

Introducción a WebRTC



Agenda



Introducción

- Qué es WebRTC
- Ventajas, desventajas y características



WebRTC APIs

- APIs disponibles
- Proceso de conexión entre pares
- Desafíos y sus soluciones



Soporte de WebRTC

- Herramientas de soporte

Esta presentación está basada, en parte, en material preparado por Mauricio Gonzalez Nappa para la asignatura “Multimedia sobre IP” del Instituto de Ingeniería Eléctrica de la Facultad de Ingeniería de la UdelAR, en setiembre 2020



Introducción



*“Voice is just another
Java Script
application”*

Henning Schulzrinne



Henning Schulzrinne

- Profesor titular y director del Internet Real-Time Laboratory, de la Universidad de Columbia
- Es coautor de SIP, RTP y RTSP, protocolos clave para la comunicación de audio y vídeo sobre Internet



Comunicaciones en tiempo real para la web

- WebRTC permite agregar capacidades de comunicación en tiempo real a aplicaciones, basado en un estándar abierto.
- Admite el envío de video, voz y datos genéricos entre pares, lo que permite crear soluciones de comunicación multimedia.
- La tecnología está disponible en todos los navegadores modernos, así como en clientes nativos de las principales plataformas de comunicaciones.
- Las tecnologías detrás de WebRTC se implementan como un estándar web abierto y están disponibles como APIs de JavaScript regulares en todos los principales navegadores.
- El proyecto WebRTC es de código abierto y cuenta con el apoyo de Apple, Google, Microsoft y Mozilla, entre otros.



Fuente: <https://webrtc.org/>



Ventajas de WebRTC

- Utiliza componentes “nativos” de los navegadores:
SIN necesidad de descargar software o plugins
- Es open source, gratuito y estandarizado
- Reutiliza protocolos existentes (por ejemplo RTP y SDP)
- Es seguro por diseño

Web  RTC



Desafíos de WebRTC

- El “debugging” en caso de problemas puede ser complicado
 - Tanto la señalización como el medio está encriptado.
 - No hay muchas herramientas disponibles para dar soporte.
- Si “la voz es simplemente otra aplicación JS” ... puede adolecer de los problemas de cualquier aplicación web: Su performance depende del PC o dispositivo, y de la carga y uso de otras aplicaciones concurrentes.
- Los usuarios pueden estar dentro de redes diferentes, sin conexión entre sí.
 - En este caso, deben poder conectarse a través de Internet...
....pero están “detrás de Firewalls con NAT” que tienen restricciones.

Web  RTC



WebRTC APIs



WebRTC APIs

MediaStream (getUserMedia)

Accede a streams de audio/video.
Interfaz con micrófonos y cámara

RTCPeerConnection

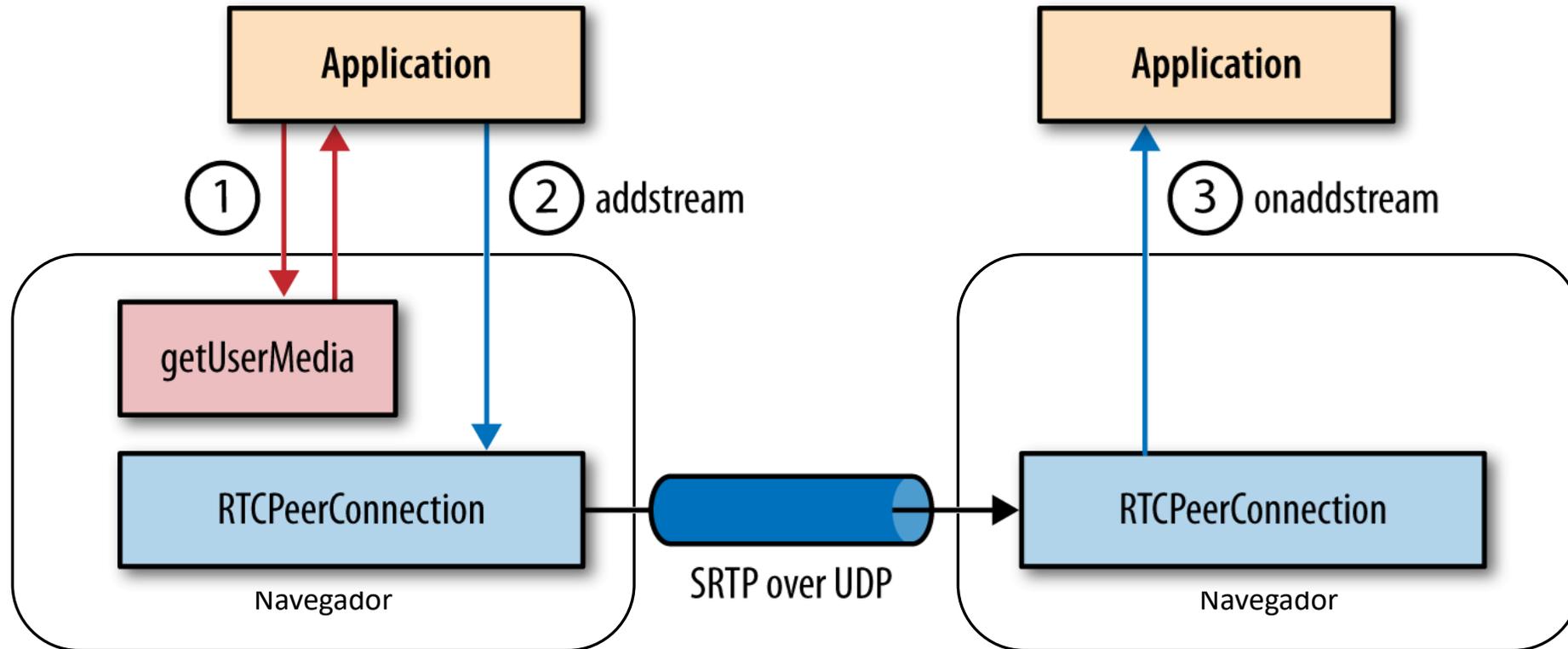
Gestiona las comunicaciones de audio/video.
Cuenta con funciones de encriptación y gestión
de ancho de banda.

RTCDataChannel

Gestiona comunicaciones de datos.



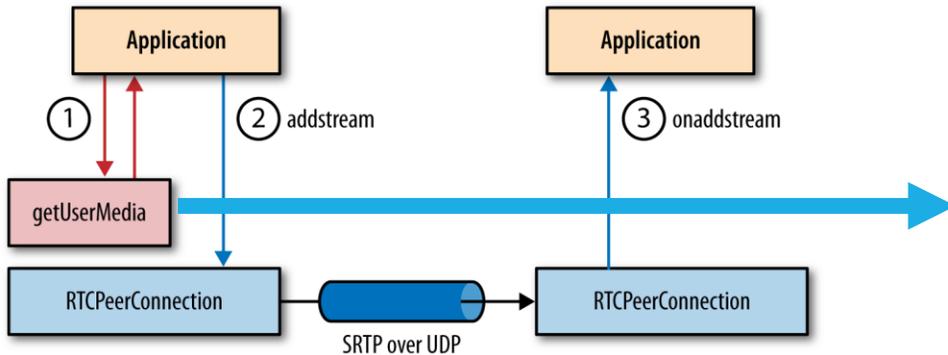
WebRTC APIs



Tomado de: <https://blog.csdn.net/fanbird2008/article/details/18623141>



getUserMedia/GetDisplayMedia Requests



Se obtiene información de las cámaras y micrófonos del dispositivo, y una asociación lógica que permite usarlos

Filter by origin including

Caller origin: <https://soporteuc.isbel.com.uy>
Caller process id: 32260

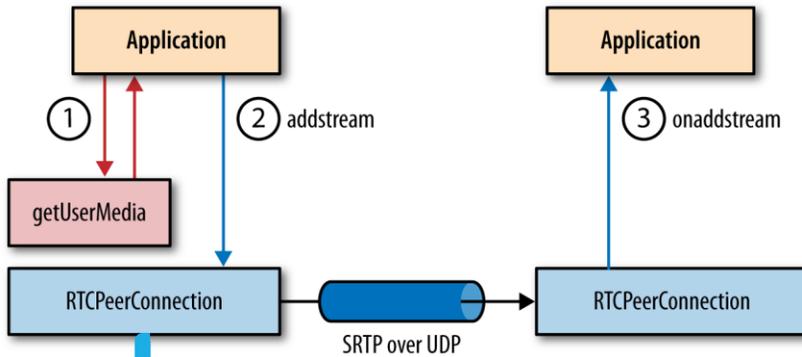
getUserMedia call
Time: 08:43:10 GMT-0300 (hora estándar de Uruguay)
Audio constraints: true
Video constraints: true

getUserMedia result
Time: 08:43:16 GMT-0300 (hora estándar de Uruguay)
Stream id: 163630c6-5569-49f7-85ab-5f6f730bd87e

Audio track: id:c69a46f5-86da-44ec-ab97-2d7c7cbd3c03
label: Predeterminado - Micrófono (Avaya HC050) (2e7e:071c)

Video track: id:7390263a-1b9a-4cfa-b212-6bb0718bba83 label: Avaya HC050 (2e7e:071c)





Se establece una conexión con
entre origen y destino y se
mantiene durante toda la
sesión
¡Este proceso es complejo!

(detalles más adelante)

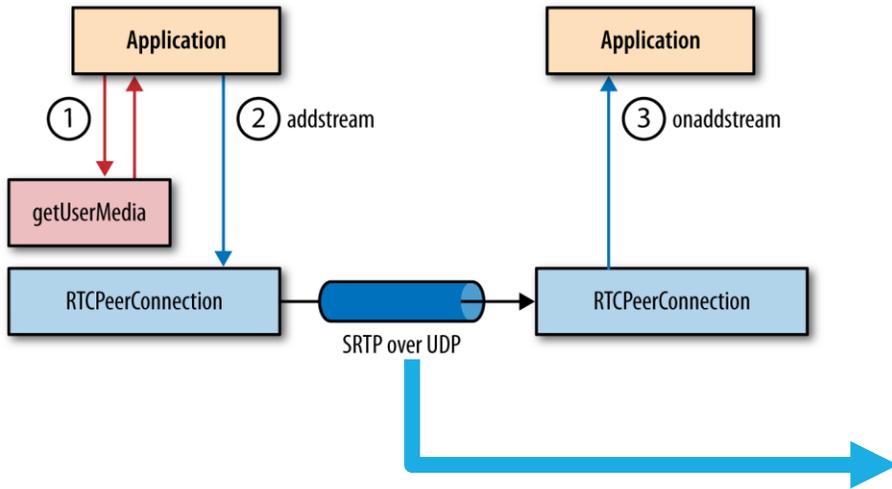
Time	Event
13/4/2023, 8:47:55	▶ transceiverAdded
13/4/2023, 8:47:55	▶ createOffer
13/4/2023, 8:47:55	negotiationneeded
13/4/2023, 8:47:55	▶ createOfferOnSuccess (type: "offer", 2 sections)
13/4/2023, 8:47:55	▶ setLocalDescription (type: "offer", 2 sections)
13/4/2023, 8:47:55	setLocalDescriptionOnSuccess
13/4/2023, 8:47:55	▶ signalingstatechange
13/4/2023, 8:47:55	▶ transceiverModified
13/4/2023, 8:47:55	▶ icegatheringstatechange
13/4/2023, 8:47:55	▶ icecandidate(sdpMid: 0, sdpMLineIndex: 0, type: host)
13/4/2023, 8:47:55	▶ icecandidate(sdpMid: 0, sdpMLineIndex: 0, type: host)
13/4/2023, 8:47:55	▶ icecandidate(sdpMid: 0, sdpMLineIndex: 0, type: host)
13/4/2023, 8:47:56	▶ icecandidate(sdpMid: 0, sdpMLineIndex: 0, type: host)
13/4/2023, 8:47:56	▶ icecandidate(sdpMid: 0, sdpMLineIndex: 0, type: host)
13/4/2023, 8:47:56	▶ icecandidate(sdpMid: 0, sdpMLineIndex: 0, type: srflx)
13/4/2023, 8:48:06	▶ setRemoteDescription (type: "answer", 2 sections)
13/4/2023, 8:48:06	▶ iceconnectionstatechange
13/4/2023, 8:48:06	setRemoteDescriptionOnSuccess
13/4/2023, 8:48:06	▶ signalingstatechange
13/4/2023, 8:48:06	▶ transceiverModified
13/4/2023, 8:48:06	▶ connectionstatechange
13/4/2023, 8:48:06	▶ iceconnectionstatechange
13/4/2023, 8:48:06	▶ icegatheringstatechange
13/4/2023, 8:48:06	▶ connectionstatechange
13/4/2023, 8:48:25	▶ createOffer
13/4/2023, 8:48:25	▶ createOfferOnSuccess (type: "offer", 2 sections)
13/4/2023, 8:48:25	▶ setLocalDescription (type: "offer", 2 sections)
13/4/2023, 8:48:25	setLocalDescriptionOnSuccess
13/4/2023, 8:48:25	▶ signalingstatechange
13/4/2023, 8:48:25	▶ setRemoteDescription (type: "answer", 2 sections)
13/4/2023, 8:48:25	setRemoteDescriptionOnSuccess
13/4/2023, 8:48:25	▶ signalingstatechange
13/4/2023, 8:48:35	▶ icecandidateerror
13/4/2023, 8:48:35	▶ icecandidateerror
13/4/2023, 8:52:22	close
13/4/2023, 8:52:22	▶ connectionstatechange

Oferta de
capacidades locales
(códecs, etc.) e
infraestructura de
servidores de
medios a usar

Recepción de
capacidades remotas
(códecs, etc.) y
acuerdo de
infraestructura de
servidores de
medios a usar

Establecimiento de
la comunicación

Fin de la
comunicación



Intercambio de medios (audio, video), en forma segura (encriptada) entre origen y destino

Llamada 5.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

Aplique un filtro de visualización ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
6394	153.754463	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17698, Time=4022627056
6395	153.760771	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x972F56AE, Seq=13215, Time=2223298274
6396	153.774420	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17699, Time=4022627216
6397	153.780653	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x972F56AE, Seq=13216, Time=2223298434
6398	153.794387	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17700, Time=4022627376
6399	153.800660	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x972F56AE, Seq=13217, Time=2223298594
6400	153.814437	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17701, Time=4022627536
6401	153.820805	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x972F56AE, Seq=13218, Time=2223298754
6402	153.834448	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17702, Time=4022627696
6403	153.840579	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x972F56AE, Seq=13219, Time=2223298914
6404	153.854472	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17703, Time=4022627856
6405	153.860398	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x972F56AE, Seq=13220, Time=2223299074
6406	153.874479	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17704, Time=4022628016
6407	153.880885	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x972F56AE, Seq=13221, Time=2223299234
6408	153.894385	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G.711 PCMA, SSRC=0x4D6CA111, Seq=17705, Time=4022628176

> Frame 6162: 224 bytes on wire (1792 bits), 224 bytes captured (1792 bits) on interface \Device\NPF_{07E40D70-66AE-4D27-AFE7-305F59}

> Ethernet II, Src: JuniperN_af:46:c1 (64:87:88:af:46:c1), Dst: Dell_18:6d:e4 (d8:9e:f3:18:6d:e4)

> Internet Protocol Version 4, Src: 10.0.3.15 (10.0.3.15), Dst: 10.15.115.14 (10.15.115.14)

> User Datagram Protocol, Src Port: 14374 (14374), Dst Port: 49222 (49222)

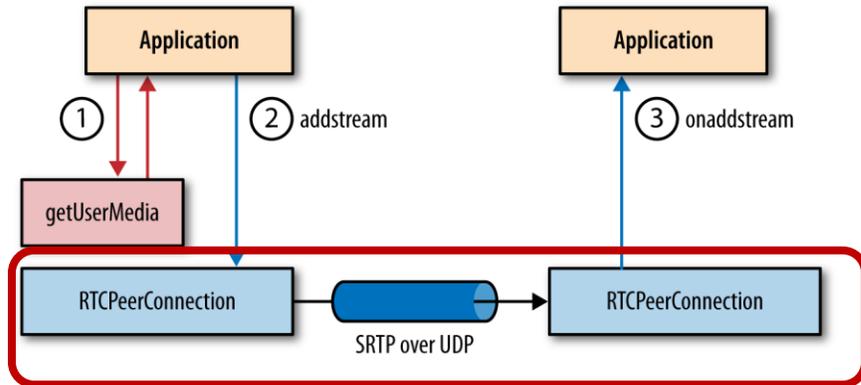
> Real-Time Transport Protocol

- > [Stream setup by DTLS-SRTP (frame 2132)]
 - 10.. = Version: RFC 1889 Version (2)
 - ..0. = Padding: False
 - ...0 = Extension: False
 - 0000 = Contributing source identifiers count: 0
 - 0... = Marker: False
 - Payload type: ITU-T G.711 PCMA (8)
 - Sequence number: 17604
 - [Extended sequence number: 83140]
 - Timestamp: 4022612016
 - Synchronization Source identifier: 0x4d6ca111 (1298964753)
 - SRTP Encrypted Payload: b67478b83edb04779e33bd004c0e9f5fb0d96365c2d318080069ffa0014cef077b885e08...
 - SRTP Auth Tag: 9fd141e072b77e076d54

Paquetes de audio (por ejemplo G.711 A-law) sobre UDP / SRTP (encriptados)

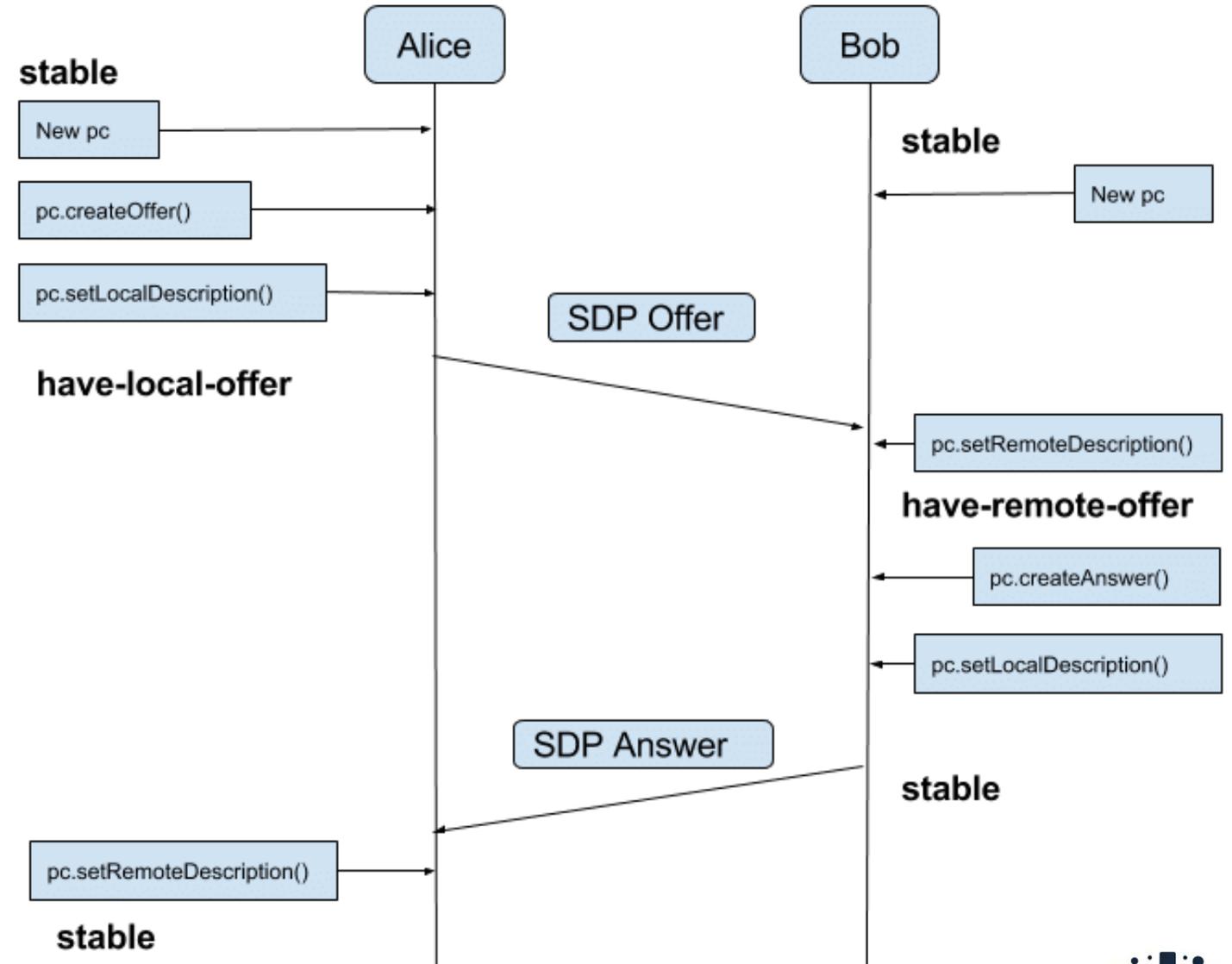


Oferta e intercambio de capacidades entre origen y destino



Volvamos al RTCPeerConnection....

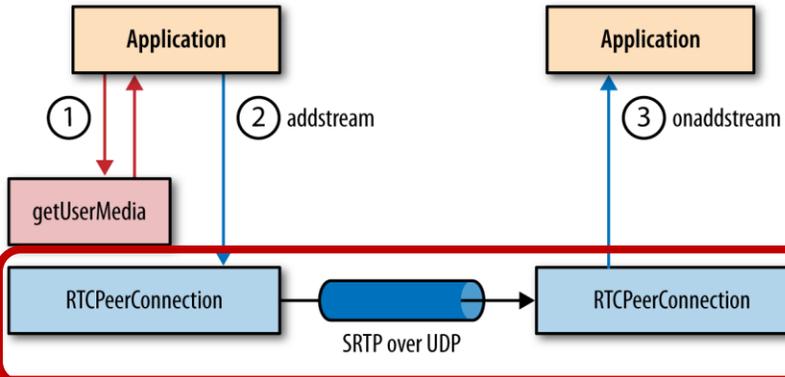
¡Origen y destino deben ponerse de acuerdo en varias cosas!



Tomado de: <https://www.callstats.io/blog/2017/12/12/signaling-state-changes>



Oferta e intercambio de capacidades entre origen y destino

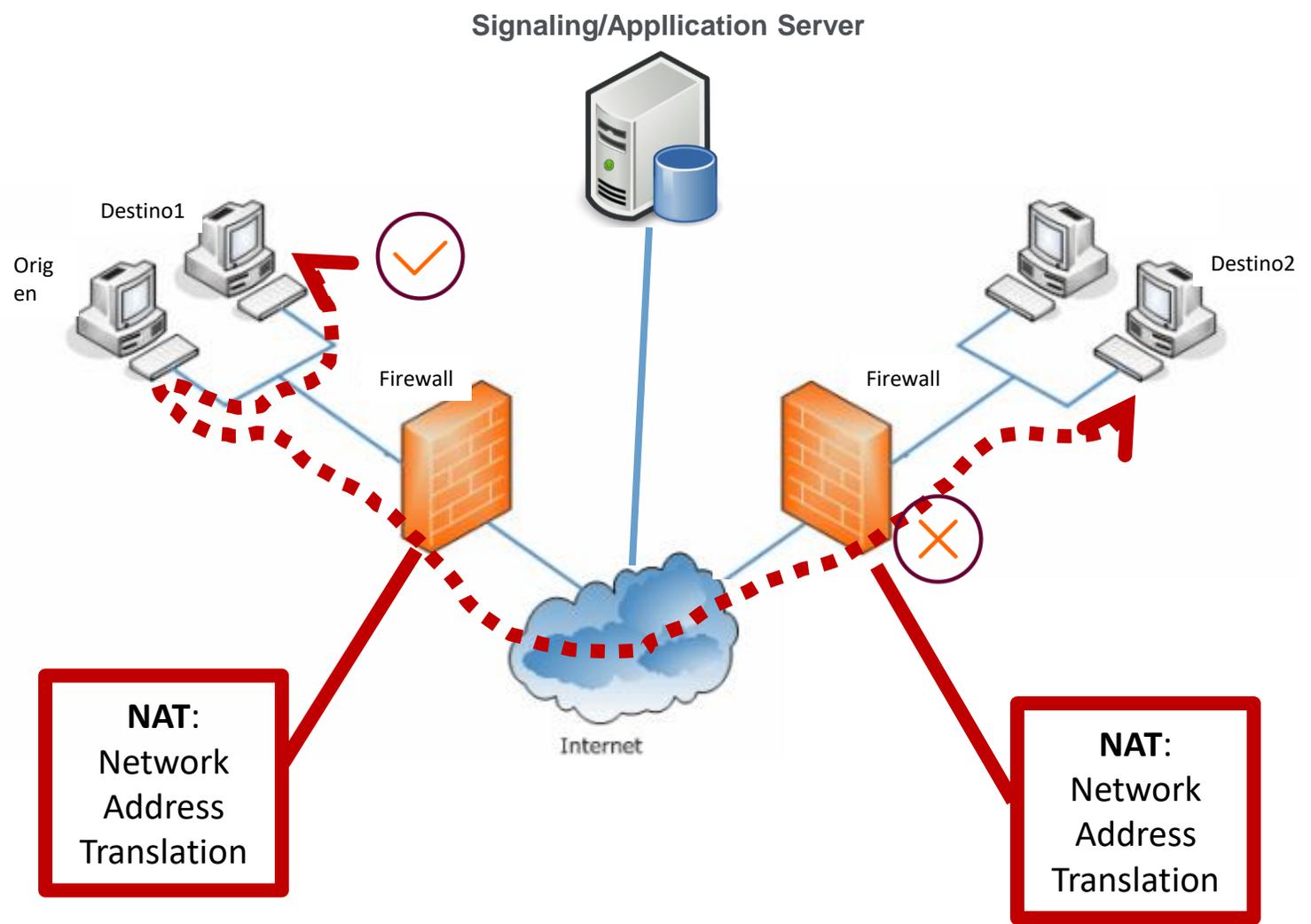
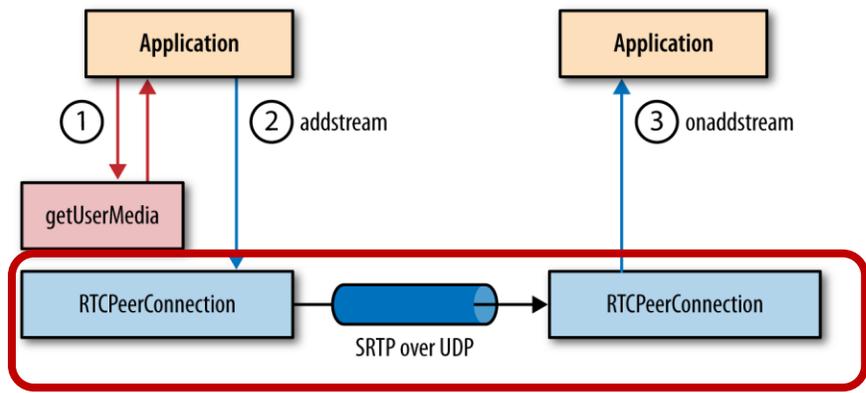


```

▼ setRemoteDescription (type: "answer", 2 sections)
Copy description to clipboard
▼ v=0 (4 more lines)
o=root 1035137076 1035137076 IN IP4 201.217.144.18
s=Asterisk PBX 13.38.1
c=IN IP4 201.217.144.18
t=0 0
m=audio 12610 RTP/SAVPF 0 8 15 (16 more lines)
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:126 telephone-event/8000
a=ice-ufrag:6340a96d093368b75187ecc50c4c5c4b
a=ice-pwd:53ba195f4efcec0a7ca19d9f4cf37a05
a=candidate:Hc0a800be 1 UDP 2130706431 192.168.0.190 12818 typ host
a=candidate:Sc9d99012 1 UDP 1694498815 201.217.144.18 12818 typ srflx raddr 192.168.0.190 rport 12818
a=candidate:Hc0a800be 2 UDP 2130706430 192.168.0.190 12819 typ host
a=candidate:Sc9d99012 2 UDP 1694498814 201.217.144.18 12819 typ srflx raddr 192.168.0.190 rport 12818
a=connection:new
a=setup:active
a=fingerprint:SHA-256 E6:31:40:56:B1:08:F5:27:98:A2:D9:9A:56:66:9B:68:12:E5:57:9B:7B:EB:96:08:F8:5F:FE
a=rtcp-mux
a=sendrecv
    
```

```

▼ setLocalDescription (type: "offer", 2 sections)
Copy description to clipboard
▼ v=0 (6 more lines)
o=- 1699581531373744841 2 IN IP4 127.0.0.1
s=-
t=0 0
a=group:BUNDLE 0
a=extmap-allow-mixed
a=msid-semantic: WMS 505c7576-6e0c-466c-ab9c-8554efe8fa55
▼ m=audio 9 UDP/TLS/RTP/SAVPF 111 63 9 0 8 13 110 126 (28 more lines) mid=0
c=IN IP4 0.0.0.0
a=rtcp:9 IN IP4 0.0.0.0
a=ice-ufrag:tdtx
a=ice-pwd:xp4J415iV2KcsvgMlGwiAEd2N
a=ice-options:trickle
a=fingerprint:sha-256 C9:AF:72:98:7E:92:C0:51:DE:78:C8:BB:46:A5:46:9D:AE:3E:D2:C0:D8:19:23:FE:BB:60:64:4F:FB:94:D2
a=setup:actpass
a=mid:0
a=extmap:1 urn:iETF:params:rtp-hdrext:ssrc-audio-level
a=extmap:2 http://www.webrtc.org/experiments/rtp-hdrext/abs-send-time
a=extmap:3 http://www.ietf.org/id/draft-holmer-rmcat-transport-wide-cc-extensions-01
a=extmap:4 urn:iETF:params:rtp-hdrext:sdes:mid
a=sendrecv
a=msid:505c7576-6e0c-466c-ab9c-8554efe8fa55 5bace4d4-2364-414a-aa22-135192d39008
a=rtcp-mux
a=rtpmap:111 opus/48000/2
a=rtcp-fb:111 transport-cc
a=fmtp:111 minptime=10;useinbandfec=1
a=rtpmap:63 red/48000/2
a=fmtp:63 111/111
a=rtpmap:9 G722/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:13 CN/8000
a=rtpmap:110 telephone-event/48000
a=rtpmap:126 telephone-event/8000
a=ssrc:3069412600 cname:gyLO6NYmtu21tYg6
a=ssrc:3069412600 msid:505c7576-6e0c-466c-ab9c-8554efe8fa55 5bace4d4-2364-414a-aa22-135192d39008
    
```



Además...

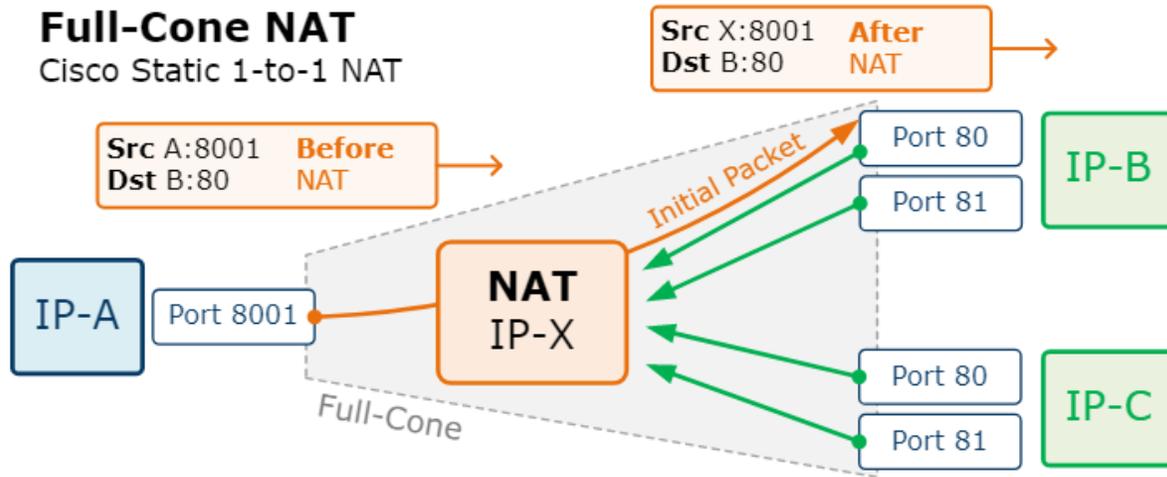
Origen y destino se deben “ver directamente” para poder intercambiar tráfico multimedia ... pero pueden estar en redes diferentes



Acerca de NAT

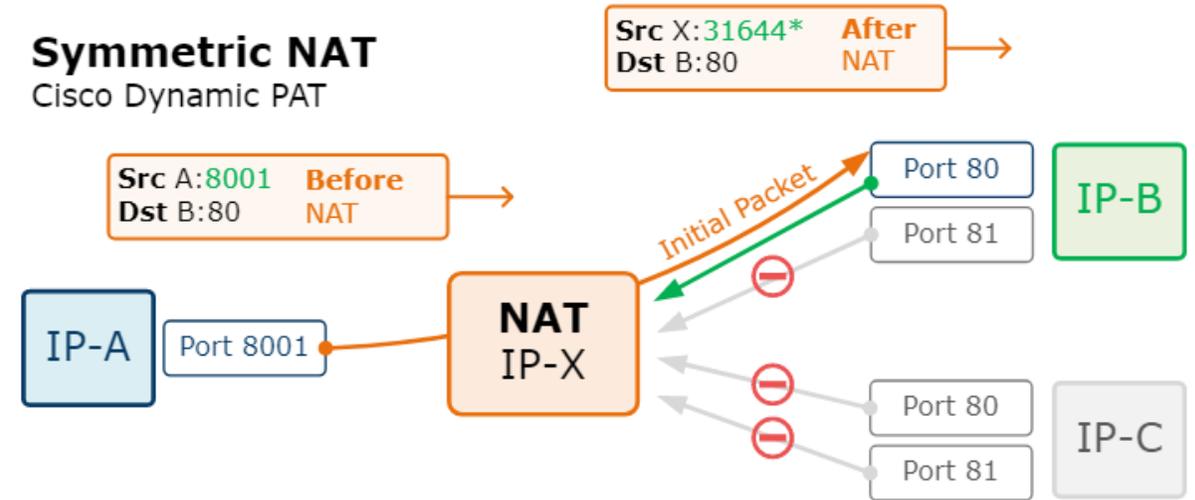
Full-Cone NAT

Cisco Static 1-to-1 NAT



Symmetric NAT

Cisco Dynamic PAT



Full-cone NAT

A una dirección interna (iAddr:iPort) se le asigna a una dirección externa (eAddr:ePort).

Todos los paquetes de iAddr:iPort se envían a través de eAddr:ePort.

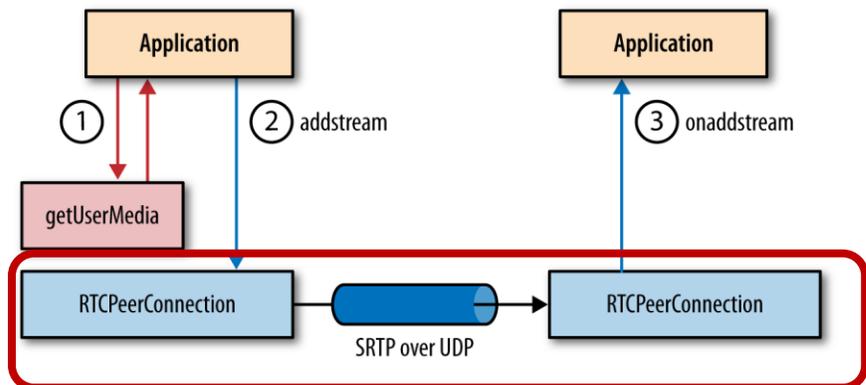
Cualquier host externo puede enviar paquetes a iAddr:iPort, enviando paquetes a eAddr:ePort.

Symetric NAT

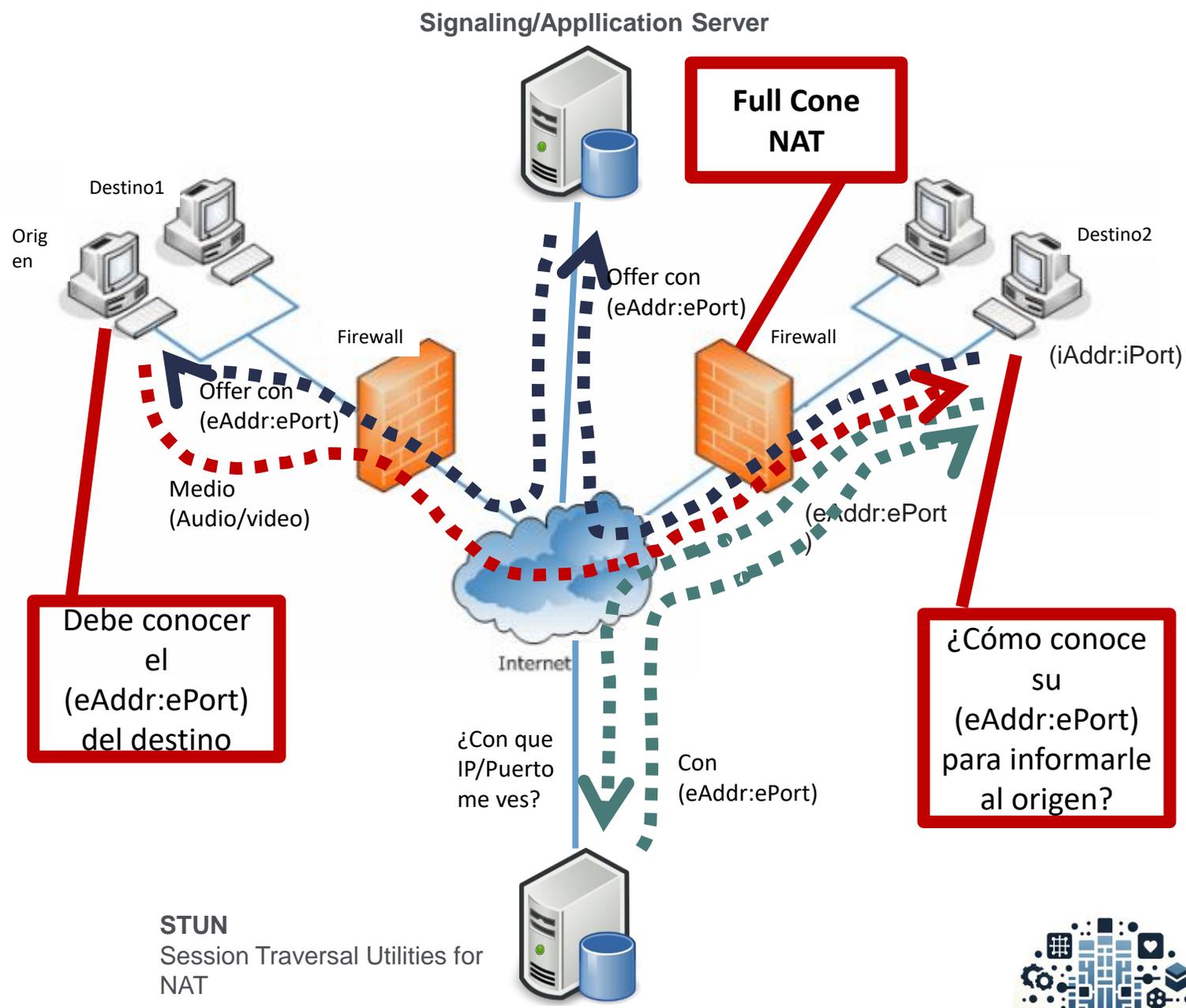
La combinación de una dirección IP interna más una dirección IP y puerto de destino se asigna a una única dirección IP y puerto de origen externo (eAddr:ePort). Si el mismo host interno envía un paquete (incluso con la misma dirección y puerto de origen) a un destino *diferente*, se utiliza una asignación *diferente* de eAddr:ePort. Solo un host externo que recibe un paquete de un host interno puede devolver un paquete (es decir, no se pueden iniciar sesiones desde el exterior).

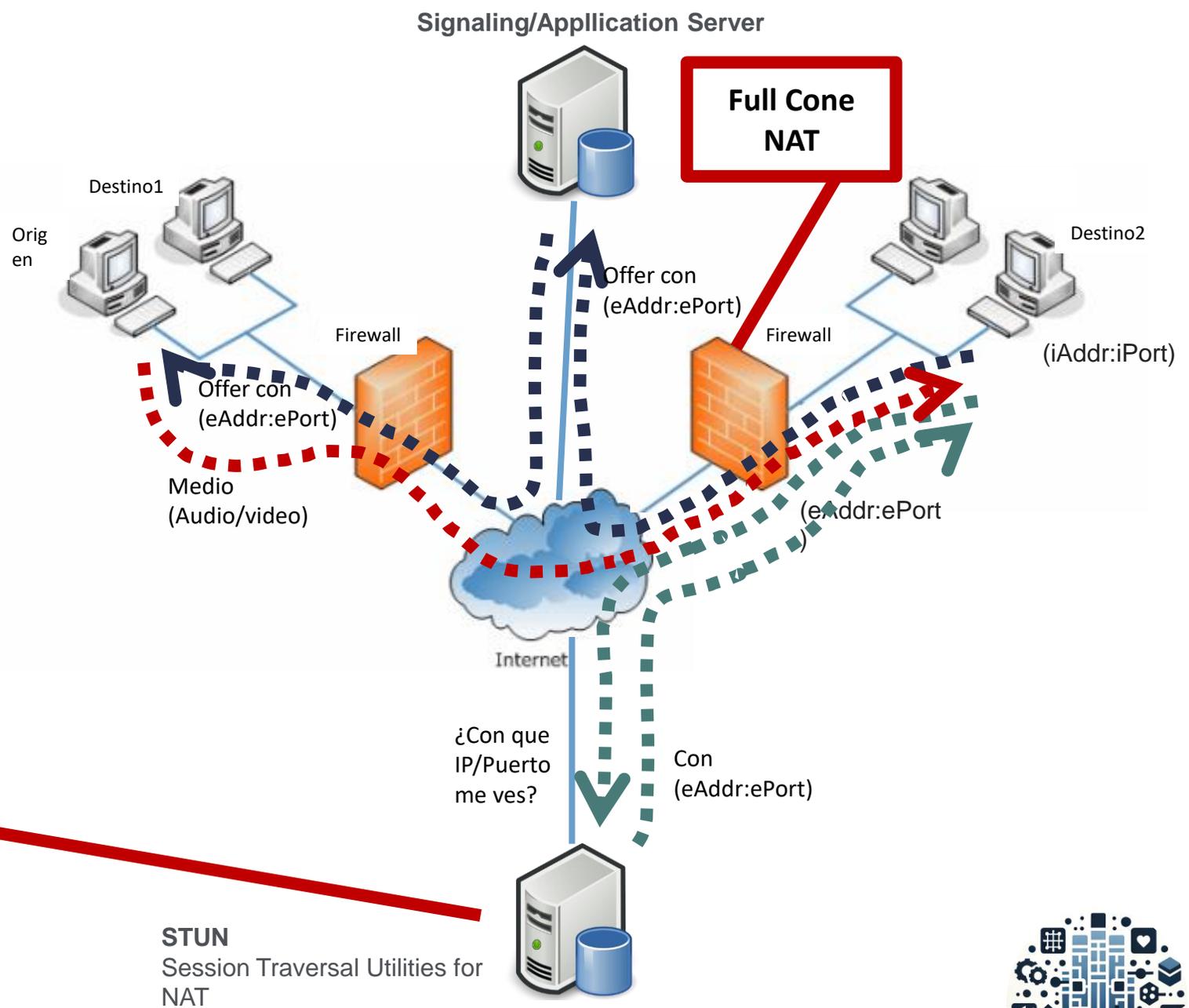
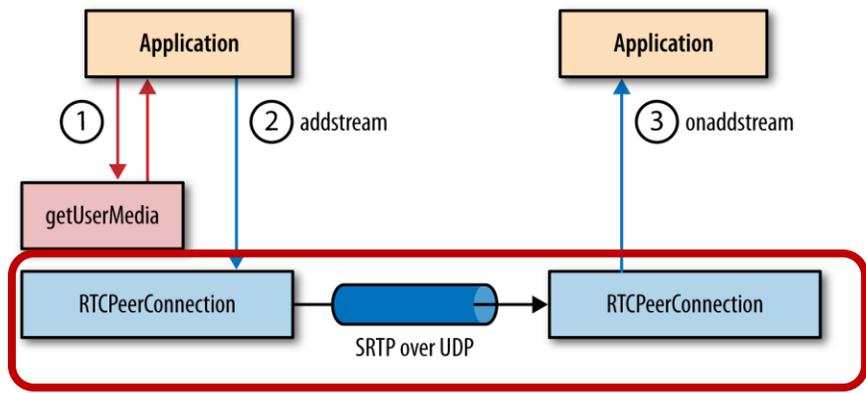
<https://www.networkacademy.io/ccie-enterprise/sdwan/tlocs-and-nat>
https://en.wikipedia.org/wiki/Network_address_translation





¿Cómo establecer sesiones de medios a través de un NAT?

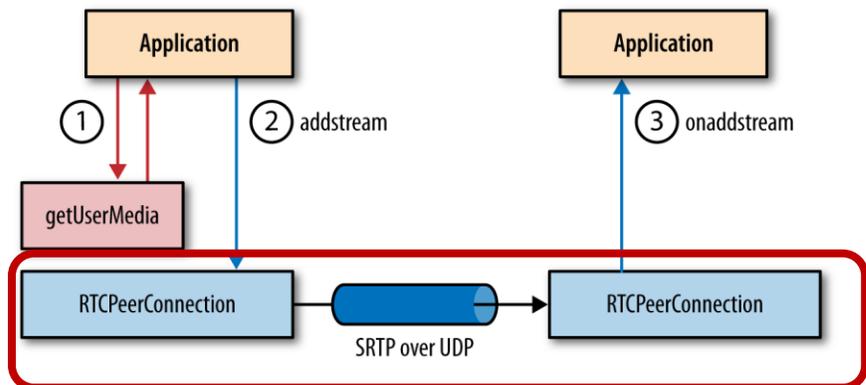




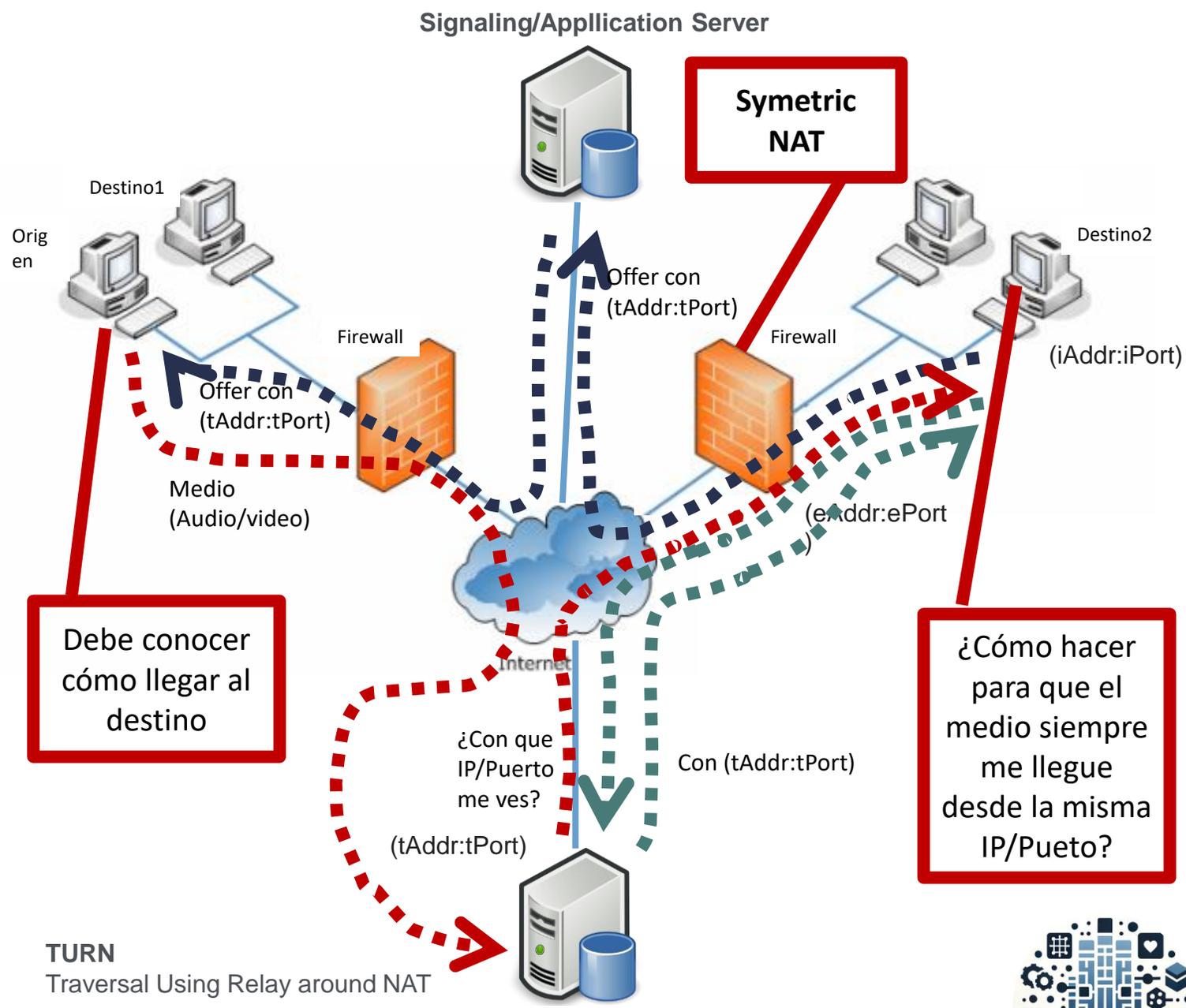
Funciona bien con un "Full Cone NAT"
 ¿pero qué pasa si hay un **Symetric NAT**?

STUN
 Session Traversal Utilities for NAT





¿Cómo establecer sesiones de audio a través de un Symetric NAT?



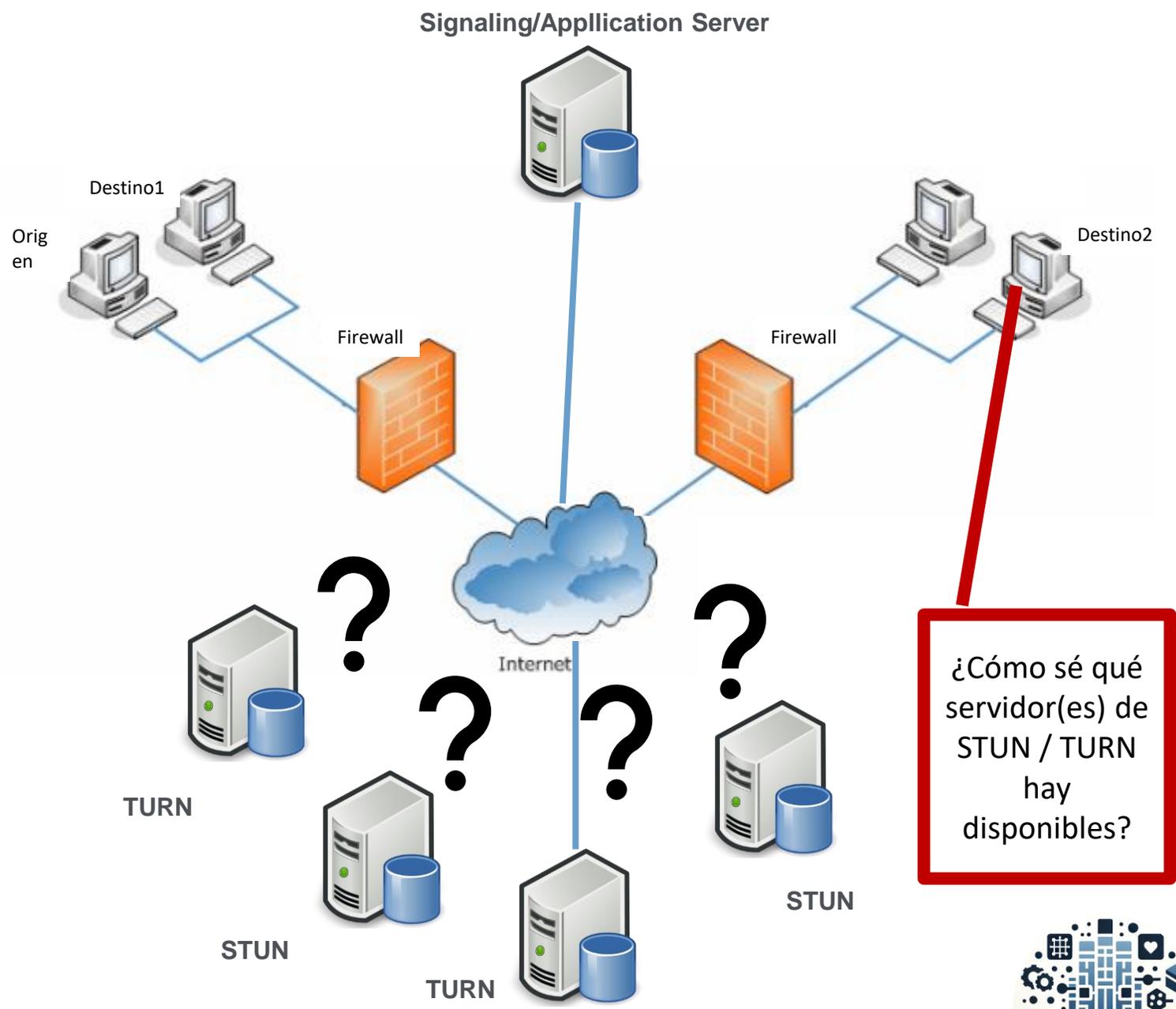
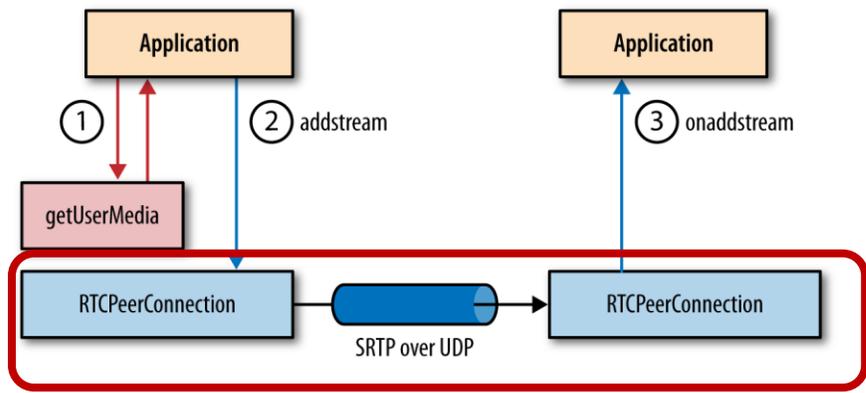
STUN: Session Traversal Utilities for NAT

Permite encontrar la dirección IP pública y el puerto de Internet asociado con el puerto local a través de NAT.

TURN: Traversal Using Relay around NAT

Permite a los clientes enviar tráfico multimedia a través de un servidor de retransmisión (TURN Server), que actúa como intermediario entre los clientes. Todo el tráfico hacia cada cliente parte siempre de la misma IP/puerto, y funciona aun con NAT simétrico. También informa la dirección IP pública y el puerto de Internet asociado con el puerto local a través de NAT.





¿Cómo sé qué servidor(es) de STUN / TURN hay disponibles?

ICE: Interactive Connectivity Establishment

ICE utiliza varios métodos para determinar la mejor ruta de red para una conexión en particular, incluso si los dispositivos se encuentran detrás de firewalls con NAT o proxies.

ICE utiliza STUN y TURN y preparar una lista de parejas IP/puerto “candidatos ICE” que pueden ser utilizados por cada extremo.

Se identifican tres tipos de candidatos:

- **host:** son directamente visibles por el usuario (están en la misma red).
Por ejemplo, las propias interfases de red del usuario.
- **server reflexive:** servidores STUN
- **relay:** servidores TURN

Las listas son intercambiadas entre origen y destino, y se prepara una matriz de posibilidades (todas los “candidatos ICE” de origen combinados con todos los “candidatos ICE” de destino)

Se ordenan y se elige la “ruta más económica”, con preferencia 1) host → 2) reflexive → 3) relay



ICE candidate grid

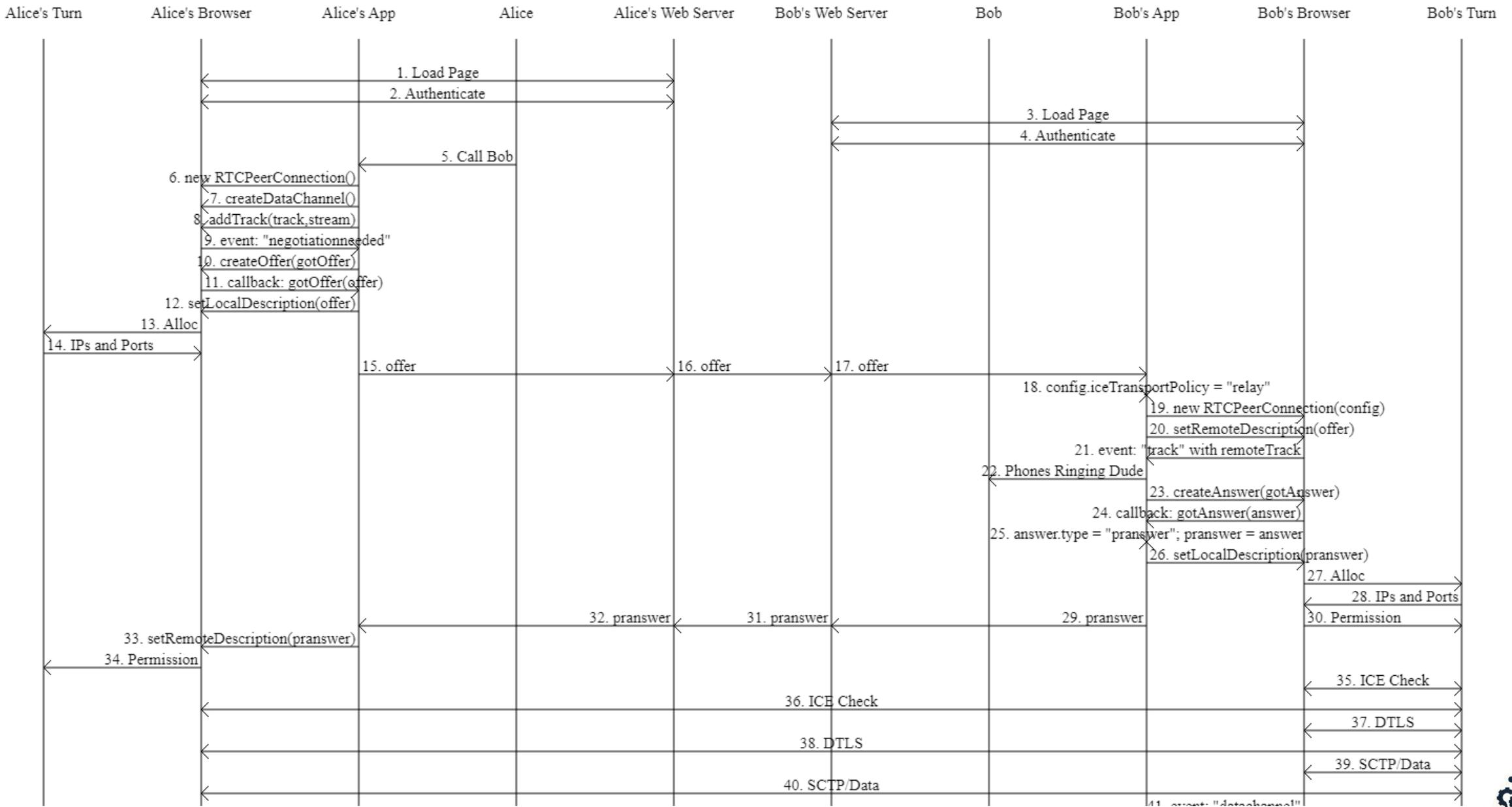
Candidate (pair) id	State / Candidate type	Network type / address	Port	Protocol / candidate type	(Pair) Priority	Bytes sent / received	STUN requests sent / responses received	STUN requests received / responses sent	RTT	Last data sent / received	Last update
CPdXi2YTiX_ lfrRn4ry	succeeded	vpn (VPN)		udp	0x7e7d1effe 000000	4290297 / 4280656	180 / 180	1 / 1	0.01s	9:19:43 / 9:19:43	9:19:44
IdXi2YTiX local-candidate		10.254.245.92	58813	host	0x7e7d1eff						
lfrRn4ry remote-candidate		192.168.0.190	16210	host	0x7effffff						
CPTi5gAVKy_ lfrRn4ry	in-progress	wifi		udp	0x7e7f1effe 000000	0 / 0	8 / 0				
ITi5gAVKy local-candidate		192.168.68.101	58811	host	0x7e7f1eff						
lfrRn4ry remote-candidate		192.168.0.190	16210	host	0x7effffff						
CPZSQ6AefB_ CMMYZCLK	succeeded	wifi		udp	0x647f1effca 000000	0 / 0	26 / 26				
IZSQ6AefB local-candidate		190.134.1.77	58811	srflx	0x647f1eff						
ICMMYZCLK remote-candidate		201.217.144.18	16210	srflx	0x64ffffff						
CPdXi2YTiX_ CMMYZCLK	in-progress	(VPN)		udp	0x64ffffffcfa 3c00	0 / 0	1 / 0	0 / 0			9:12:04
IdXi2YTiX local-candidate		10.254.245.92	58813	host	0x7e7d1eff						
ICMMYZCLK remote-candidate		201.217.144.18	16210	srflx	0x64ffffff						
CPyXG+p6qz_ CMMYZCLK	in-progress	wifi		udp	0x64ffffffcfc 3c00	0 / 0	8 / 0	0 / 0			9:12:09
lyXG+p6qz local-candidate		192.168.137.1	58812	host	0x7e7e1eff						
ICMMYZCLK remote-candidate		201.217.144.18	16210	srflx	0x64ffffff						

Tipos de candidatos:

host: son directamente visibles por el usuario (están en la misma red)

server reflexive (srflx): servidores STUN

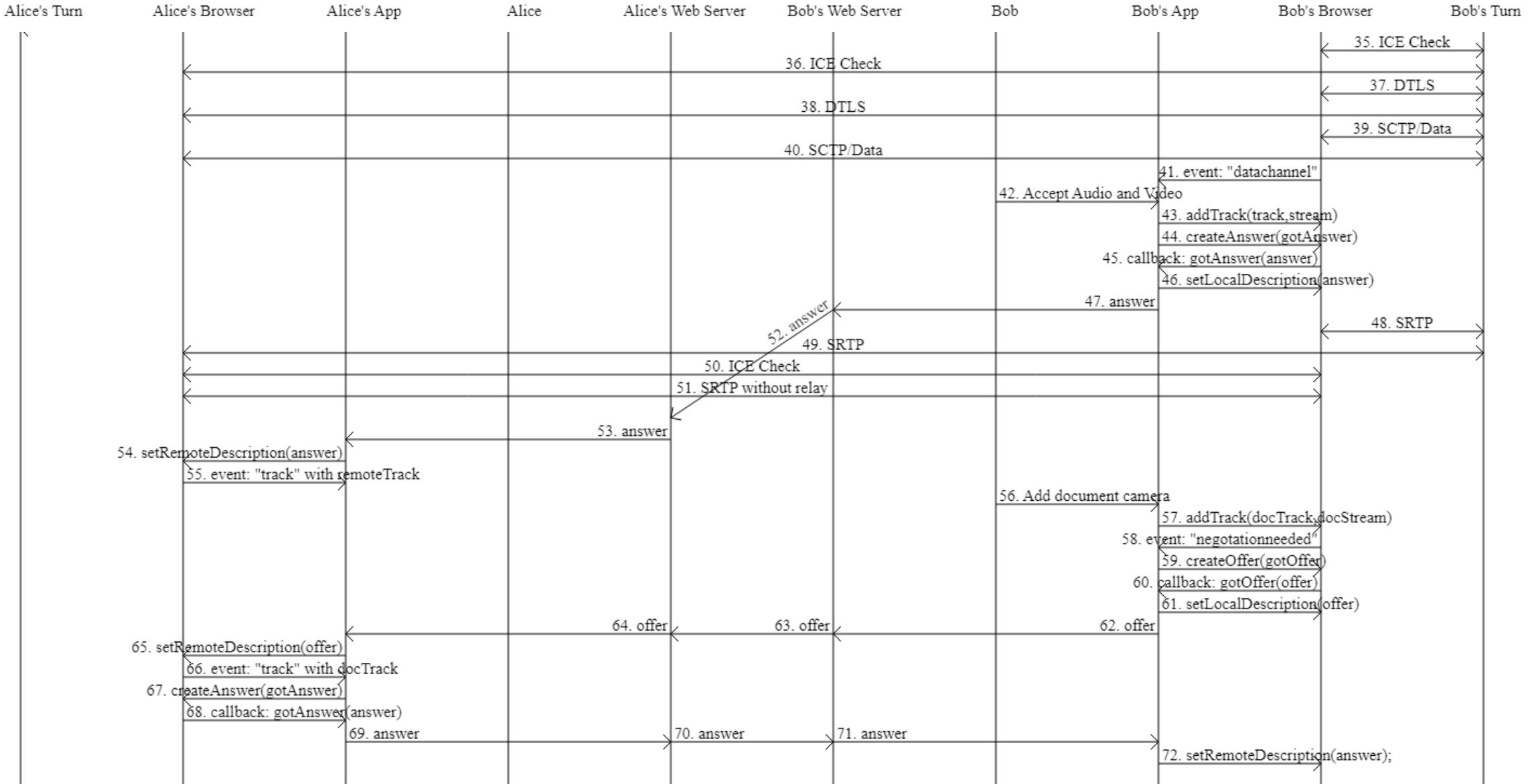
Relay: servidores TURN



<https://www.w3.org/TR/webrtc/>



Simple Call Flow



Por suerte...

todo resulta ser muy sencillo.....



Pero calma...

Hay herramientas para dar soporte...

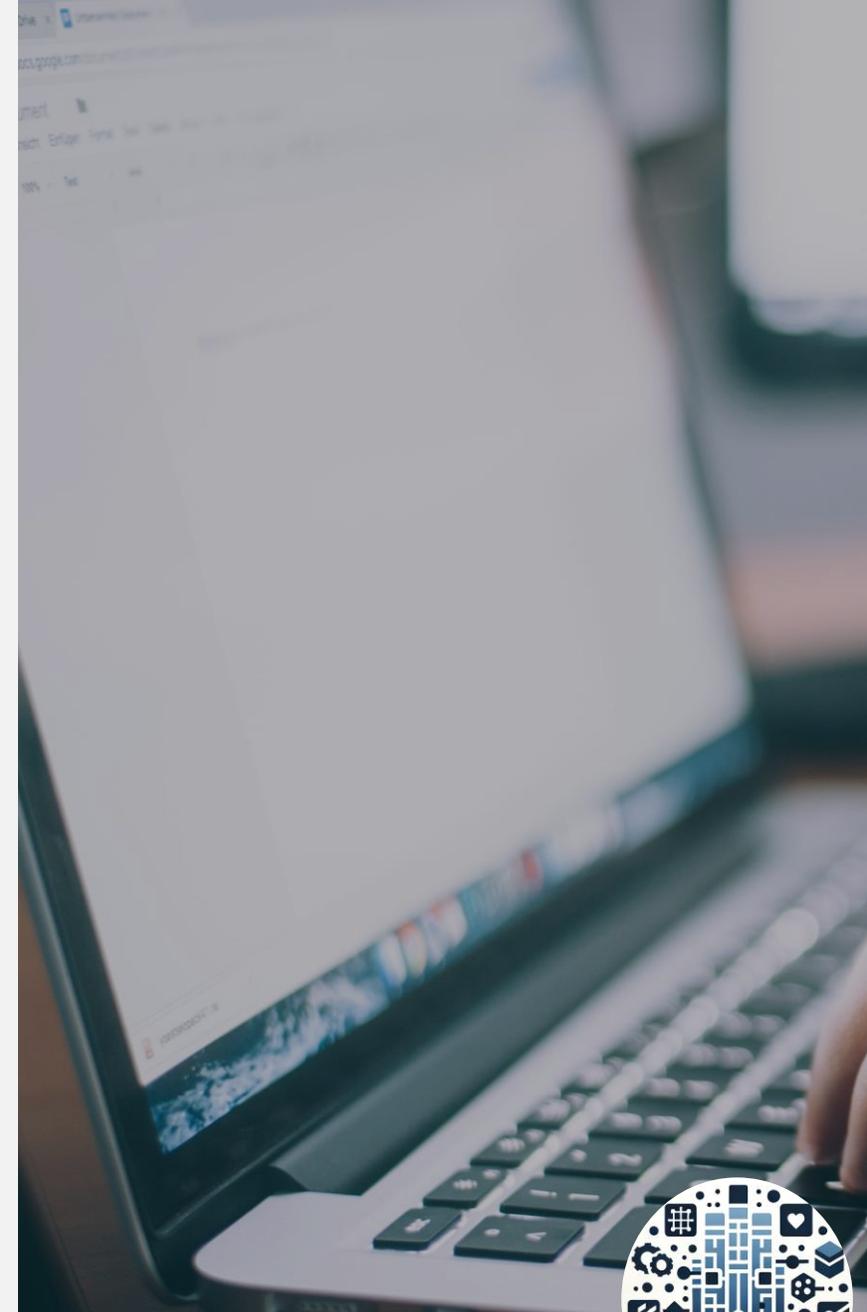


Sopрте de WebRTC



Herramientas para dar soporte a aplicaciones WebRTC

- **WebRTC STUN Check**
- **webrtc-internals/**
- **testRTC**
- **WireShark**



WebRTC STUN Check

ICE servers

STUN or TURN URI:

TURN username:

TURN password:

ICE options

IceTransports value: all relay
Gather IPv6 candidates:
Gather RTCP candidates:
ICE Candidate Pool: 10 0

Time	Component Type	Foundation	Protocol Address	Port	Priority
0.003	1 host	3246152407	udp 192.168.68.101	55886	126 32543 0
0.004	1 host	1472573272	udp 192.168.137.1	55887	126 32287 0
0.005	2 host	3246152407	udp 192.168.68.101	55888	126 32543 0
0.005	2 host	1472573272	udp 192.168.137.1	55889	126 32287 0
0.050	1 srflx	326164312	udp 167.58.230.0	55886	100 32542 255
0.054	2 srflx	326164312	udp 167.58.230.0	55888	100 32542 254
0.123	1 host	3216226383	tcp 192.168.68.101	9	90 32542 255
0.123	1 host	688541120	tcp 192.168.137.1	9	90 32286 255
0.123	2 host	3216226383	tcp 192.168.68.101	9	90 32542 254
0.123	2 host	688541120	tcp 192.168.137.1	9	90 32286 254
39.889					Done
39.892					



webrtc-internals: conjunto de herramientas propias del browser

1

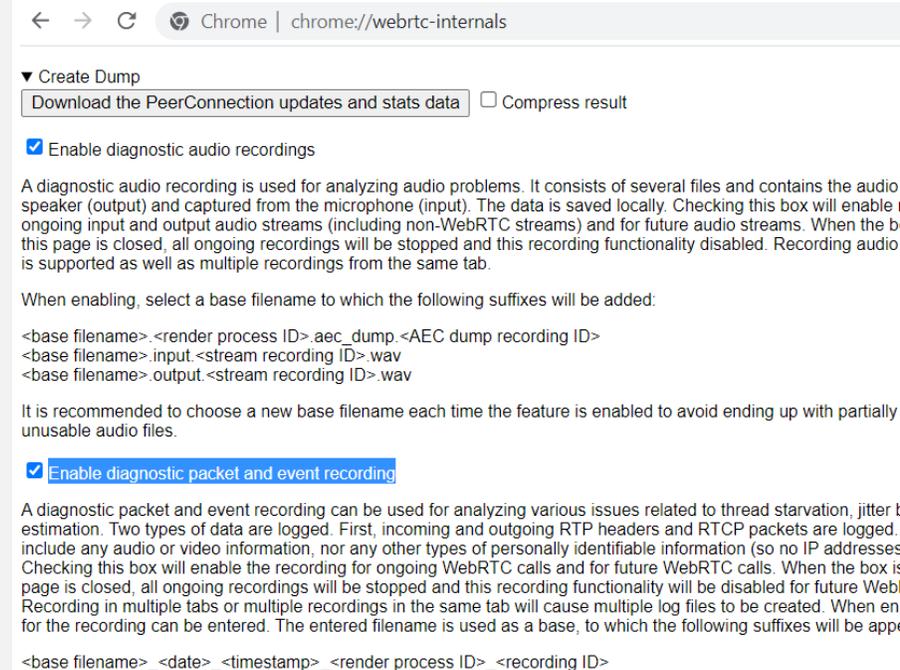
Desplegar los logs

En una nueva pestaña de chrome, escribir **chrome://webrtc-internals/**

2

Habilitar el guardado de logs

Hacer clic en **“Enable diagnostic audio recording”**.
Seleccionar el nombre adecuado para los logs de esa llamada
Hacer clic en **“Enable diagnostic packet and event recording”**.



← → ↻ Chrome | chrome://webrtc-internals

▼ Create Dump

Download the PeerConnection updates and stats data Compress result

Enable diagnostic audio recordings

A diagnostic audio recording is used for analyzing audio problems. It consists of several files and contains the audio speaker (output) and captured from the microphone (input). The data is saved locally. Checking this box will enable ongoing input and output audio streams (including non-WebRTC streams) and for future audio streams. When the browser page is closed, all ongoing recordings will be stopped and this recording functionality disabled. Recording audio is supported as well as multiple recordings from the same tab.

When enabling, select a base filename to which the following suffixes will be added:

```
<base filename>.<render process ID>.aec_dump.<AEC dump recording ID>  
<base filename>.input.<stream recording ID>.wav  
<base filename>.output.<stream recording ID>.wav
```

It is recommended to choose a new base filename each time the feature is enabled to avoid ending up with partially unusable audio files.

Enable diagnostic packet and event recording

A diagnostic packet and event recording can be used for analyzing various issues related to thread starvation, jitter estimation. Two types of data are logged. First, incoming and outgoing RTP headers and RTCP packets are logged. Second, include any audio or video information, nor any other types of personally identifiable information (so no IP addresses). Checking this box will enable the recording for ongoing WebRTC calls and for future WebRTC calls. When the browser page is closed, all ongoing recordings will be stopped and this recording functionality will be disabled for future WebRTC calls. Recording in multiple tabs or multiple recordings in the same tab will cause multiple log files to be created. When enabling for the recording can be entered. The entered filename is used as a base, to which the following suffixes will be appended:

```
<base filename>_<date>_<timestamp>_<render process ID>_<recording ID>
```



3

Ingresar a la aplicación

Entrar en uContact, loguear, etc.

4

Hacer una llamada

Luego de terminar la llamada, ir a la pestaña **Chrome://webrtc-internals** y hacer clic en el cuadro gris de la derecha (https://<host_de_ucontact>: ...).



5

Bajar los logs

Hacer clic en “**create dump**” y hacer clic en “**download de peer connections...**”.

Se descarga un archivo json (con extensión .txt) en la carpeta y nombre seleccionado.

NOTA:

Se debe hacer con la llamada aun ACTIVA

6

Guardar los logs

Copiar todos los logs y archivos de audio generados

.

7

Preparar para próximo log

Para la siguiente llamada, destildar y volver a tildar

“**enable diagnostic audio recording**” para generar logs con otro nombre.
Volver al paso 4

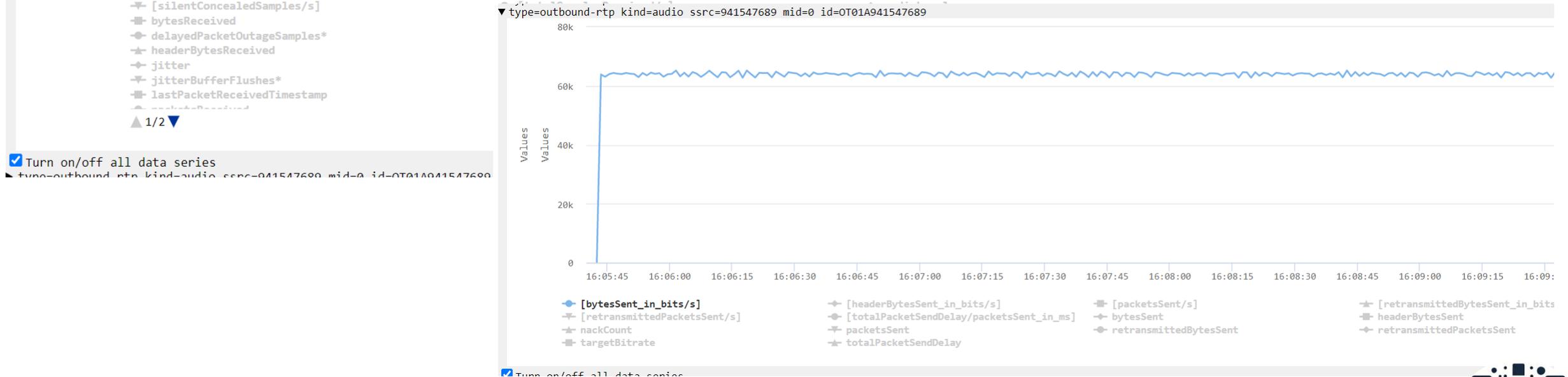
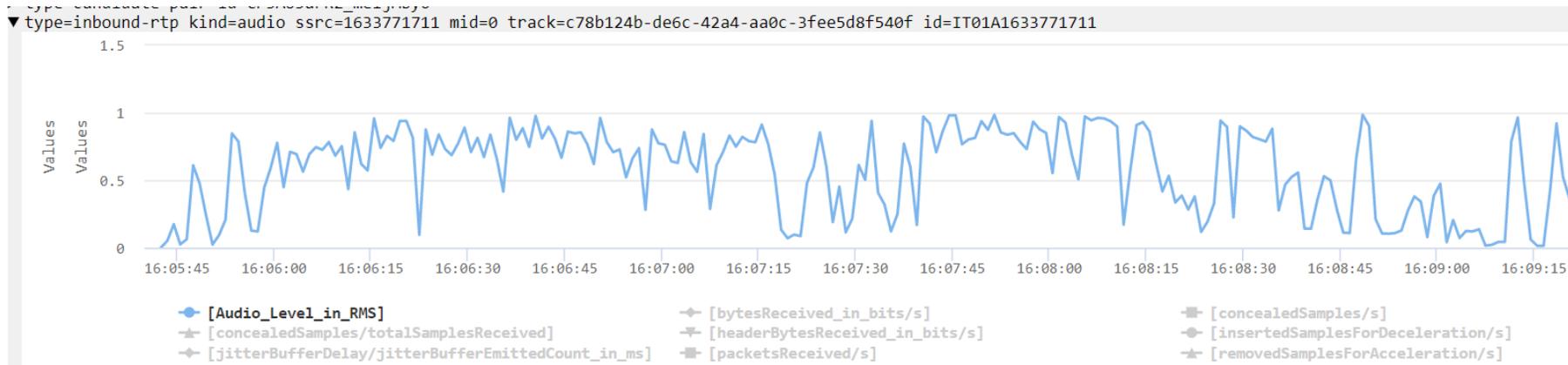


A tener en cuenta

- Este procedimiento deja archivos de AUDIO entrante y saliente separados y varios logs.
- Los audios se pueden escuchar directamente con cualquier reproductor (son WAV).
- Los logs se pueden abrir con cualquier editor de texto, están en format JSON.
- Para visualizarlo en forma amigable:
<https://fippo.github.io/webrtc-dump-importer/>
- Mas Info y detalles:
<https://testrtc.com/webrtc-internals-parameters/>



Información: Audio grabado + gráficas estadísticas





Product ▾

Services ▾

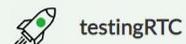
Industries ▾

Resources ▾

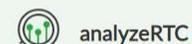
GET STARTED

What do the P

TESTING

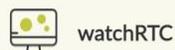


testingRTC
Stress, regression, functional and performance testing



analyzeRTC
Free analysis of webrtc-internal dump files

MONITORING

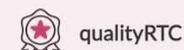


watchRTC
Monitor and analyze WebRTC sessions in real-time

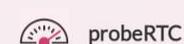


upRTC
Uptime and quality monitoring for your WebRTC infrastructure

SUPPORT



qualityRTC
Reduce handling time of connectivity and quality issues of your end users



probeRTC
Continuous network performance testing for offices



Subscribe Updates

Be the first to know about WebRTC testing and monitoring

Email*

Submit

Hi there,
I'm here to help you at a

Recent Posts

<https://testrtc.com/webrtc-internals-parameters/>



Wireshark: captura y análisis de tráfico de red

Llamada 5.pcapng

Archivo Edición Visualización Ir Captura Analizar Estadísticas Telefonía Wireless Herramientas Ayuda

Aplique un filtro de visualización ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
2225	119.204278	10.0.3.15	10.15.115.14	TCP	1514	https(443) → 65
2226	119.204278	10.0.3.15	10.15.115.14	TCP	1514	https(443) → 65
2227	119.204278	10.0.3.15	10.15.115.14	TCP	1514	https(443) → 65
2228	119.204278	10.0.3.15	10.15.115.14	TCP	1514	https(443) → 65
2229	119.204307	10.15.115.14	10.0.3.15	TCP	54	65270 → https(4
2230	119.205676	10.0.3.15	10.15.115.14	TCP	575	https(443) → 65
2231	119.219472	10.15.115.14	stun.l.google.com	STUN	62	Binding Request
2232	119.220292	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G. 711
2233	119.235076	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G. 711
2234	119.244630	10.15.115.14	10.0.3.15	SRTP	224	PT=ITU-T G. 711
2235	119.251309	10.15.115.14	10.0.3.15	TCP	54	65270 → https(4
2236	119.259368	10.0.3.15	10.15.115.14	SRTP	224	PT=ITU-T G. 711

Wireshark · RTP Stream Analysis · Llamada 5.pcapng

Stream 0 Stream 1 Gráfica

Paquete	Sequence	Delta (ms)	Jitter (ms)	Skew	Ancho
2233	15972	0.000000	0.000000	0.000000	
2236	15973	24.292000	0.268250	-4.292000	
2238	15974	15.869000	0.509672	-0.161000	
2240	15975	19.750000	0.493442	0.089000	
2243	15976	25.431000	0.802040	-5.342000	
2245	15977	24.628000	1.041162	-9.970000	
2246	15978	10.889000	1.545527	-0.859000	

Stream

10.0.3.15:14374 → 10.15.115.14:49222

SSRC 0x4d6ca111

Max Delta 31.897000 ms @ 3365

Max Jitter 1.569226 ms

Mean Jitter 0.167942 ms

Max Skew -11.668000 ms

RTP Packets 7129

Wireshark · Secuencias RTP · Llamada 5.pcapng

Source Address	Source Port	Destination Address	Destination Port	SSRC	Start Time	Duración	Payload	Paquetes	Lost	Min Delta (ms)	Mean Delta (ms)	Max Delta (ms)	Min Jitter	Mean Jitter	Max Jitter
10.0.3.15	14374	10.15.115.14	49222	0x4d6ca111	119.235076	195.38	g711A	7129	0 (0.0%)	8.272000	19.999699	31.897000	0.027804	0.167942	1.569226
10.15.115.14	49222	10.0.3.15	14374	0x972f56ae	118.980623	195.64	g711A	9783	0 (0.0%)	6.680000	19.999971	33.538000	0.048346	0.140451	1.730700

2 streams, 2 selected, 16912 total packets. Right-click for more options.

Limitar filtro de visualización Hora de día

Find Reverse Analizar Prepare Filter Play Streams Copjar Exportar Cerrar Ayuda

2248 119.376693 10.0.3.15 10.15.115.14 SRTP 224 PT=ITU-T G. 711

2249 119.380418 10.15.115.14 10.0.3.15 SRTP 224 PT=ITU-T G. 711

2250 119.395630 10.0.3.15 10.15.115.14 SRTP 224 PT=ITU-T G. 711

2251 119.400450 10.15.115.14 10.0.3.15 SRTP 224 PT=ITU-T G. 711

> Frame 2231: 62 bytes on wire (496 bits), 62 bytes captured (496 bits) on interface \Device\NPF_{07E40D70-66A

> Ethernet II, Src: Dell_18:6d:e4 (d8:9e:f3:18:6d:e4), Dst: JuniperN_af:46:c1 (64:87:88:af:46:c1)

> Internet Protocol Version 4, Src: 10.15.115.14 (10.15.115.14), Dst: stun.l.google.com (64.233.186.127)

> User Datagram Protocol, Src Port: 49222 (49222), Dst Port: 19302 (19302)

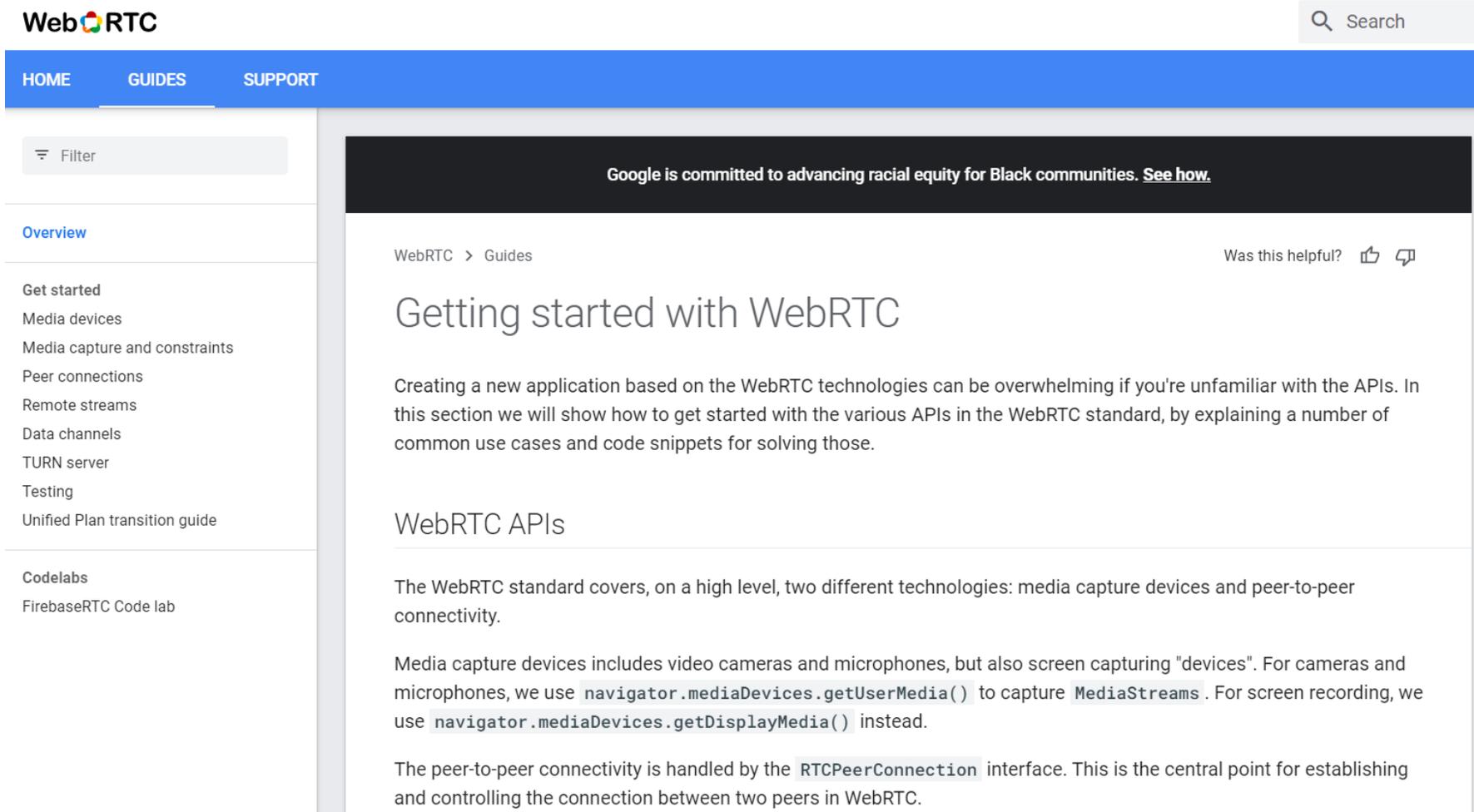
> Session Traversal Utilities for NAT

1 streams, 6: Ir a paquete, Ni: Paquete de problema siguiente

Prepare Filter Play Streams Exportar Cerrar Ayuda



Para desarrolladores: webrtc.org/



The screenshot shows the WebRTC website interface. At the top left is the WebRTC logo. To the right is a search bar with a magnifying glass icon and the text 'Search'. Below the logo is a blue navigation bar with 'HOME', 'GUIDES', and 'SUPPORT' links. On the left side, there is a sidebar with a 'Filter' button and a list of categories: 'Overview', 'Get started', 'Media devices', 'Media capture and constraints', 'Peer connections', 'Remote streams', 'Data channels', 'TURN server', 'Testing', 'Unified Plan transition guide', 'Codelabs', and 'FirebaseRTC Code lab'. The main content area features a black banner with the text 'Google is committed to advancing racial equity for Black communities. See how.' Below this is a breadcrumb trail 'WebRTC > Guides' and a 'Was this helpful?' section with thumbs up/down icons. The main heading is 'Getting started with WebRTC'. The text below explains that creating a new application based on WebRTC technologies can be overwhelming and that this section will show how to get started with various APIs. It then introduces 'WebRTC APIs' and explains that the standard covers media capture devices and peer-to-peer connectivity. It details that media capture devices include video cameras and microphones, and that screen capturing is also included. It provides code snippets for capturing MediaStreams using `navigator.mediaDevices.getUserMedia()` and for screen recording using `navigator.mediaDevices.getDisplayMedia()`. Finally, it states that peer-to-peer connectivity is handled by the `RTCPeerConnection` interface, which is the central point for establishing and controlling the connection between two peers in WebRTC.

<https://webrtc.org/>



Oda a WebRTC.....



WebRTC, tecnología de la comunicación,
que nos permite hablar en cualquier ocasión,
con soporte confiable y sin complicación,
una maravilla de la era digital, sin comparación.

Videoconferencias y streaming en tiempo real,
ya no son cosas de pocos afortunados,
WebRTC lo hace todo muy natural,
para estar siempre conectados

Con su API fácil de usar,
y su protocolo seguro y sin par,
WebRTC nos acerca a los demás,
en un mundo cada vez más global.

Cuando algo falla en la conexión,
y parece que todo va en desilusión,
brindamos el soporte adecuado,
para solucionar el problema en un momento complicado.

Así que gracias WebRTC, de corazón,
por hacer posible lo imposible,
por conectar nuestras almas sin distinción,
y por hacer que la comunicación, sea más accesible.



ChatGPT



¿Preguntas?



¡Muchas gracias!

