webRTC and Related Technologies

An Introduction – Q1 2013

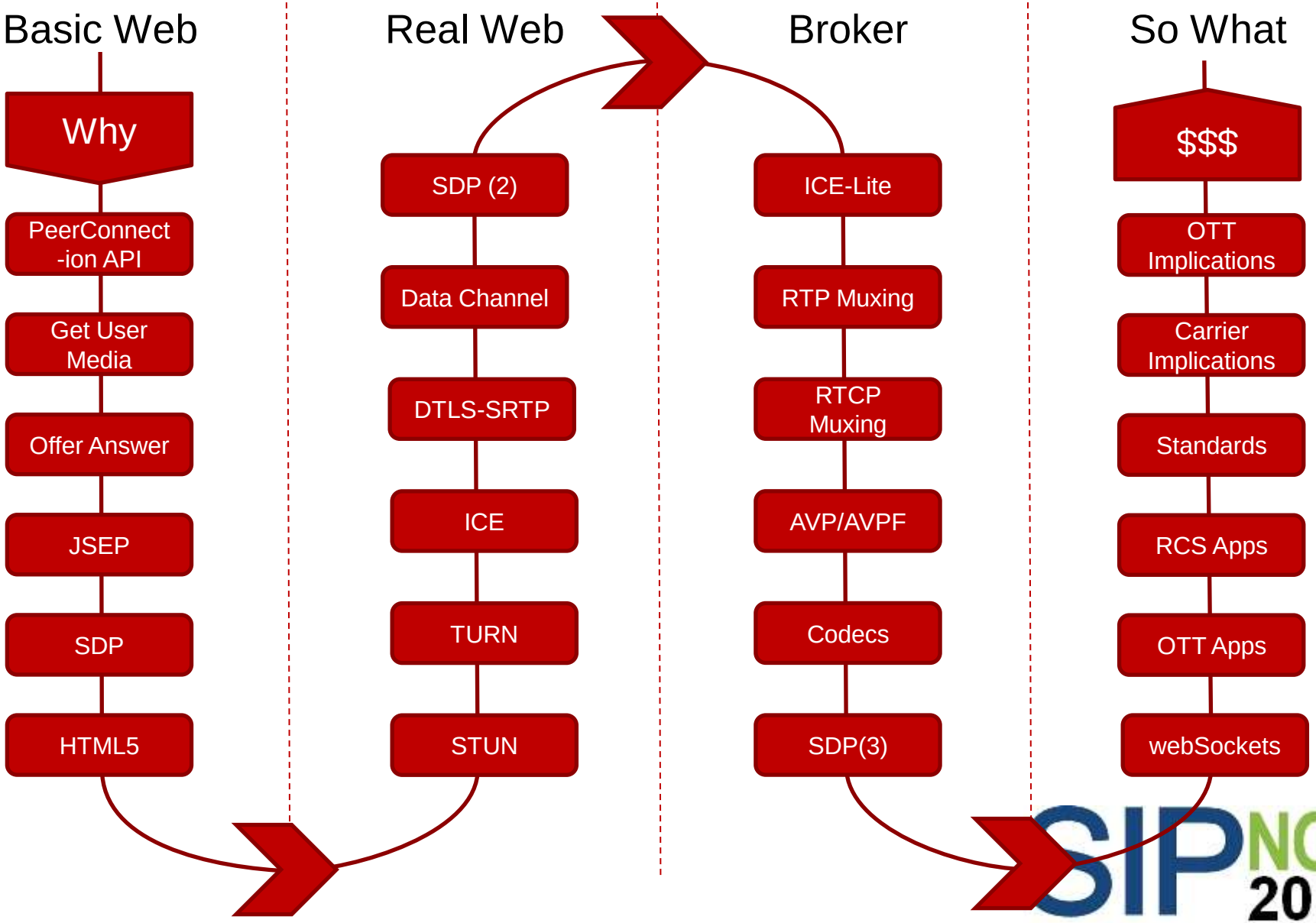
SIP NOC 2013
SIPFORUM

Web Real Time Communications - Outline

- Motivation
- **Applicable Technologies**
  - **HTML5**
  - **webRTC**

    What it is
    What is it not
    Session Management Alternatives
    Standards Perspective
    Technology Perspective
- What are the industry implications ?

The Quick tour



Assertion:

7.3 Billion people by 2016
2 Billion Internet Connected People

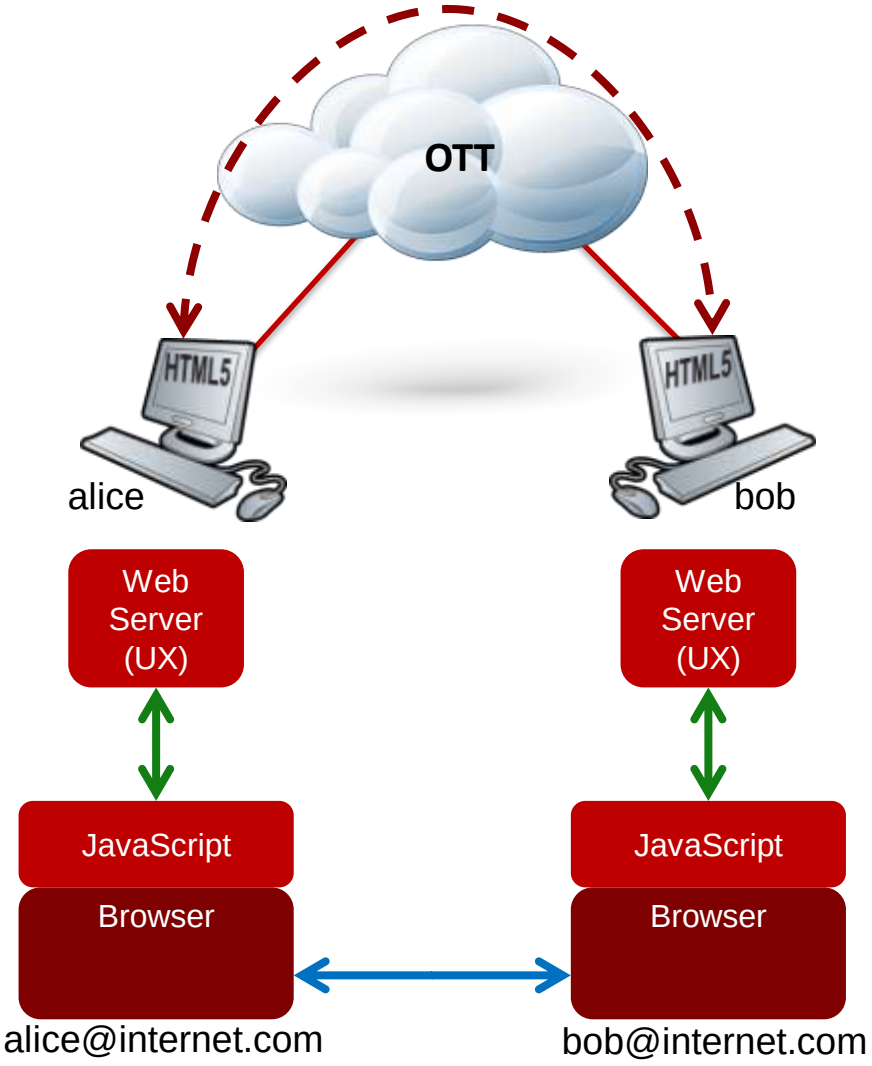
Multiple Connected Devices per Consumer

- **The Internet Service Provider industry wants to dominate Real Time Communications**
- WebRTC is the technology that enables that goal
  - Embedded in “every” browser
  - Fully capable of high quality secure multimedia communication media streams
- Aggressive Strategies
  - Full Voice, Video and Data
  - Simple to build-to APIs
  - Composite application user experiences
  - Diverse and specialized user experiences
  - Retain the Carrier for devices, pipes, and taking regulatory heat

WebRTC

SIPNOC 2013
SIPFORUM

Say you are “the internet” and you want to make a call…



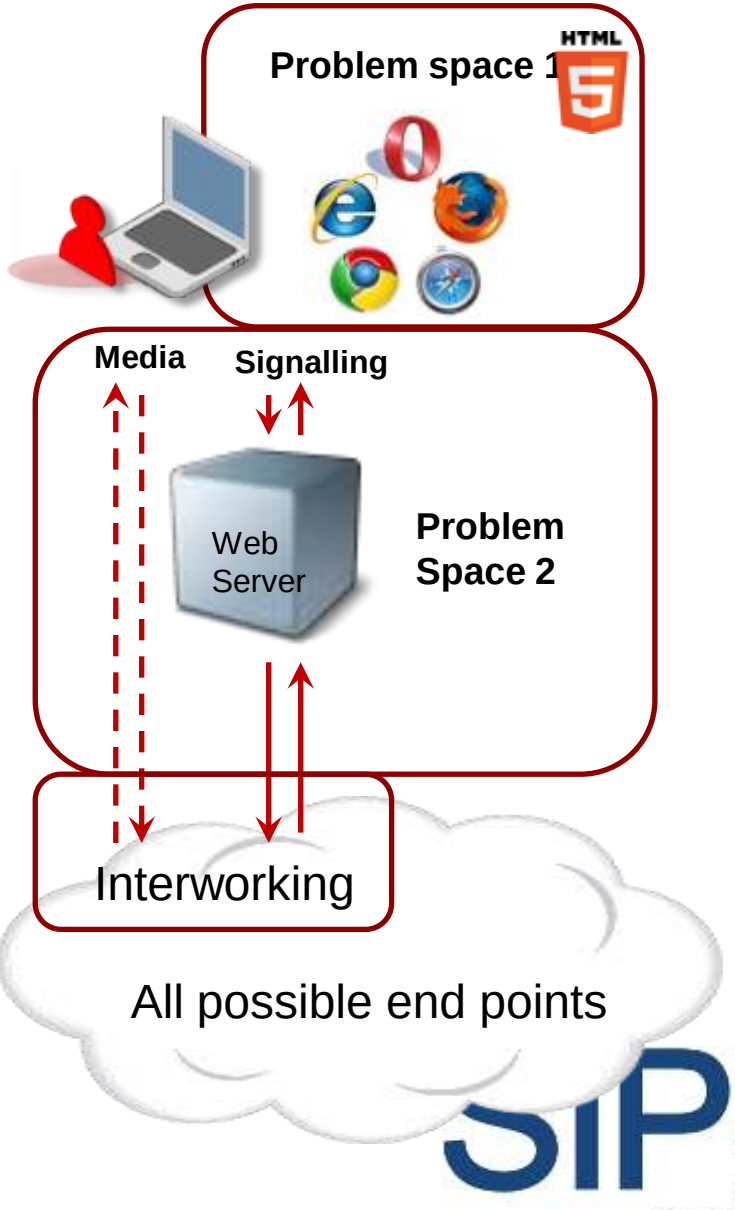
- Embed the media capabilities in everybody’s browser
- Provide Point to Point Media Exchange
- Deliver the user experience as a web page
- Use internet addressing schemes
- Use a session model that suits the controlling user experience
- Don’t make it the app, make it ***part*** of the App

Browser Real Time Communication Standards

- **Problem space 1 (the browser):**
  - Open standards for web browser APIs to access the device hardware and enable real time communications

    W3C WebRTC initiative
- **Problem space 2 (communication):**
  - Open standards for communication establishment between web browser end points

    IETF RTCweb initiative



Standards development being closely match by rapid industry experimentation.

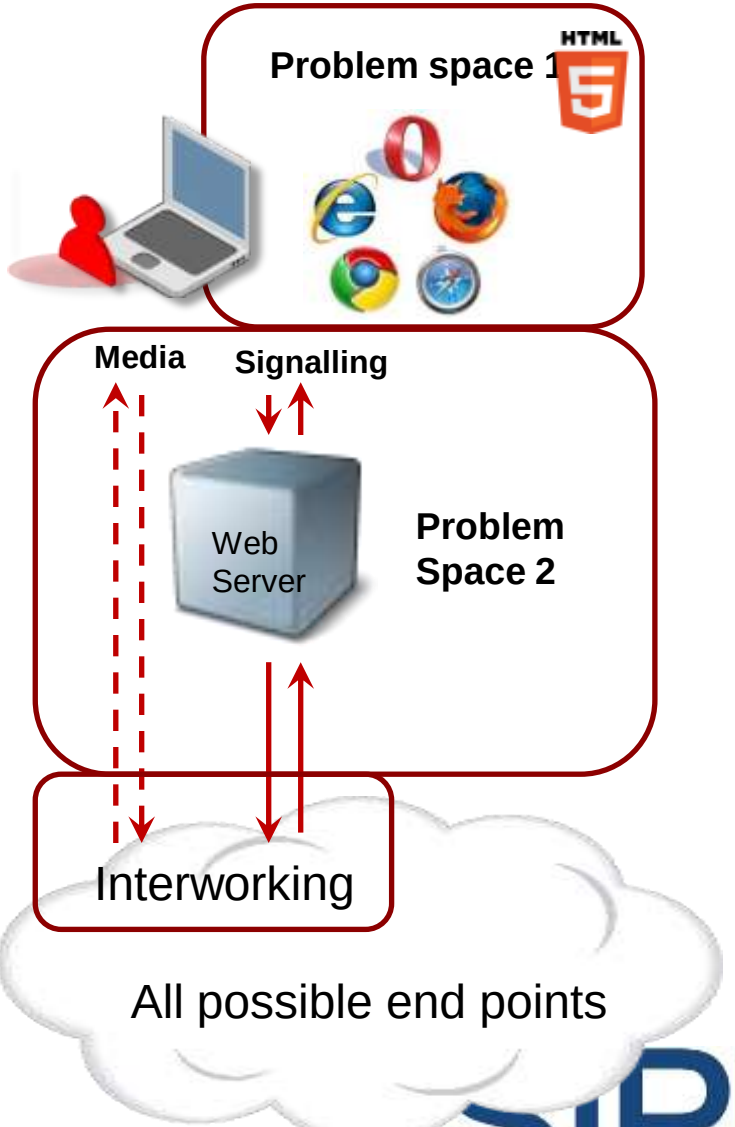
Browser Real Time Communication Standards

- **Problem space 1 (the browser):**
  - Open standards for web browser APIs to access the device hardware and enable real time communications

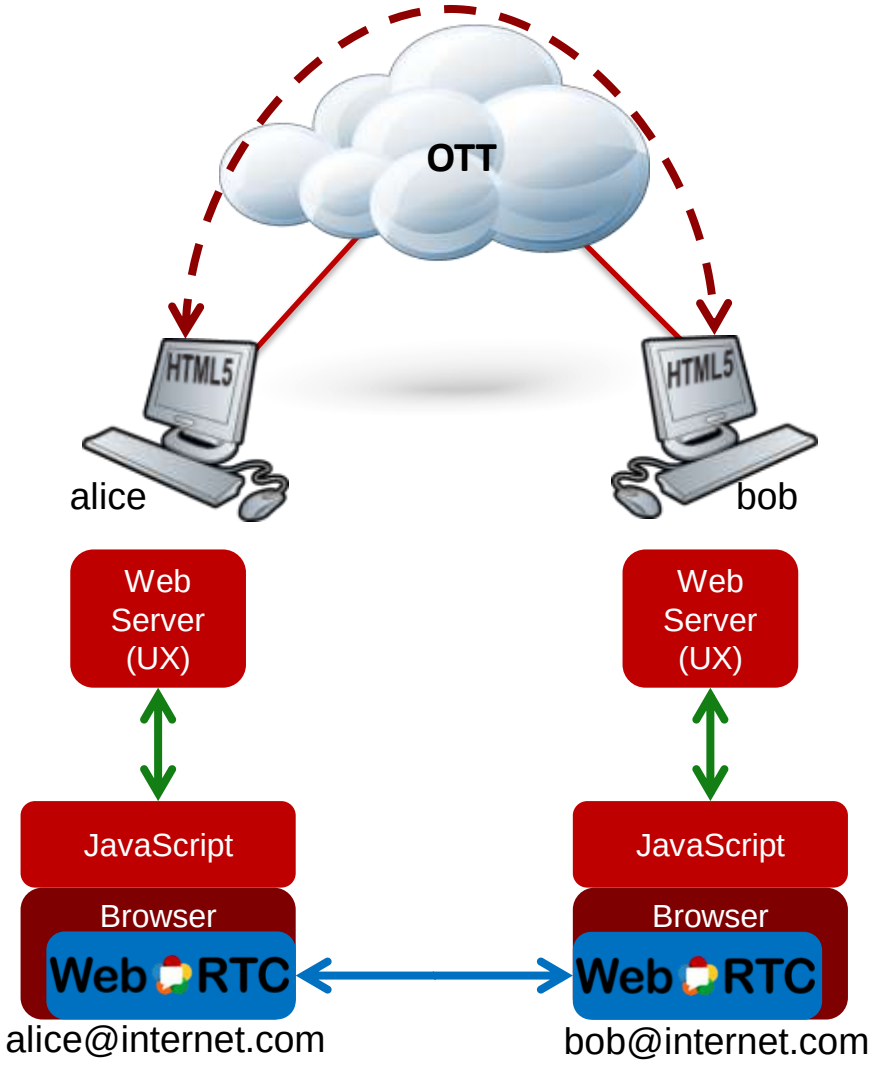
    W3C WebRTC initiative
- **Problem space 2 (communication):**
  - Open standards for communication establishment between web browser end points

    IETF RTCweb initiative

Bundled together these initiatives are commonly called WebRTC



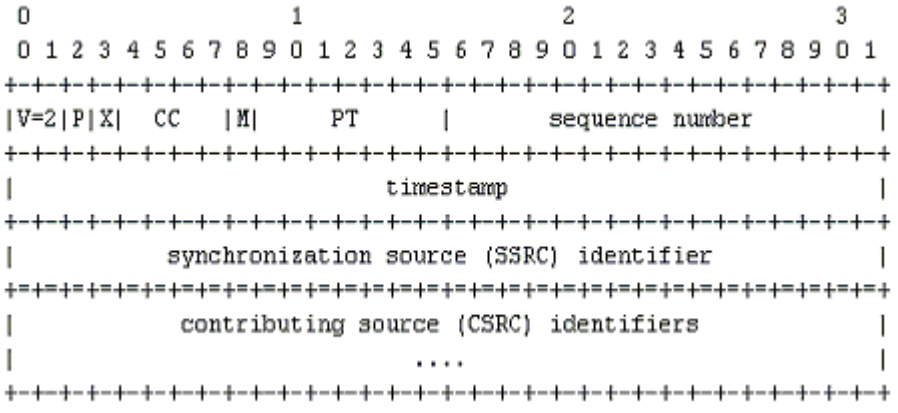
Say you are “the internet” and you want to make a call…

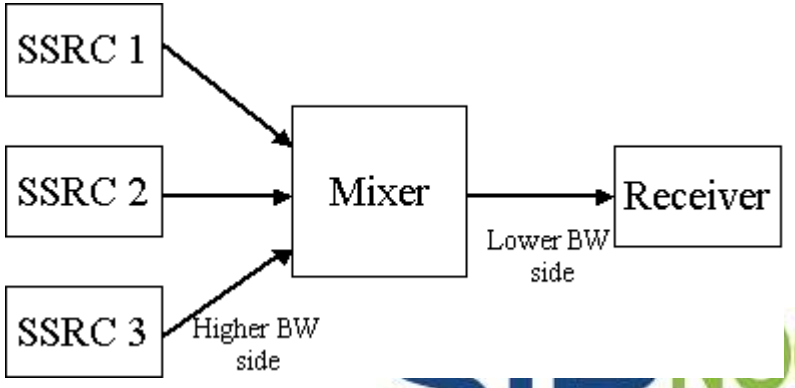


- webRTC technologies are embedded in the Browser
- Primary job is Media Delivery
- Media Delivery is built on UDP and RTP
- UDP
  - Efficient streaming with real-time efficiency
  - Limitations with respect to retrans-mission, stream preservation etc.
- RTP
  - Encapsulate the stream in descriptive information
  - Used primarily for streaming media
  - Provides sequencing, synchronization and loss detection
  - Provides no reliability, no packet repair, no retransmission.

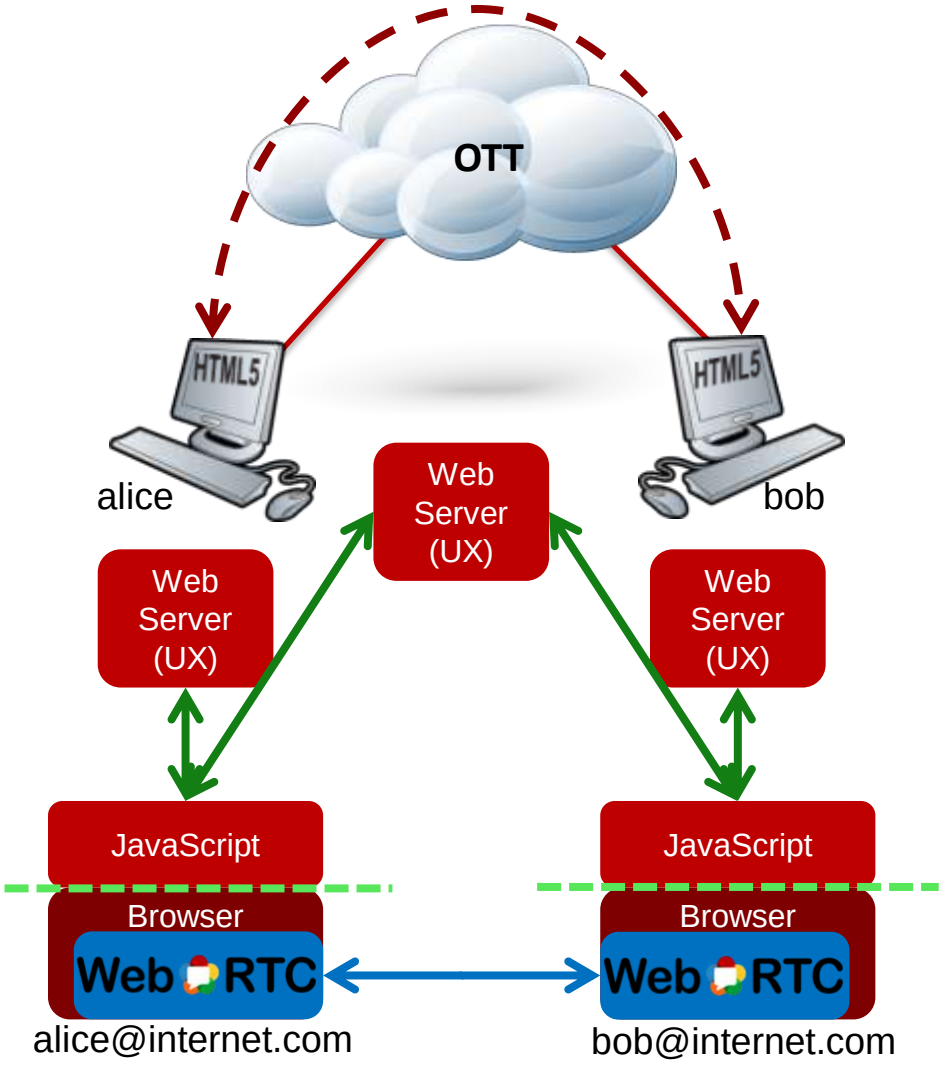
Real-time Transport Protocol (RTP)

- Used for Audio and Video in IP Networks
- Augmented with RTCP (Control Protocol) for monitoring and control
- Handles sequencing and loss detection, but not retransmission
- Enables multiple sources to be synchronized into a single stream
- The guts of the protocol header:
  - Header info - version (V): 2 bits, padding (P): pad to fixed block size for encryption, extension (X)
  - CSRC count (CC): 4 bits: The CSRC count contains the number of CSRC identifiers that follow the fixed header.
  - payload type (PT): 7 bits: This field identifies the format (e.g. encoding) of the RTP payload and determines its interpretation by the application.
  - sequence number: 16 bits: The sequence number increments by one for each RTP data packet sent, and may be used by the receiver to detect packet loss and to restore packet sequence. The initial value of the sequence number is random
  - timestamp: 32 bits The timestamp reflects the sampling instant of the first octet in the RTP data packet. Can be used in sorting out synchronization of multiple streams, or in the resolution of frame timing in some Video Coding schemes.
  - SSRC: 32 bits The SSRC field uniquely identifies the synchronization source.
  - CSRC list: 0 to 15 items, 32 bits each: The CSRC list identifies the contributing sources for the payload contained in this packet. The number of identifiers is given by the CC field. If there are more than 15 contributing sources, only 15 may be identified. CSRC identifiers are inserted by mixers, using the SSRC identifiers of contributing sources.



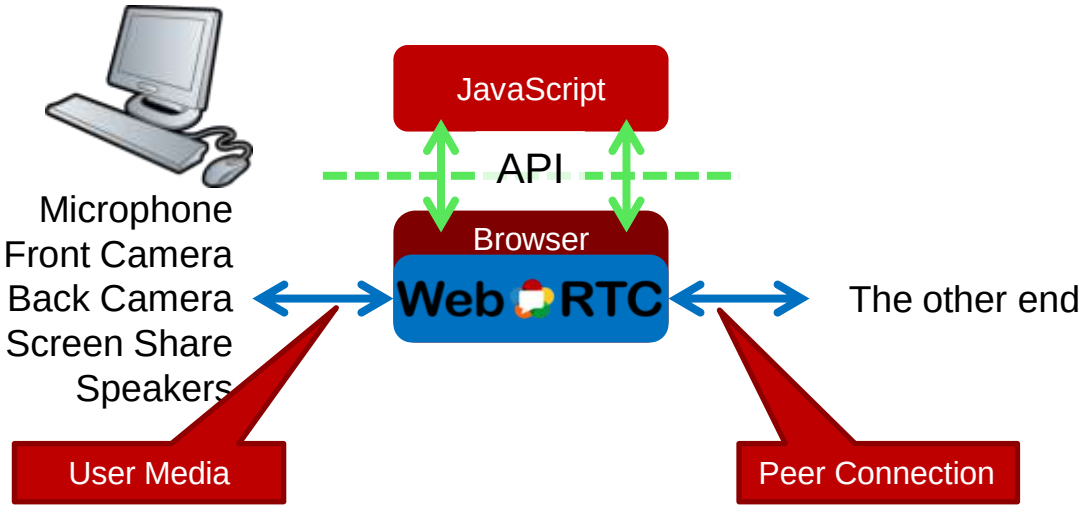


Controlling the Browser - JavaScript



- Mechanism needed to control the Browser RTC actions
  - JavaScript is being chosen
  - A Uniform API for all browsers
- Addressed by W3C (mostly)
- Makes the RTC completely subservient to the Web-delivered application
- Allows for Page-by-page, user-by-user, situation-by-situation, call-by-call flexibility
- Users better be using the same application

Controlling the Browser - JavaScript

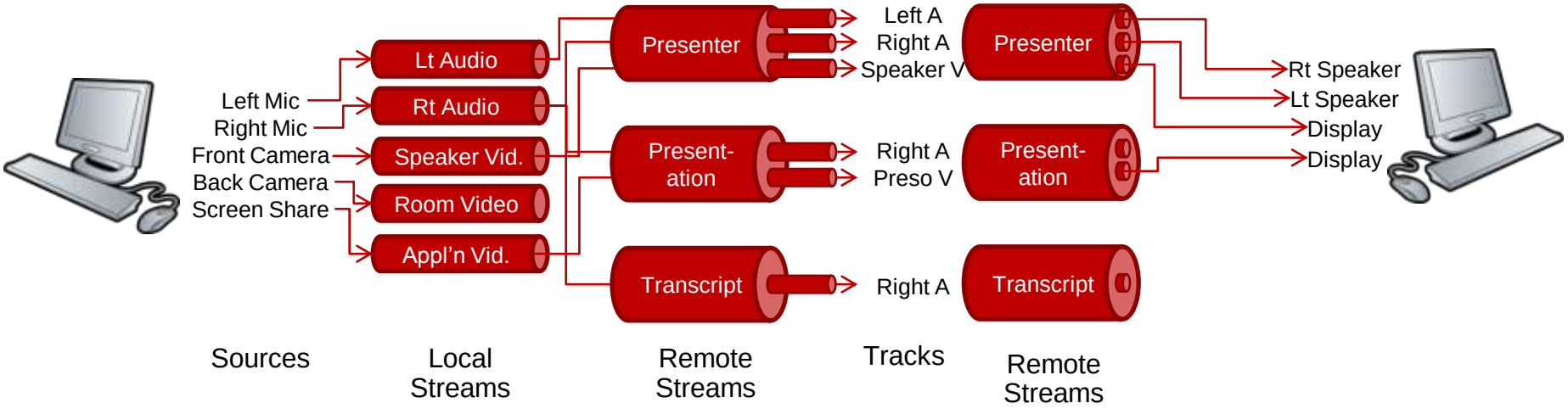


Three fundamentals control end-to-end Real Time Communications

- Describing the Media: Arrangement of Tracks, Channels etc.
- The Get User Media API
- The Peer Connection API

A standardized API is being been developed – All browsers to implement

Describing the Media



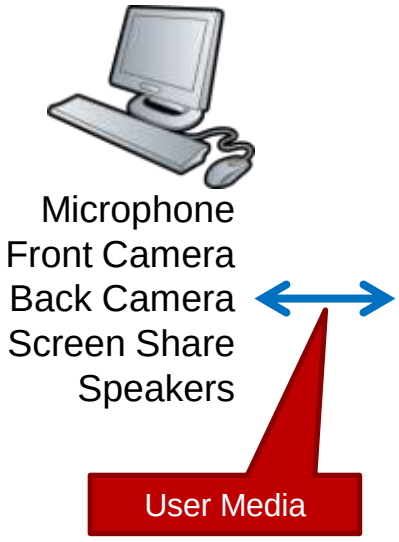
- The media description is passed across the API, to allow the JavaScript to create the media experience that it wants.
- The **MediaStream** Object:
  - getAudioTracks, getVideoTracks, getTrackById
  - addTrack, removeTrack, attribute boolean ended;
  - EventHandler(s) onended, onaddtrack, onremovetrack;
- Gives JavaScript applications fine-grained application control

Retrieve tracks in a stream

Manage tracks in a stream

Discover important events

Getting the User Media (Camera, Microphone…)



- Before the web application can access the user's media input devices it must let getUserMedia() create a **LocalMediaStream** .
- Once the application is done using media sources, it may revoke its own access by calling stop() on the **LocalMediaStream**
- On a call to getUserMedia() the browser must confirm with the user that the cameras and microphones can be used in the context of the browsed media.
- The outcome is the Local instance of the **MediaStream** object

Getting at the Other end of the Pipe

- The RTCPeerConnection API handles the Connection between two peers
- Manages the Local and Remote MediaStream descriptions
  - createOffer
  - createAnswer

  Later in section on JSEP
  - setLocalDescription

    readonly attribute RTCSessionDescription localDescription;

    SDP Creation for Local resources
  - setRemoteDescription

    readonly attribute RTCSessionDescription

    readonly attribute RTCPeerState

    SDP Creation for far end resources
  - addStream

    Manages tracks in a stream
  - removeStream

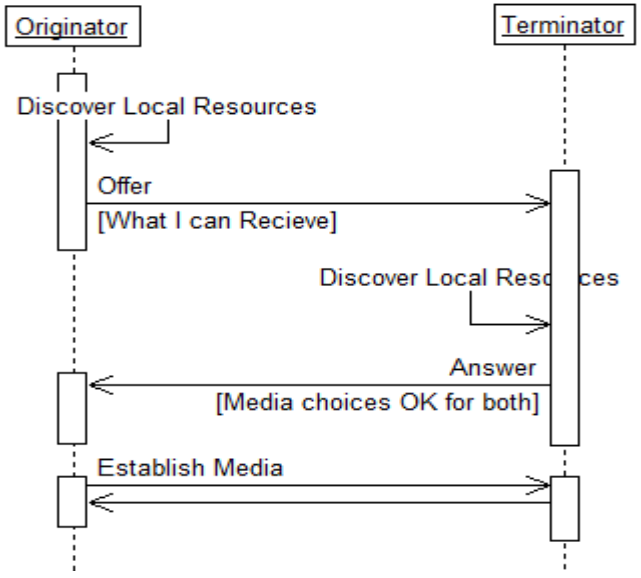
    Manages tracks in a stream
  - close
- Handles NAT Traversal information (more on that later)
  - updateIce
  - addIceCandidate

    readonly attributes RTCIceState, iceState, MediaStreamArray, MediaStreamArray remoteStreams;
- Supports Raw Data from peer to peer
  - createDataChannel

    attribute EventHandler - ondatachannel;
- Manages Events
  - attribute EventHandler - onnegotationneeded, onicecandidate, onopen, onstatechange, onaddstream, onremovestream, onicechange;

    Allows JavaScript action on events on the Local and Remote Streams

Offer Answer Model



- webRTC **Requires** the use of the Offer Answer Model
  - Analogous to a Business Card exchange
  - Hey ! Send your media over here !
  - OK, and you send yours over here
  - Alright ! I can do that.
  - We're communicating !
- JSEP enforces the law
  - IETF Specification that describes the use of offer answer
  - JSEP covers the Media Session Description
  - JSEP ***does not*** govern how the offer and answer are used by the application to create a session

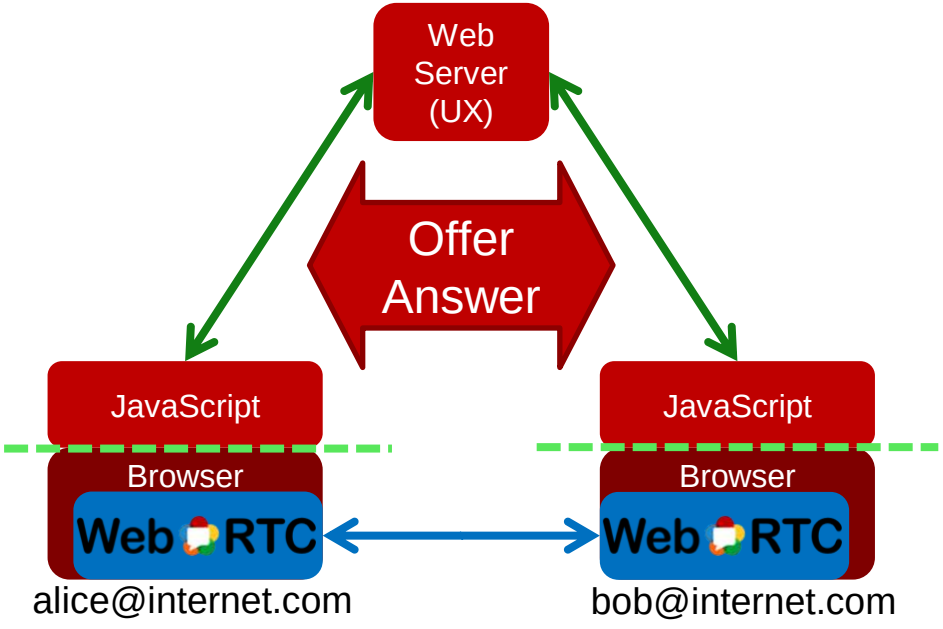
JSEP (JavaScript Session Establishment Protocol)

- Provides the Description and Operation of the JavaScript APIs that will govern the Session Characteristics
- Establishes the use of Offer / Answer
- Establishes the use of the **Session Description Protocol** (SDP) to describe the offer and answer
- The offer
  - CreateOffer in the RTCPeerConnection element of the Javascript API
  - Builds an offer SDP Containing supported configurations for the session:
    descriptions of the local MediaStreams attached to this PeerConnection,
    the codec/RTP/RTCP options supported by this implementation,
    any ICE candidates that have been gathered by the ICE Agent
    Constraint and control information
- The answer
  - CreateAnswer in the RTCPeerConnection element of the Javascript API
  - Builds an answer SDP Containing supported session configurations:
    descriptions of the local MediaStreams attached to this PeerConnection,
    the codec/RTP/RTCP options supported by this implementation,
    any ICE candidates that have been gathered by the ICE Agent
    Constraint and control
  - ***Constrained by the Offer – a subset***.

From the Draft: However, the actual mechanism by which these offers and answers are communicated to the remote side, including addressing, retransmission, forking, and glare handling, is left entirely up to the application.

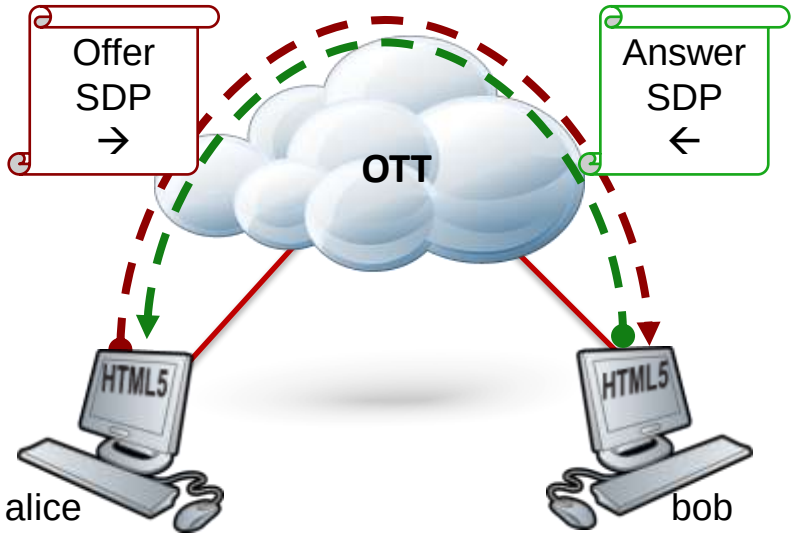
Implications of JSEP Adoption

- Media Capability control is hardwired in the Browser, evolving with Hardware and Browser releases
- JSEP defines that the interaction with the Browser is about Media capabilities only
- The Session State Machine is implemented in JavaScript, and determined by the Web Server that downloads the JavaScript
- Each application (or each instance of an application) can create it's own user experience and session state machine.



- Each Application creates its own user experience
- Supports web-page converged applications where communications is just an integrated component.
- No standard at the UX , App and session Layers

Session Description Protocol (for completeness)




```
v=0
o=alice 2890844526 2890844526 IN IP4
        host.anywhere.com
c=IN IP4 host.anywhere.com
t=0 0
m=audio 49170 RTP/AVP 0
a=rtpmap:0 PCMU/8000
m=video 51372 RTP/AVP 31 32
a=rtpmap:31 H261/90000
a=rtpmap:32 MPV/90000
```

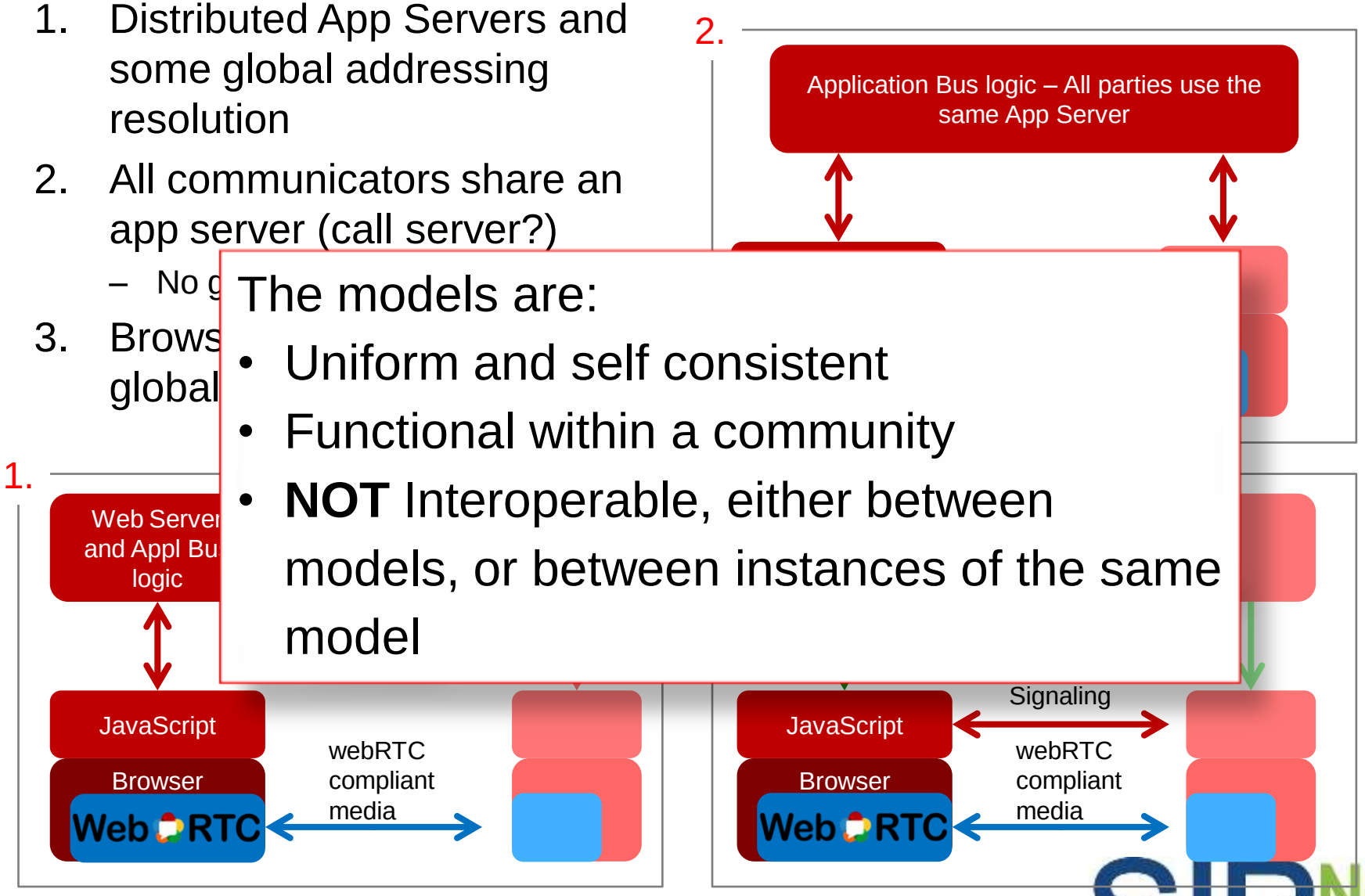
- C-lines (Connection)
  - <network type> <address type> <connection address>
  - Can show up in the Media section and override session level c=
- M-lines (Media)
  - <media> <port> <transport> <fmt list>
  - fmt is Typically the RTP media payload type(s)
  - Applies to lines following the m=
- A-lines (attribute)
  - rtpmap:<payload type> <encoding name>/<clock rate>[/<encoding parameters>]
  - Multiple attributes are allowed per m= to cover multiple payload types
- v= is a version number
- o= is an “owner”
- s= is session name
- t= is a session time

Creating a Session – not always like a telecom network

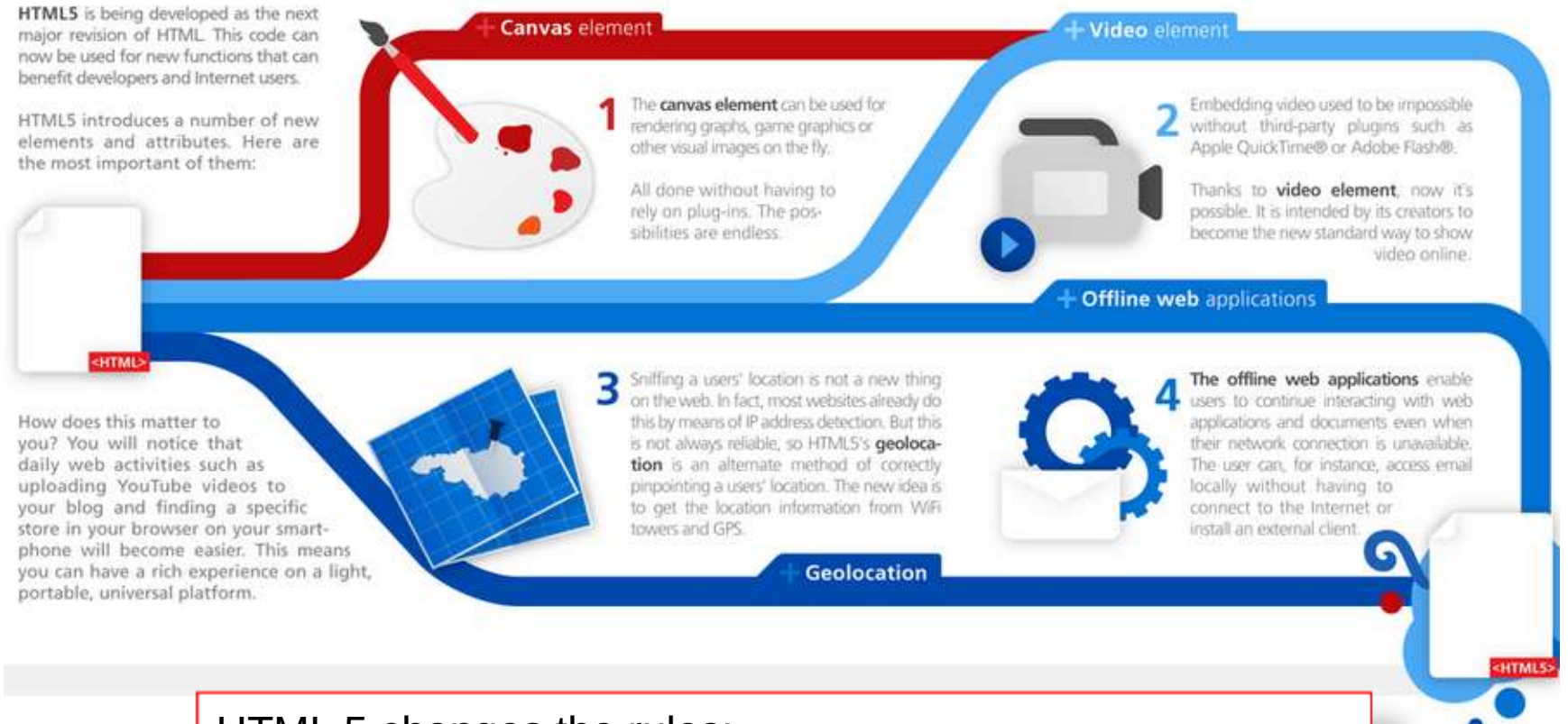
1. Distributed App Servers and some global addressing resolution
2. All communicators share an app server (call server?)
   - No g
3. Brows
   global

The models are:

- Uniform and self consistent
- Functional within a community
- **NOT** Interoperable, either between models, or between instances of the same model



Driving the User Experience: HTML5



HTML 5 changes the rules:
It permits the deployment of single “App-like” solutions to multiple devices and device types. No more developing, deploying and managing separate applications on a device-by-device basis.

source: focus.com

HTML5: Superior User Experience

- Delivers an “App-like” experience
  - Native look and feel, widgets, window layouts
  - Installable and Manageable Web Apps
  - Touch and swipe events, notifications, etc.
  - Browser chrome control and hiding
  - Browser breakout

iOS Web Icons

Sencha Touch Hybrid

- Removes technology limitations in current solutions
  - Video embedding
  - RTC Capability embedding
  - UI Enhancements – corners, transparency etc.
  - From Cookies to Local Storage and SQL capabilities
  - Allows offline execution

Windows 8

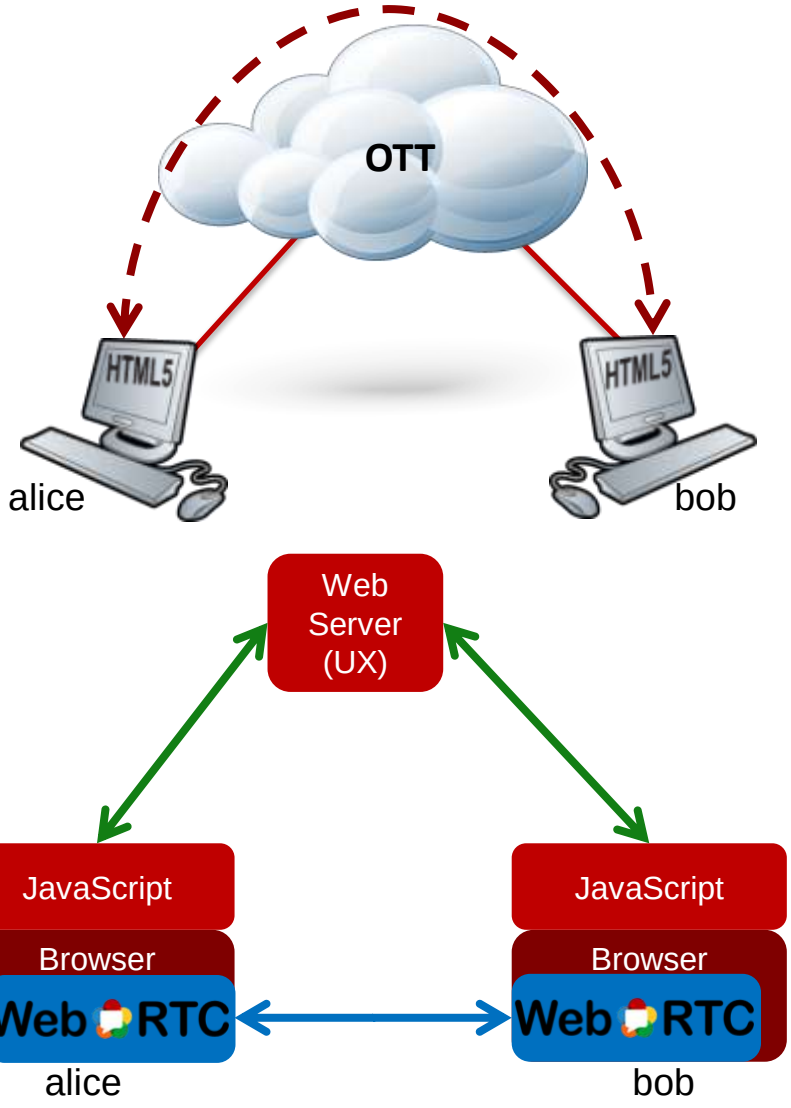
Facebook Spartan

- Adoption now proceeding full steam ahead

Current Browsers	Score	Bonus
Maxthon 3.4.5 »	457	15
Chrome 23 »	448	13
Opera 12.10 »	419	9
Firefox 17 »	392	10
Safari 6.0 »	378	8
Internet Explorer 10 »	320	6

Current TVs	Score	Bonus
Sharp Aquos »	365	6
Toshiba »	365	6
Sony Internet TV »	357	8
Philips NetTV »	342	16
GoogleTV »	341	8
Toshiba »	325	2
Samsung Smart TV 2012 »	302	12
Panasonic Smart Viera »	240	2

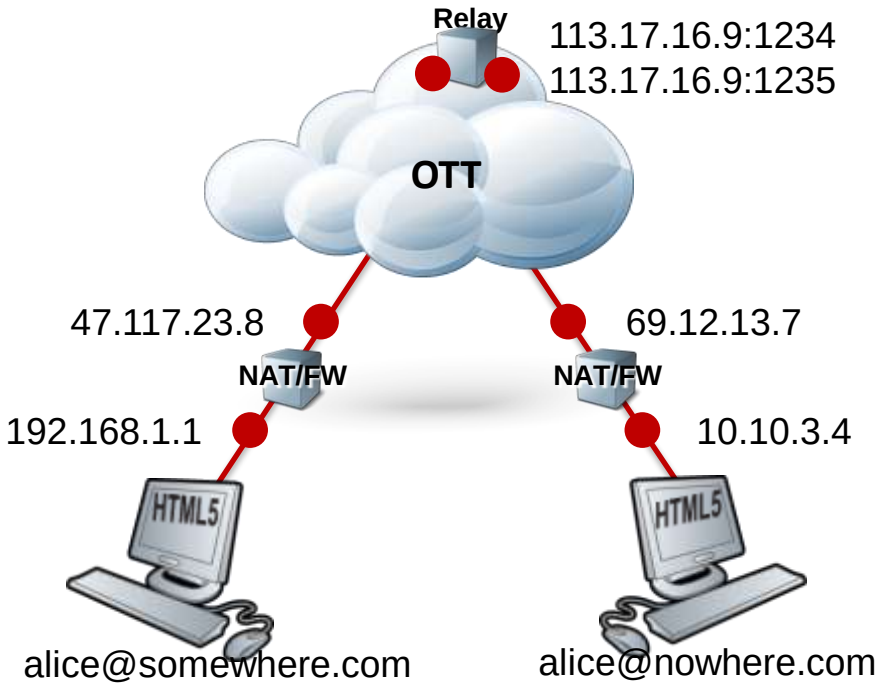
Tying it all together (simple version !)



Web-to-Web is easy !

- Get local media and attach to devices (JavaScript APIs)
- Set up remote connection for connection to IP addresses across the world
- Bundle it into an API for sending to the other guy
- Describing it in SDP
- Finding the other guy (Offer Answer)
- Presenting the User Experience

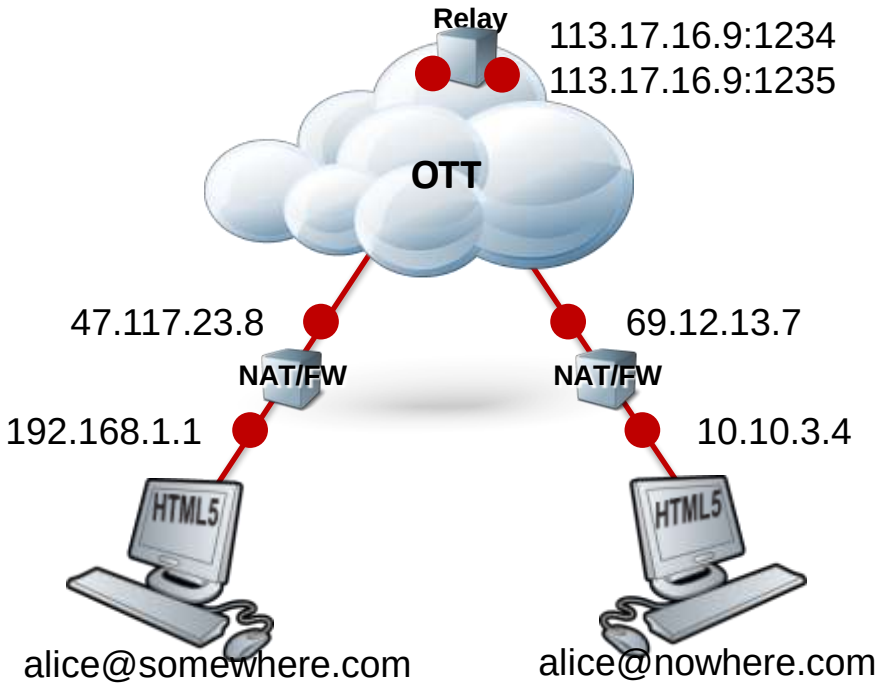
Getting Ugly – the evil NAT monster



Describe the problem with sending my address

- I don't know my public address, only my private one.
- If I can find out my public address data still needs to flow out for data to get in
- I may need to result to a media relay somewhere in the network
- I may have multiple private addresses (WiFi and Wired), multiple public addresses (VPN and OTT) and even multiple relay servers
- Which addresses do we use to trade media ?

The Solution: ICE (Interactive Connectivity Establishment)

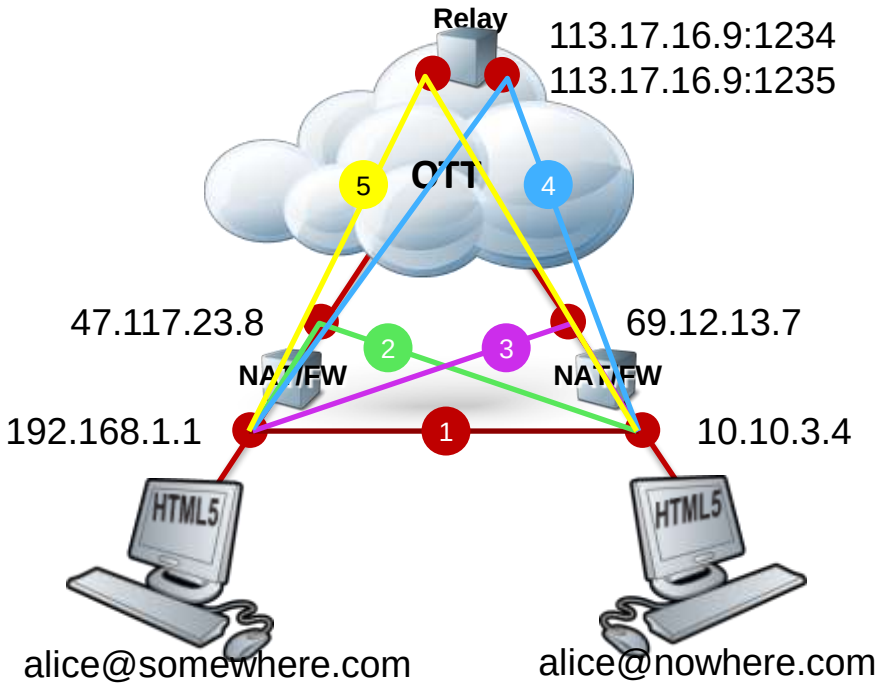


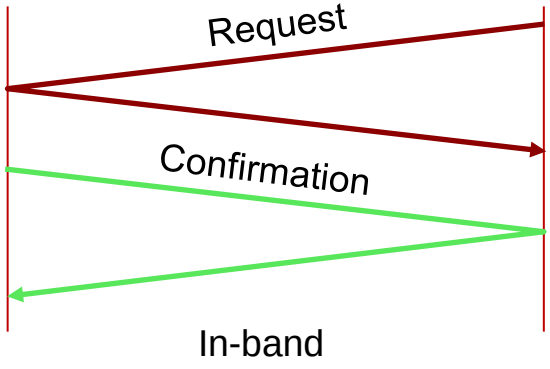
ICE is a collection of approaches to ensure NAT Traversal in all cases

- Combination of STUN, TURN and offer-answer
- Can take a long time to complete the optimal choice
- Negotiated as part of RTP stream after SDP exchange

- Step 1: Allocation – Find out my public addresses – a STUN or TURN Server
- Step 2: Prioritization – Prioritize them, using address type, local preference, and group info
- Step 3: Initiation – encode the candidates (see step 1) into the Offer SDP.
- Step 4: Allocation – find out the public addresses for the Answering party (can be done earlier)
- Step 5: Information – respond with an ANSWER SDP that contains the candidate information
- Step 6: Verification – try every pair in priority order with a media check query/response (media level)
- Step 7: Coordination – controlling party chooses path, and signals completion with a flagged check
- Step 8: Communication – finally ! Media
- Step 9: Confirmation – Adjust for discovery of alternatives at the SIP level with re-invite

The Solution – ICE Step 6

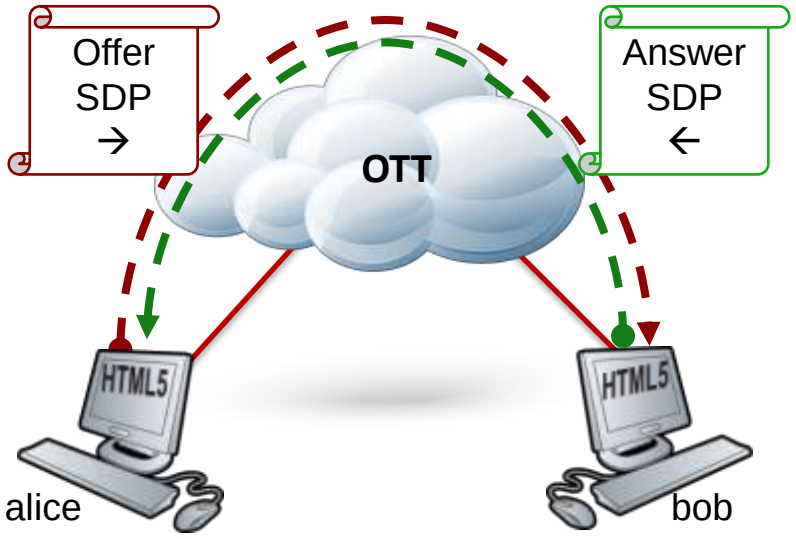




Describe STUN, TURN, and ICE (Multiple Charts)

- Step 1: Allocation
- Step 2: Prioritization
- Step 3: Initiation
- Step 4: Allocation
- Step 5: Information
- Step 6: Verification
- Step 7: Coordination
- Step 8: Communication
- Step 9: Confirmation

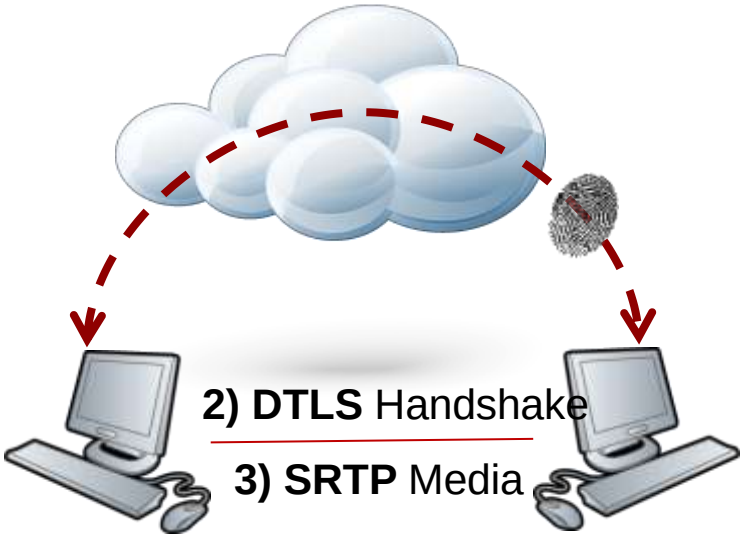
Update on SDP Carriage of ICE Candidates



- The SDP Contains “a=“ lines that are specific to ICE execution
- Final media session addresses are negotiated as an outcome of the SDP Offer Answer
- → Media can be delayed after “answer”
- → provisional candidates may be involved.

- a=candidate: foundation component-id transport priority connection-address port cand-type [rel-addr] [rel-port] *(extension-att-name extension-att-value)
  - foundation and component identify the stream and RTP / RTCP component of the stream
  - Priority is a computed priority as discussed above.
  - Candidate type is: host, relay, server or peer reflexive (with root addresses)
- Examples
  - a=candidate:1 1 UDP 2130706431 10.0.1.1 8998 typ host
  - a=candidate:2 1 UDP 1694498815 192.0.2.3 45664 typ srflx raddr 10.0.1.1 rport 8998
- a=ice-ufrag:<user ID fragment> and a=ice-pwd:<password> are used to pass credentials for authentication of the streams

SDP's role in DTLS-RTP



- Self signed certificates exchanged in the media plane avoids the need for Public Certificates
- DTLS handshake carries public keys.
- Fingerprint carried in SDP is used to verify these.
- The fingerprint binds the DTLS key exchange in the media plane to the signalling plane.
- The secured and verified DTLS exchange is used in creating the SRTP Stream

- DTLS-SRTP is seen by the Internet community as more desirable given the untrusted control plane environment, while building on established functionality of TLS and SRTP
- DTLS-SRTP is currently mandatory in WebRTC.

SIP INVITE with DTLS-SRTP information

Invite from Alice to Bob

```
INVITE sip:bob@example.com SIP/2.0
To: <sip:bob@example.com>
From: "Alice"<sip:alice@example.com>;tag=843c7b0b
Via: SIP/2.0/TLS ua1.example.com;branch=z9hG4bK-0e53sadfkasldkfj
Contact: <sip:alice@ua1.example.com>
Call-ID: 6076913b1c39c212@REVMTEpG
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, UPDATE
Max-Forwards: 70
Content-Type: application/sdp
Content-Length: xxxx
Supported: from-change
v=0
o=- 1181923068 1181923196 IN IP4 ua1.example.com
s=example1
c=IN IP4 ua1.example.com
a=setup:actpass
a=fingerprint: SHA-1 \
4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
t=0 0
m=audio 6056 RTP/AVP 0
a=sendrecv
a=tcap:1 UDP/TLS/RTP/SAVP RTP/AVP
a=pcfg:1 t=1
```

- SDP Capabilities Negotiation (RFC 5939)
- adds alternatives to “m=“ line,
  - with indicated preferences,
  - in a backwards compatible way

Fingerprint, used to verify certificate receive in user plane DTLS handshake

Transport protocol options:
- UDP/TLS/RTP/SAVP = SRTP-DTLS,
- RTP/AVP = RTP

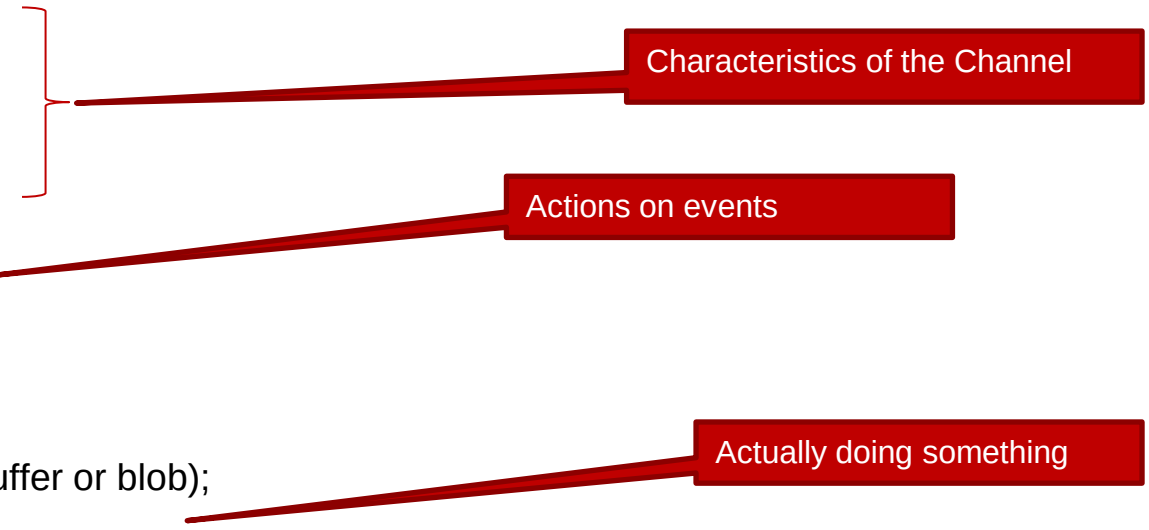
Indicated preference for SRTP-DTLS configuration

Sending Raw Data – Collaboration, Files, etc.

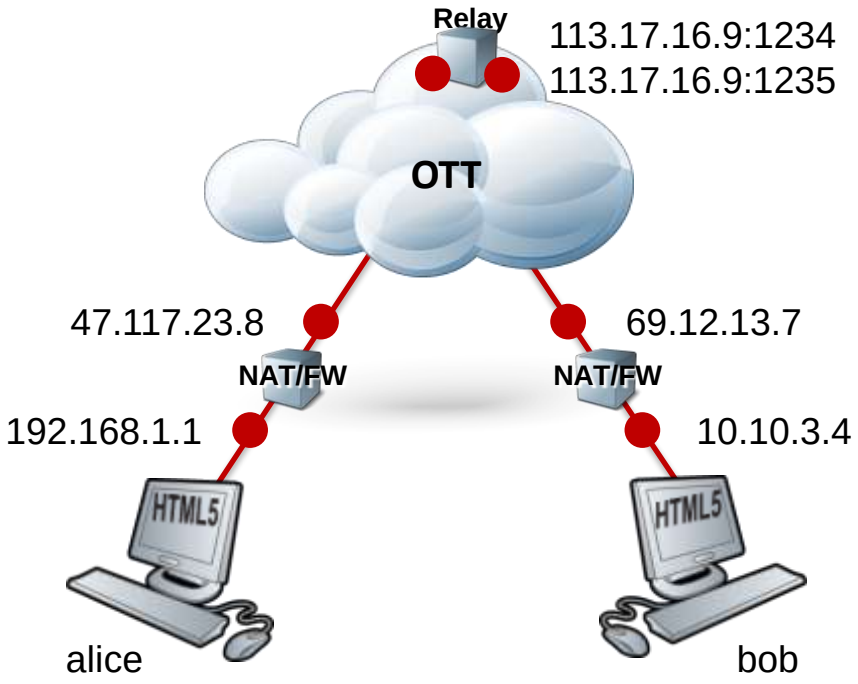
- Multi-media would not be the same without the ability to exchange anything and everything
- DataChannel API allows the creation of a stream of anything
  - Strings, Array of characters, or just a binary blob
- DataChannel API – Raw Data Exchange between webRTC Peers
  - attributes:

    attribute label;
    attribute reliable;
    attribute readyState;
    attribute bufferedAmount;
    attribute onopen;
    attribute onerror;
    attribute onclose;
  - **close** ();

    attribute onmessage;
    attribute binaryType;
  - **send** (string, arraybuffer or blob);



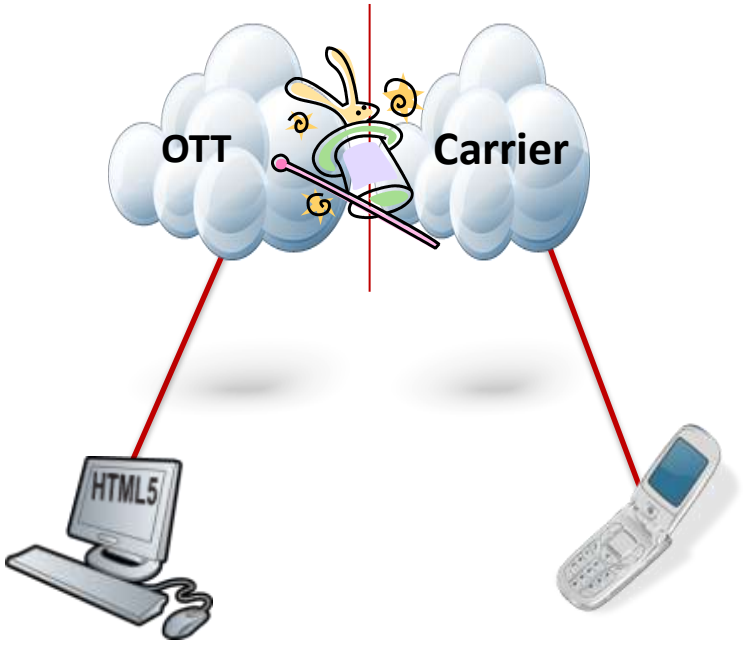
Tying it all together (the real OTT version)



- The real OTT Version is really just a NAT friendly version of the basic interaction
- ICE is used for NAT Traversal
  - Slower media setup
  - High inherent complexity
  - 100% effective
- Security is provided via DTLS-SRTP
- The DataChannel API permits exchange of arbitrary data
  - Used for other collaboration functions like screen share and P2P file transfer

Considerable complexity for the "simple case"

What about Carrier involvement?



alice@somewhere.com

987-6543@carrier.com

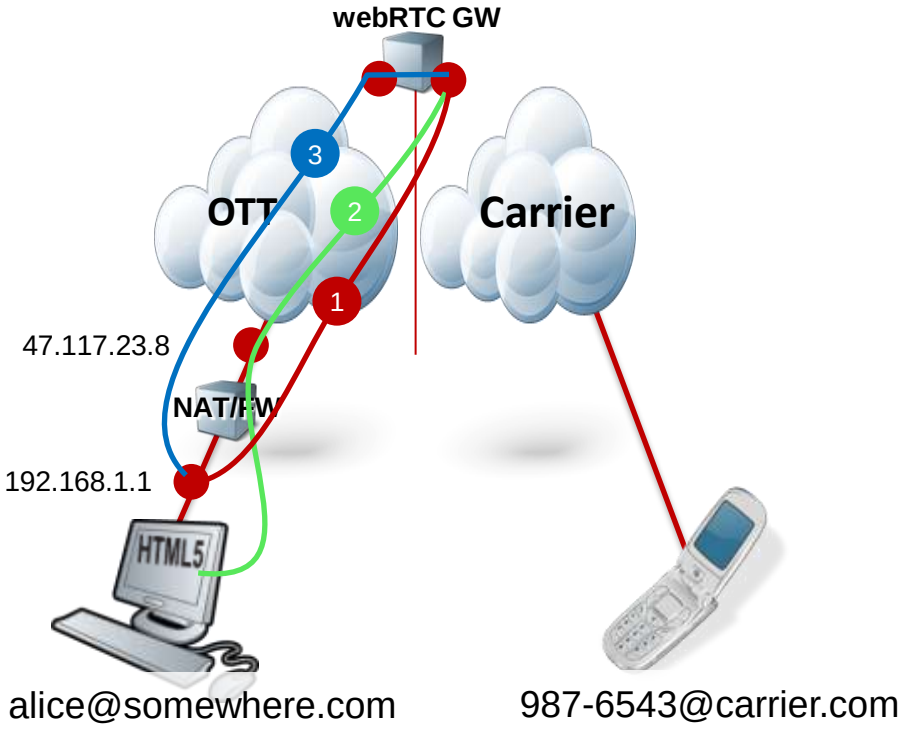
The carrier network must….

- Look like a webRTC Endpoint to All Browsers
  - What are the other webRTC Compliance attributes ?
- Conform to RCS communication application behavior for all Phones.
- Find a way to federate addressing for radically different domains*
- Compensate for different session and application models*

WHAT HAPPENS WHEN…
THE CARRIERS WANT TO PARTICIPATE ?

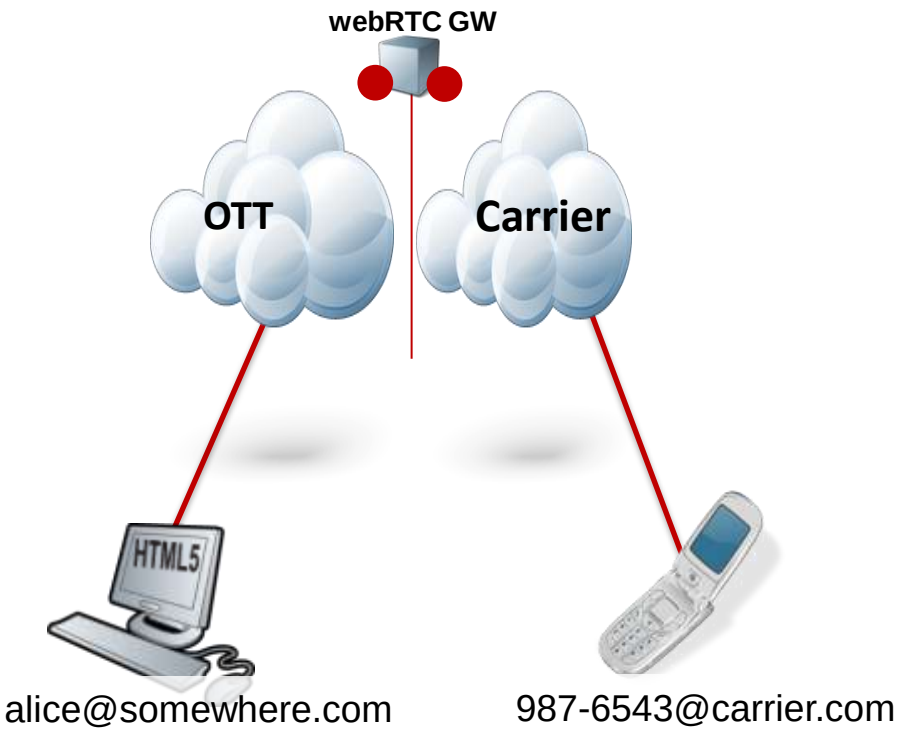
* Snooze check - Don’t let yourself be fooled

ICE Lite



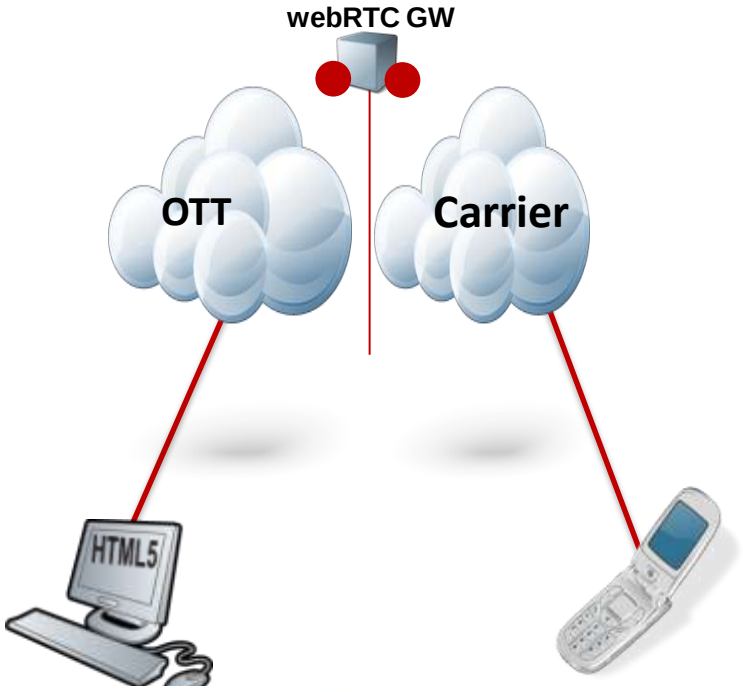
- ICE Lite does the ICE on behalf of the Telephone
- Performed by the edge of the Carrier Network
- Capable of querying the address candidates offered by the OTT
- Capable of responding to the queries of the OTT client
- Not allowed to declare itself the controller in the ICE choice process

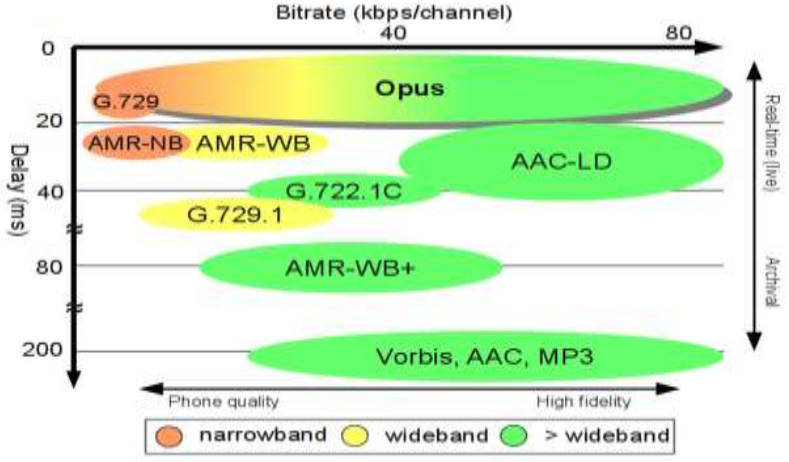
All in One Bucket – Media Muxing



- Multiple media streams and the use of ICE don't mix
  - Added delay
  - Added complexity
- The webRTC solution is to multiplex media into a single RTP Stream
  - Includes multiplexing RTCP
- Multiple mechanisms are under evaluation at this time
  - Grouping Sources [RFC 5888], or
  - Directly extending the SDP
- Non-multiplexed RTP is also required for interaction with legacy systems

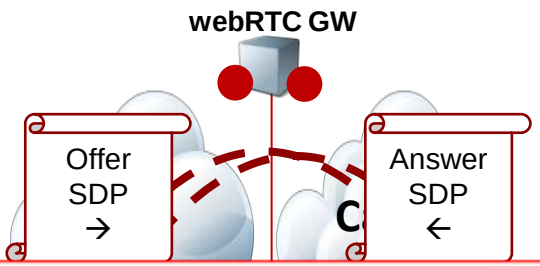
Codecs





- webRTC mandates different codecs
- Prime considerations:
  - High performance in high packet loss environment
  - Superior user experience available
  - lack of licensing encumbrances
- Audio Codec requirements:
  - OPUS
  - G.711 A-law and Mu-law
- Still a battle in standards over mandatory video codecs
  - VP8 and H.264 are the choices
  - Lack of a rule → no rules
  - No rules → we must transcode on an on-demand basis

Back to our good Friend SDP



- Add in changes to cover Channel Muxing, ice lite, and SRTP Key Exchange.
- a=ice-lite for the gateway ICE
- a=group BUNDLE for muxing RTP
- a=rtcp-mux for muxing in the RTCP traffic

```
v=0
o=genbanduser 2890844526 2890844526 IN IP4 gcfw.genba
s=-
c=IN IP4 64.129.37.225
t=0 0
a=ice-lite
a=group BUNDLE 1 2
m=audio 49178 RTP/SAVPF 99
a=rtpmap:99 opus/48000
a=mid:1
a=ssrc:43218 cname Hg6J8iKgrfdtLihs
a=ssrc 43218 msid:jeorhfds
a=ssrc: information: microphone
a=rtcp-mux
m=video 49178 RTP/SAVPF 98
a=rtpmap:98 VP8 90000
a-mid:2
a=ssrc:39322 cname:dkSK9dqork7fJwf3
a=ssrc:39322 msid:sldfiedf
a=ssrc:39322 information front camera
a=ssrc:93847 cname:sdi7FeW3jse2iHdw
a=ssrc:93847 msid:sldfiedf
a=ssrc:93847 information: back camera
a=ssrc:17339 cname:ruoWe35GfqowlaDf
a=ssrc:17339 msid:sldfiedf
a=ssrc:17339 information:presentation
a=rtcp-mux
a=candidate:1 1 UDP 2130706431 172.18.57.42 8998 typ host
a=candidate:2 1 UDP 1694498815 64.129.37.225 45664 typ srflx raddr 172.18.57.42 rport 8998
a=ice-ufrag:mckafred
a=ice-pwd:kashhfuds
```

Bundle the next two media lines

Media group media id

Describes the stream

RTCP will be multiplexed in

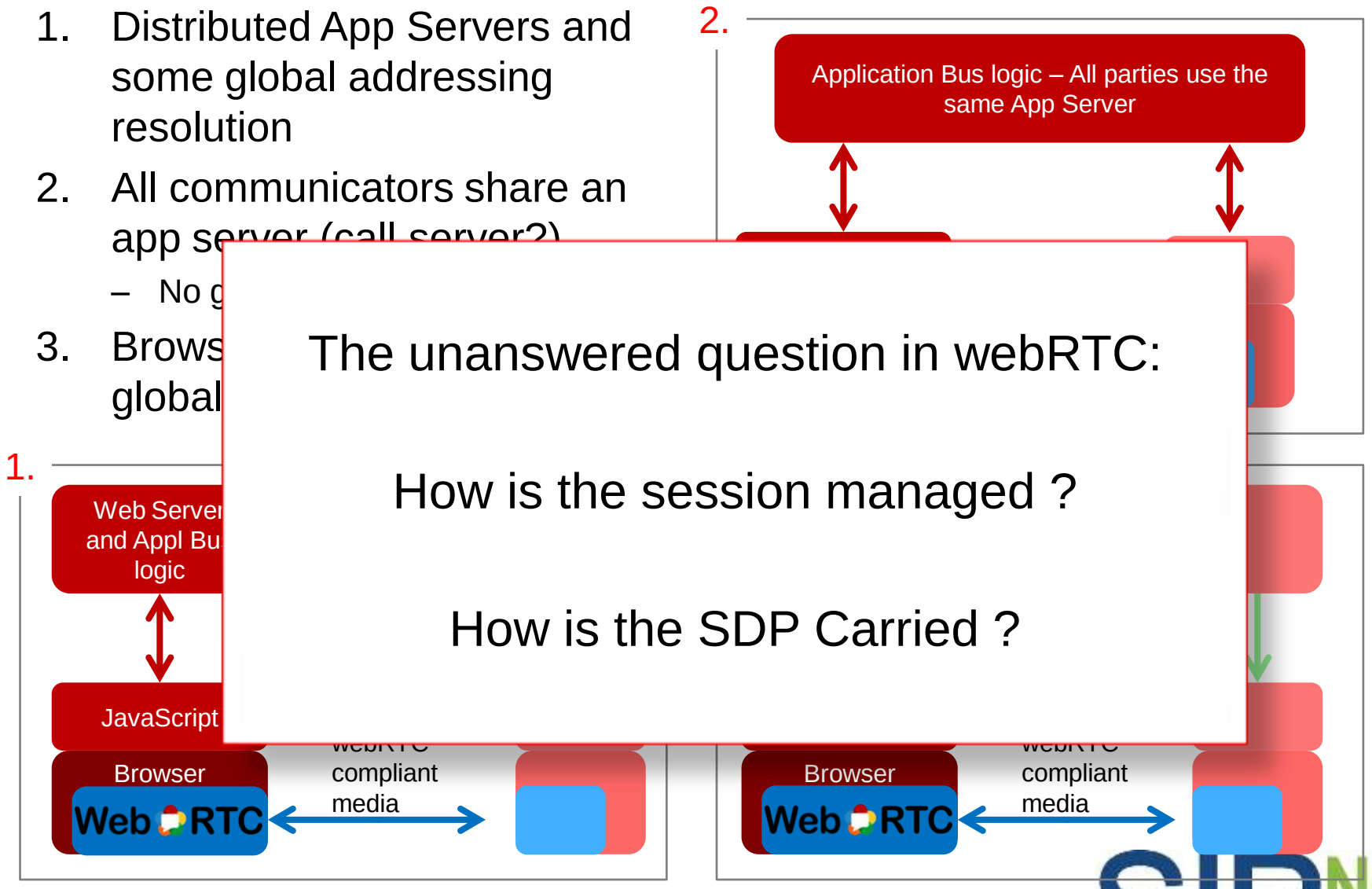
All part of the same stream

3 separate video streams

ICE candidate addresses and credentials

Back to the topic of Sessions

1. Distributed App Servers and some global addressing resolution
2. All communicators share an app server (call server?)
   - No g
3. Brows
   global



The unanswered question in webRTC:

How is the session managed ?

How is the SDP Carried ?

SIP (no need for details in this audience)

F1 INVITE Alice -> atlanta.com proxy

INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=19
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142

- Comprehensive
- Well practiced at Offer / Answer
- Aligns well with existing communications networks
- Proven capabilities

REpresentational State Transfer (REST)

REST is an approach or technique, NOT a specification or protocol

Each URL addresses a Resource and e
that resource.

```
GET http://www.example.com/phonebook/Us
```

```
POST /phonebook HTTP/1.1
Host: www.example.com
Content-Type: application/soap+xml; cha
Content-Length: 299
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/
soap:encodingStyle="http://www.w3.org/
 <soap:Body pb="http://www.example.com/
  <pb:GetUserDetails>
   <pb:UserID>12345</pb:UserID>
  </pb:GetUserDetails>
 </soap:Body>
</soap:Envelope>
```

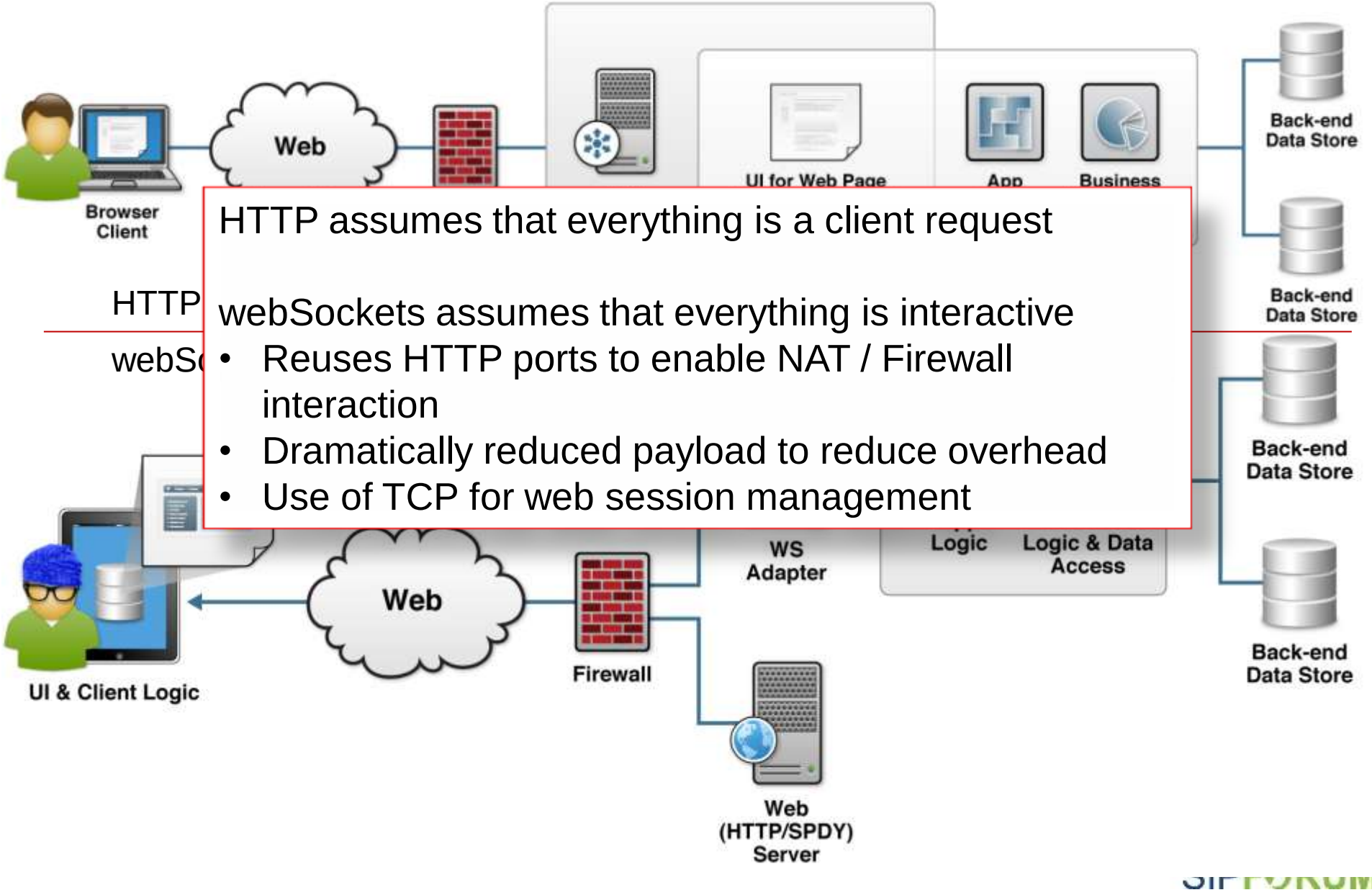
- Not a Standard
- Integrates well with web environments
- Easily specialized
- Familiar tool of a large development community
- Simple
- Not inherently Peer to Peer

XMPP / Jingle

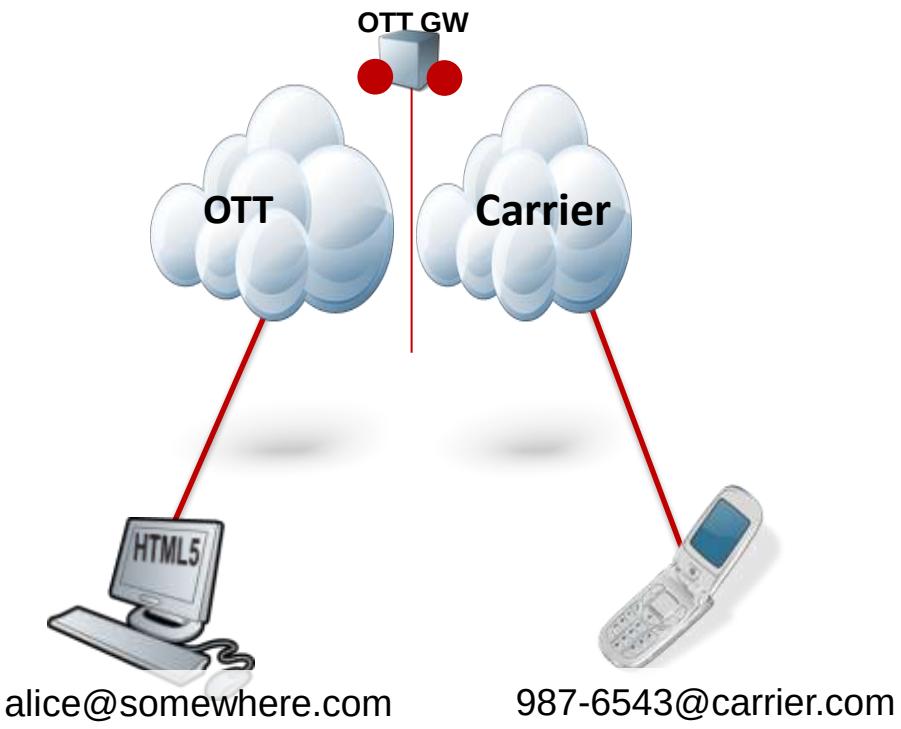
```
<iq from='juliet@capulet.lit/balcony'
    id='rc61n59s'
    to='romeo@montague.lit/orchard'
    type='set'>
  <jingle xmlns='urn:xmpp:jingle:1'
          action='session-accept'
          responder='juliet@capulet.lit/bal
          sid='a73sjjvkla37jfea'>
    <content creator='initiator' name='thi
      <description xmlns='urn:xmpp:jingle
      <transport xmlns='urn:xmpp:jingle:t
    </content>
  </jingle>
</iq>
```

- In common use in OTT environments
- Aligns with webRTC choices like ICE
- Established Open Source assets

HTTP vs. webSockets



Tying it all together (Carrier version)



The chaos of freedom interworked with the complexity of years of experience

- Isolation at **both** the Media and Signaling layer
- SIP Compliance on one side, flexibility on the other
- Full compliance to a webRTC media path facing the OTT
  - Codec diversity
  - ICE NAT Traversal
  - SRTP
  - Media Muxing
  - Etc…
- Full compliance to a regulated SIP environment in the Carrier
  - Legal Intercept
  - Dial Plan and addressing
  - IMS Data and Authentication model

A difference of Perspective

Carrier

- Session
  - SIP Compliant
  - Voice, Video, Chat, Video
  - Full Telephony State
  - Regulatory compliance
  - Universal rigid addressing
  - Session based
- Application
  - Common
  - Universal
  - Consistent
  - telephony

OTT

- Session
  - No a-priori rules
  - Voice, Video, Chat, Screen-share, Pictures, co-browse, …
  - Community model addressing
  - Unregulated
  - Often conference based
- Application
  - No Rules
  - Specialized subsets, and integrated communications
  - Different models (wall, timeline, tweet ,,,)

A mature solution ?

- Standards Completion
- Uniform Adoption
- QoS Assurance
- ICE and “post dial delay”
- Optimal codecs for diverse networks
- Inter-domain communication
  - Addressing
  - Application Models
- etc.

References

SIP NOC 2013
SIPFORUM

Additional WebRTC resources

- WebRTC
  - IETF rtcweb working group charter: http://datatracker.ietf.org/wg/rtcweb/charter/
  - W3C Real Time Communications wiki: http://www.w3.org/2011/04/webrtc/wiki/Main_Page
  - Web RTC Security: https://wiki.mozilla.org/Security/Discussions/WebRTC
  - Web RTC Open Source Project: http://www.webrtc.org/
  - Web RTC blog: http://bloggeek.me
- WebRTC examples
  - Vidtel WebRTC Video Conferencing http://youtu.be/99QbZmA4XNc
  - Drum - real-time web meeting solution with an integrated audio conferencing capability http://thisisdrum.com
  - Twelephone - HTML5 WebRTC browser-based voice and video telephone http://www.youtube.com/watch?v=I-2FKvItckQ&feature=player_embedded
  - Zingaya - one-click web call service http://blog.zingaya.com/2012/11/26/zingaya's-one-click-web-call-service-powered-by-webrtc-is-now-live/
  - Voxeo Labs and Solaiemes Bring WebRTC Video Calls to Existing Mobile Numbers - http://voxeolabs.com/2012/11/voxeo-labs-and-solaiemes-bring-webrtc-video-calls-to-existing-mobile-numbers/

References

IETF RFCs		
RFC 2327	SDP: Session Description Protocol	http://www.ietf.org/rfc/rfc2327.txt
RFC 3264	An Offer/Answer Model with SDP	http://www.ietf.org/rfc/rfc3264.txt
RFC 768	User Datagram Protocol	http://www.ietf.org/rfc/rfc768.txt
RFC 3550	RTP: A Transport Protocol for Real-Time Applications	http://www.ietf.org/rfc/rfc3550.txt
RFC 5245	Interactive Connectivity Establishment (ICE):	http://tools.ietf.org/html/rfc5245
RFC 6455	The WebSocket Protocol	http://tools.ietf.org/html/rfc6455
IETF RTCWEB Drafts		
draft-ietf-rtcweb-audio		http://tools.ietf.org/wg/rtcweb/draft-ietf-rtcweb-audio/
draft-ietf-rtcweb-data-channel		http://tools.ietf.org/wg/rtcweb/draft-ietf-rtcweb-data-channel/
draft-ietf-rtcweb-jsep		http://tools.ietf.org/wg/rtcweb/draft-ietf-rtcweb-jsep/
draft-ietf-rtcweb-overview		http://tools.ietf.org/wg/rtcweb/draft-ietf-rtcweb-overview/
draft-ietf-rtcweb-qos		http://tools.ietf.org/wg/rtcweb/draft-ietf-rtcweb-qos/
draft-ietf-rtcweb-rtp-usage		http://tools.ietf.org/html/draft-ietf-rtcweb-rtp-usage-05
draft-ietf-rtcweb-security-arch		http://tools.ietf.org/wg/rtcweb/draft-ietf-rtcweb-security-arch/
draft-ietf-rtcweb-use-cases-and-requirements		http://tools.ietf.org/wg/rtcweb/draft-ietf-rtcweb-use-cases-and-requirements/
W3C webRTC Documents		
Real-time Communication Between Browsers		http://dev.w3.org/2011/webrtc/editor/webrtc.html
Media Capture and Streams		http://dev.w3.org/2011/webrtc/editor/getusermedia.html

And a good book on the topic:
WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web
Alan B. Johnston and Daniel C Burnett
http://webrtcbook.com/