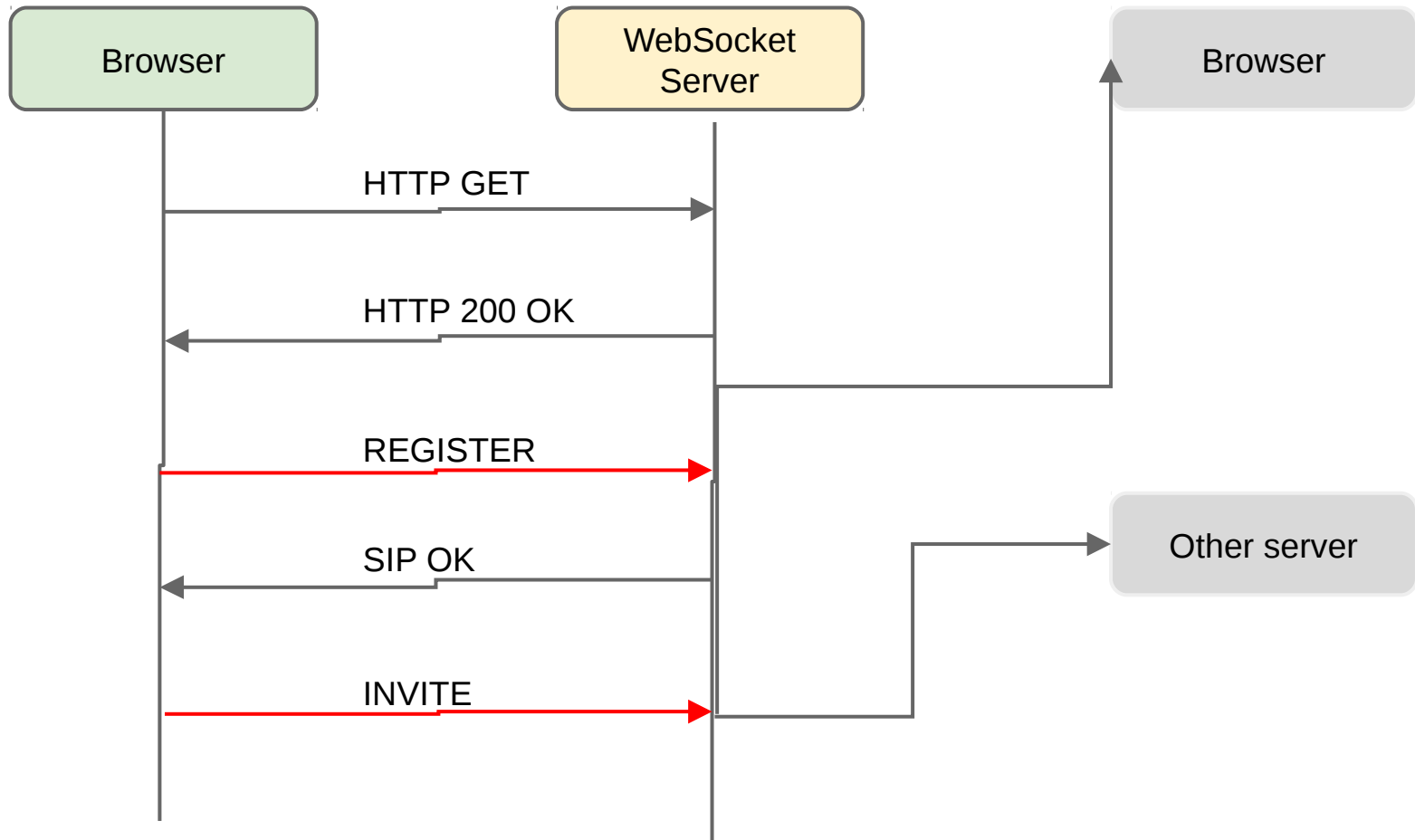
- Media is Secure RTP, DTLS
- WebRTC is not a plugin thus runs in the browser sandbox
- Initial HTTP Connection can be HTTPS.
- Can have federated identity between Web and SIP
- Reuse existing SIP and Web Security paradigms

HTML5 WebRTC

Server Side

SIP Over WebSockets

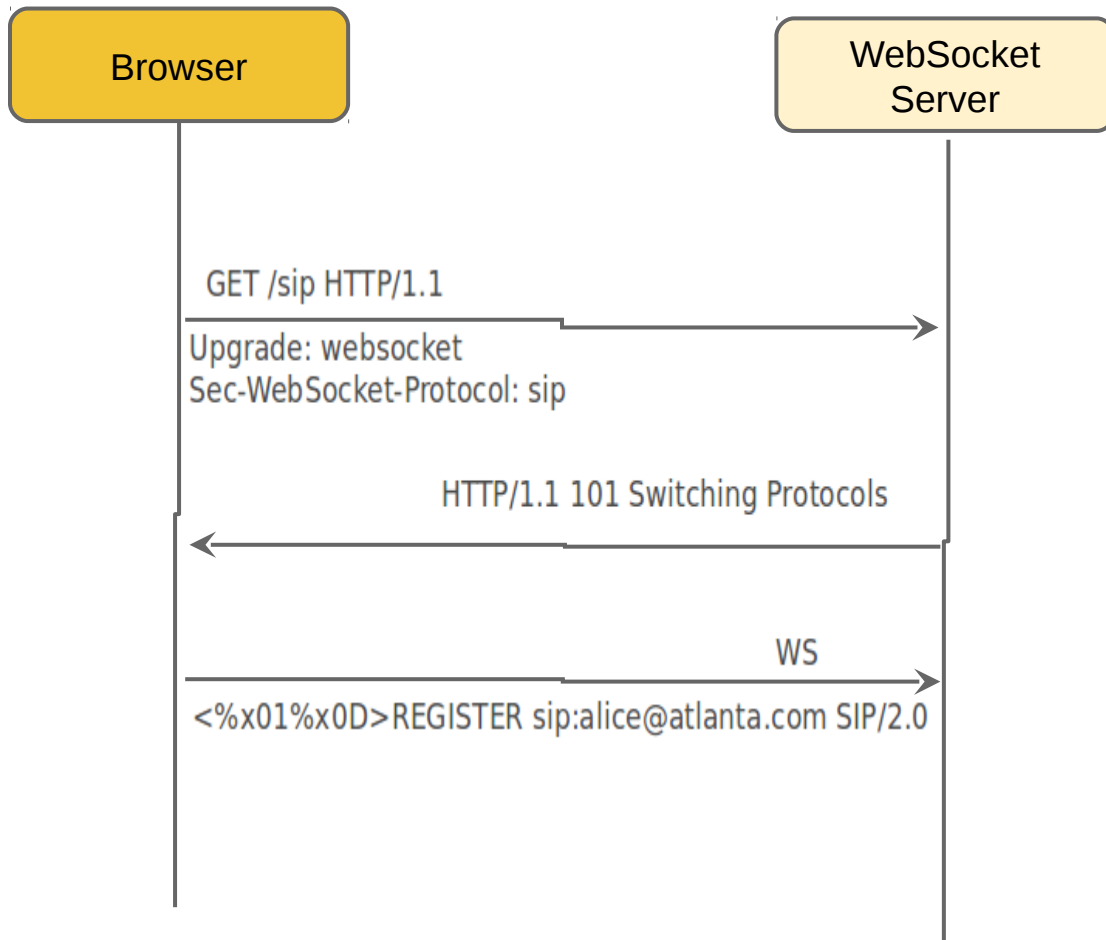
Typical Flow



SIP Over WebSockets

Flow Detailed

<http://tools.ietf.org/html/draft-ietf-sipcore-sip-websocket-08> : Still a draft



- Regular HTTP request with Upgrade header
- Switch to normal mode
 - No HTTP any more, just plain subprotocol
 - ..except it's masked so plaintext can't be misinterpreted and avoid security issues
- SIP Messages carried in WebSocket Frames
- New SIP Transports : WS or WSS (for Secure using TLS)
 - Addresses advertised by browsers are invalid => literally "df7ja123ls0d.invalid"
 - Via, Contact, everything

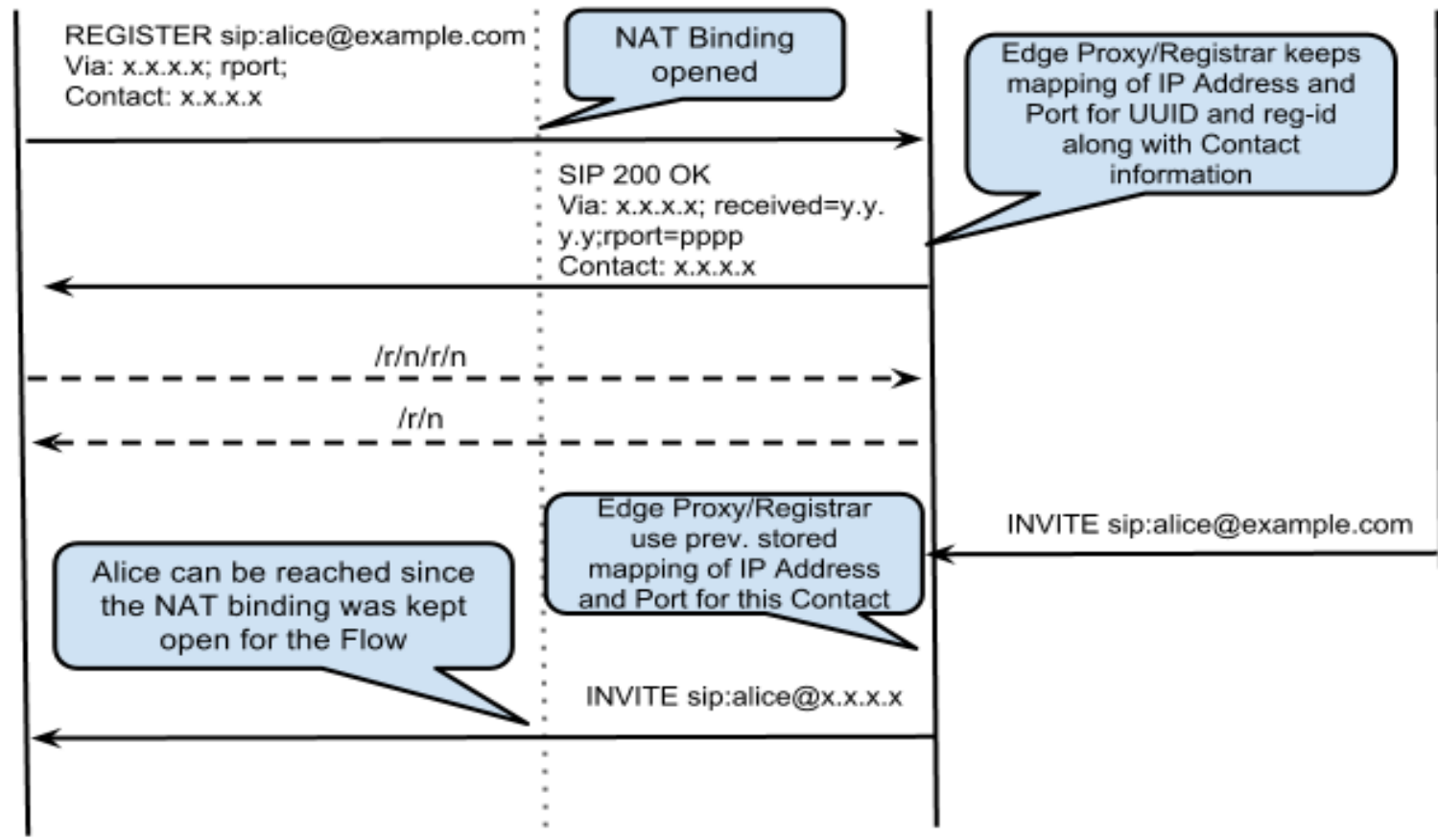
Why SIP Over WebSockets ?

- Adding the Browser as just another endpoint
- Deliver support for reusable applications that don't have to care about transport and if they do can still determine the transport type
- Applications see the real addresses instead of the invalid ones
- Applications : B2BUA, UAC, UAS and Proxy
- Allow for Convergent Applications : SIP + HTTP + Media

Handling Lost Connection

- WebSockets offers a keepalive mechanism and RFC 5626 (SIP Outbound) at SIP Layer is used to detect whether or not a connection is up
 - allowing for re-REGISTER and in browser notifications to reset the media streams and application status

KeepAlives



and take appropriate action

Bridging Existing SIP Infrastructure

- WebRTCComm : JavaScript Client Side Library to create HTML5 Cross Browser Web Phones with no plugins.
 - Contributed to Mobicents and Maintained by Orange Labs

```
webRtcCommClient.call(calleePhoneNumber,  
callConfiguration);
```

- Uses SIP Over WebSockets to communicate with the backend and bridge the existing telco SIP/IMS Infrastructure.

More gotchas

- Difficult to analyse with Wireshark
 - Masking is in effect
 - No dissectors yet
 - Best dissector is the generic frame dissector
- Difficult for unit testing
 - There is too much browser-specific behavior
 - No peer-to-peer testing - so no more shootist-shootme tests, all calls must be through a proxy or B2BUA (server role)

Questions ???

Don't Wait 'til the end, interrupt is mandatory !!!



Thank you !



<http://telestax.com/>