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.

WebRTC

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

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.

WebRTC

WebRTC APIs

WebRTC APIs

MediaStream (getUserMedia)

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

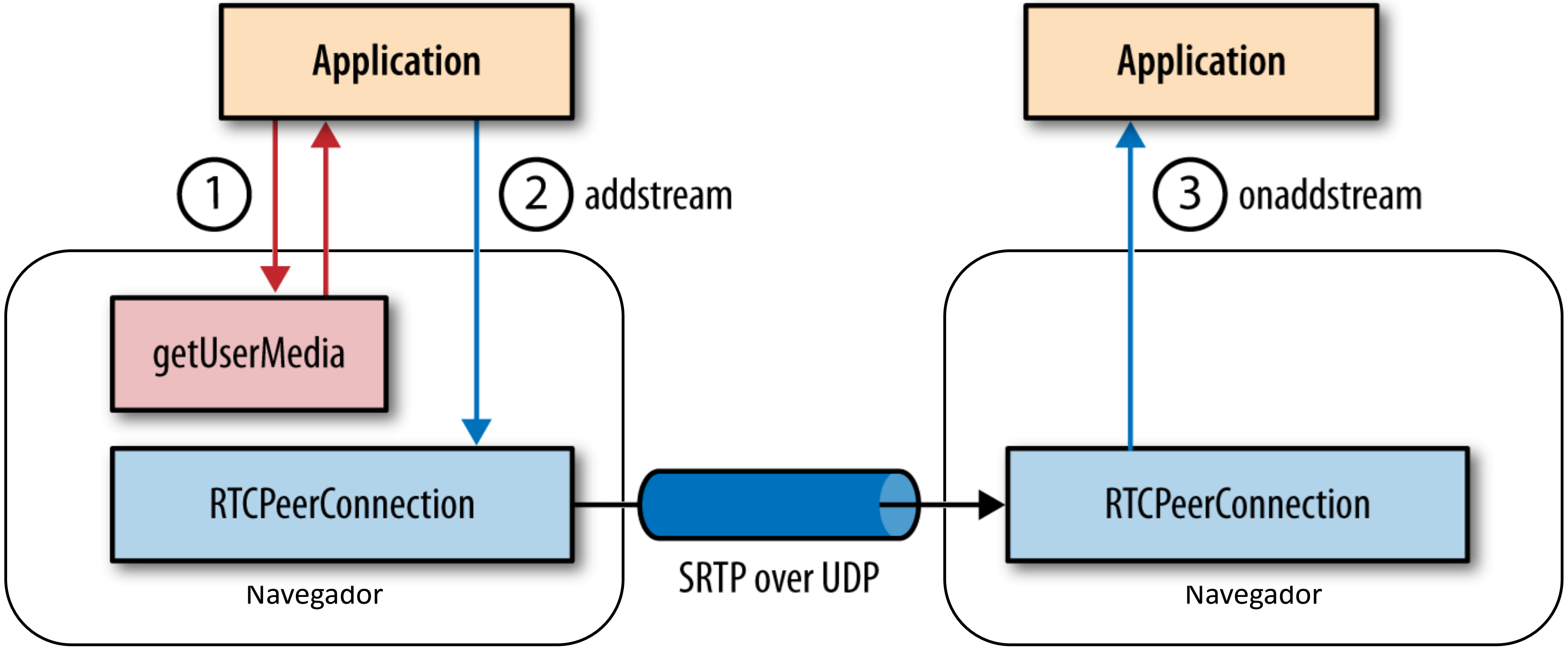
RTCPeerConnection

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

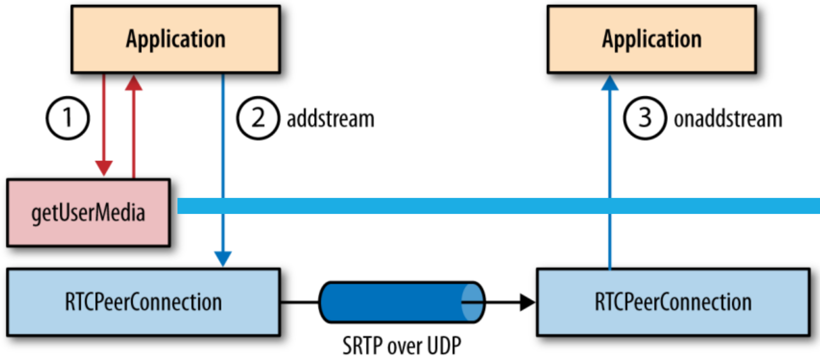
RTCDataChannel

Gestiona comunicaciones de datos.

WebRTC APIs



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



GetUserMedia/GetDisplayMedia Requests

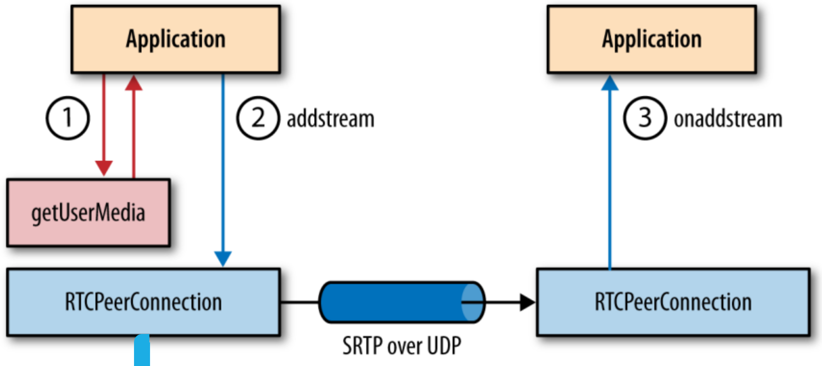
Filter by origin including

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



https://soporteuc.isbel.com.uy/views/agen
[rid: 347, lid: 1, pid: 32260]

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: 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

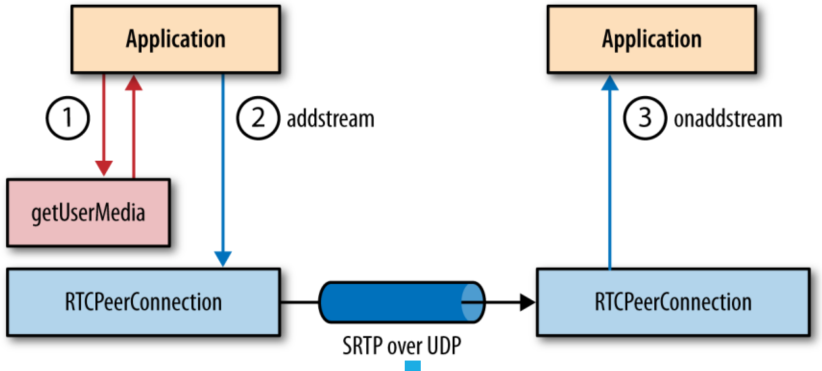
Recepción de capacidades remotas (códecs, etc.) y acuerdo de infraestructura de servidores de medios a usar

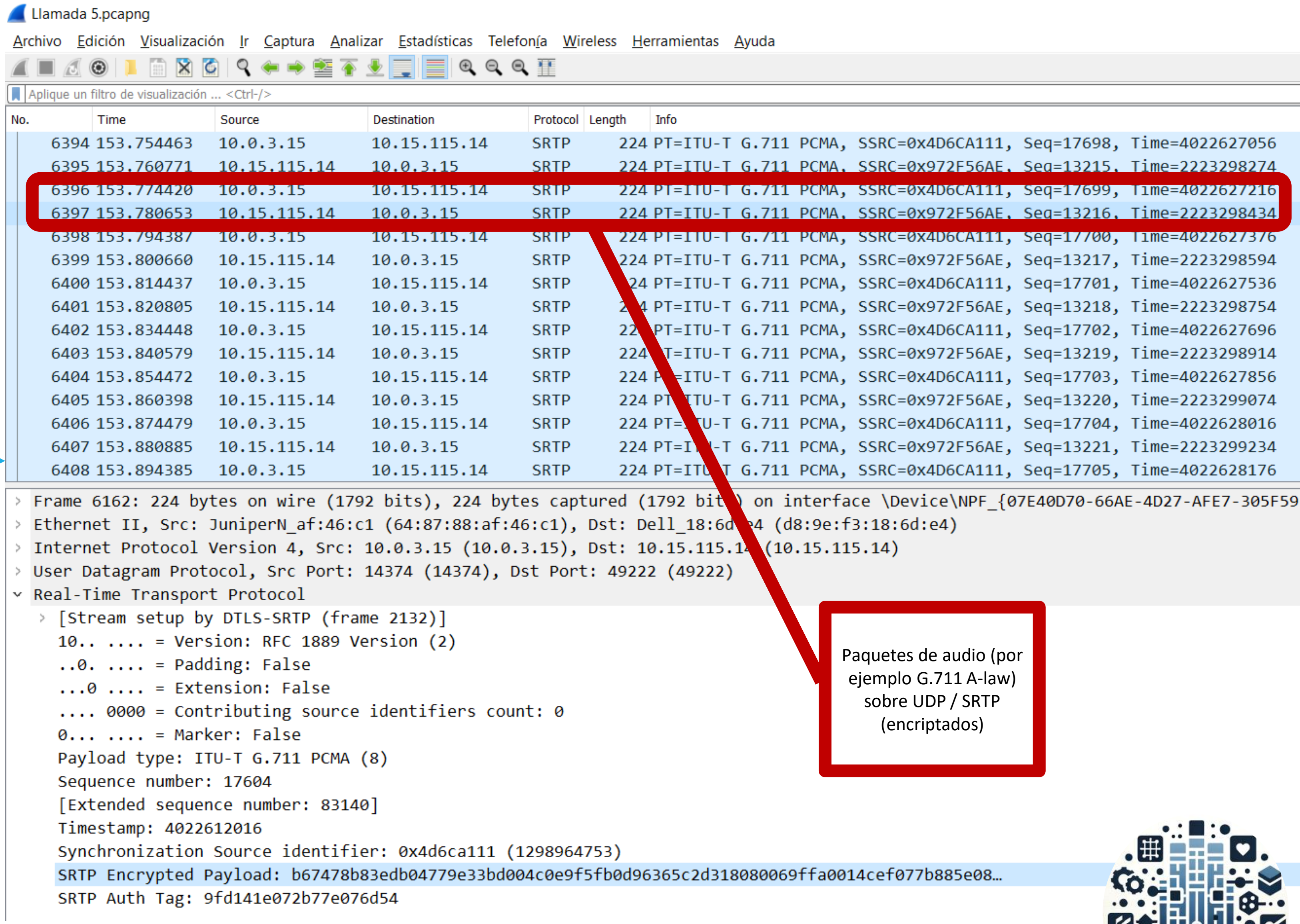
Establecimiento de la comunicación

Fin de la comunicación

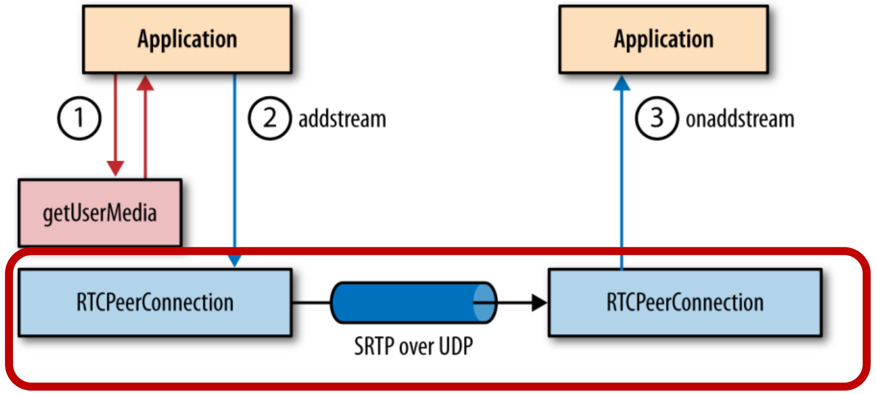
Se establece una conexión con entre origen y destino y se mantiene durante toda la sesión
¡Este proceso es complejo!

(detalles más adelante)

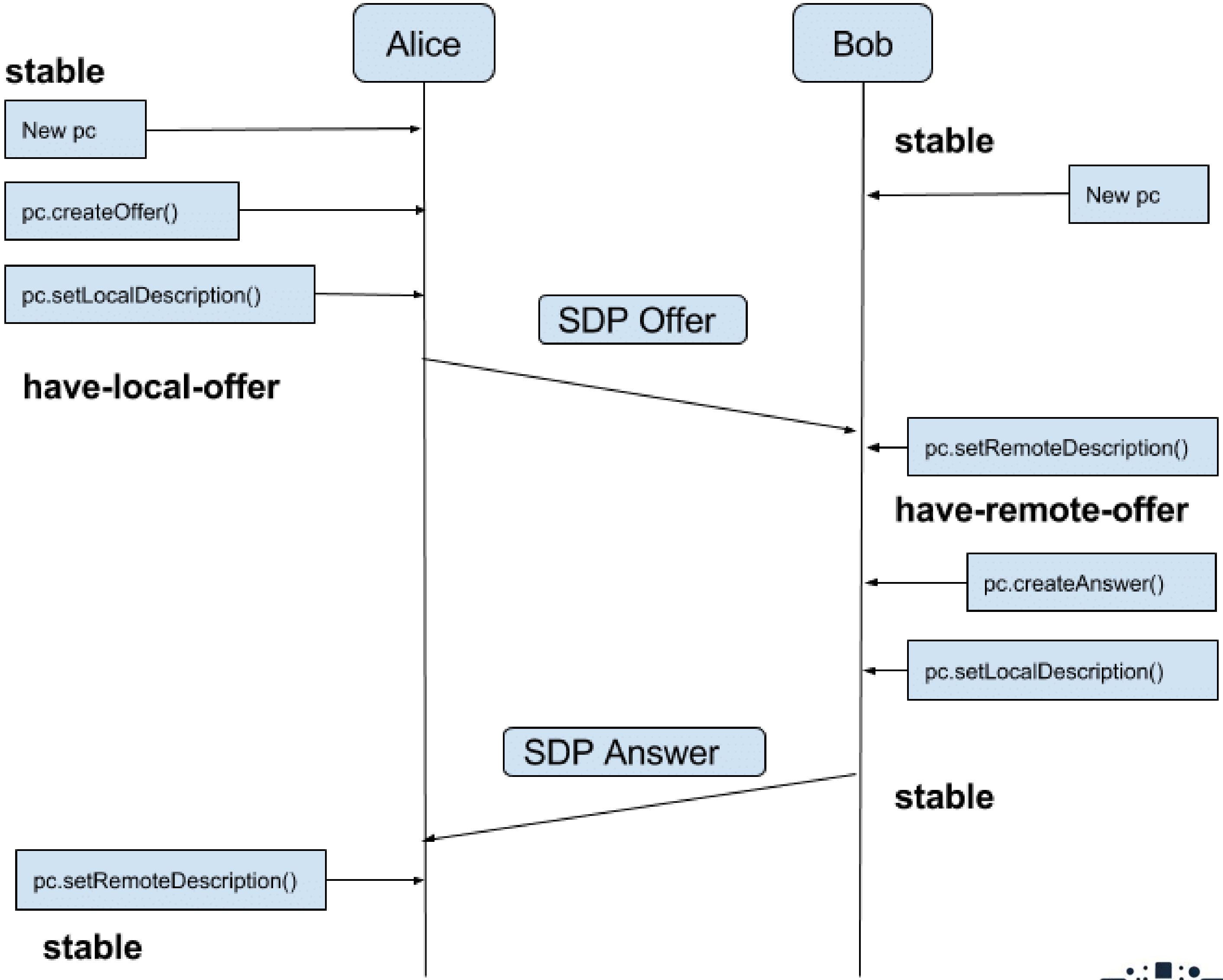




Intercambio de medios (audio, video), en forma segura (encriptada) entre origen y destino



Oferta e intercambio de capacidades entre origen y destino

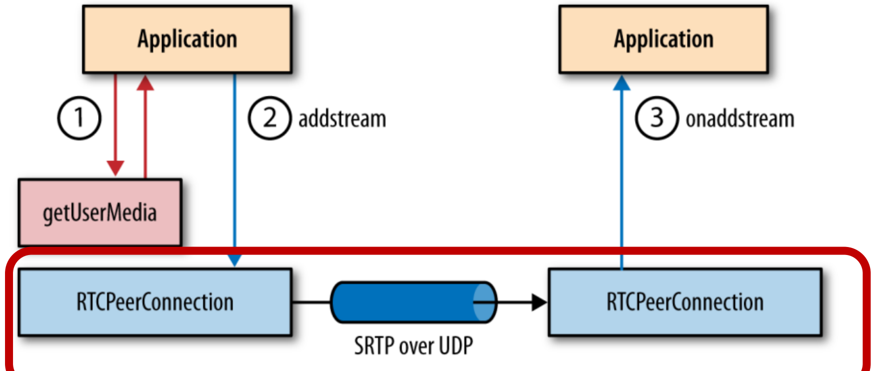


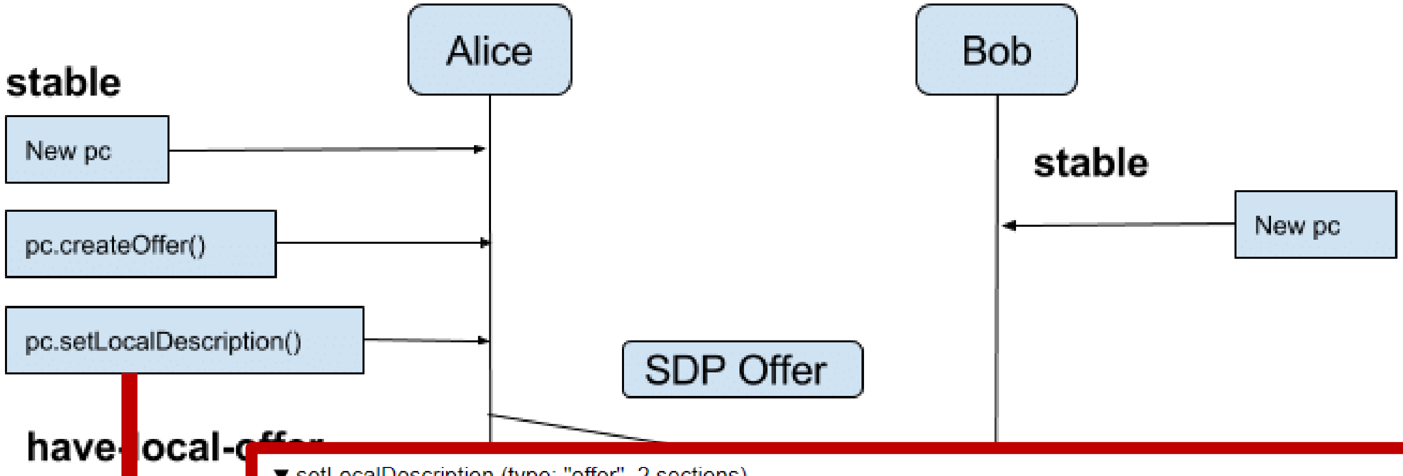
Volvamos al RTCPeerConnection….

¡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)

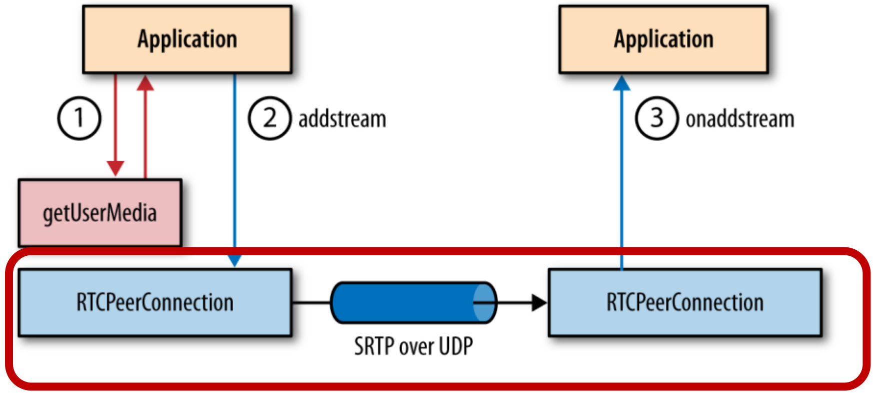
Copy description to clipboard

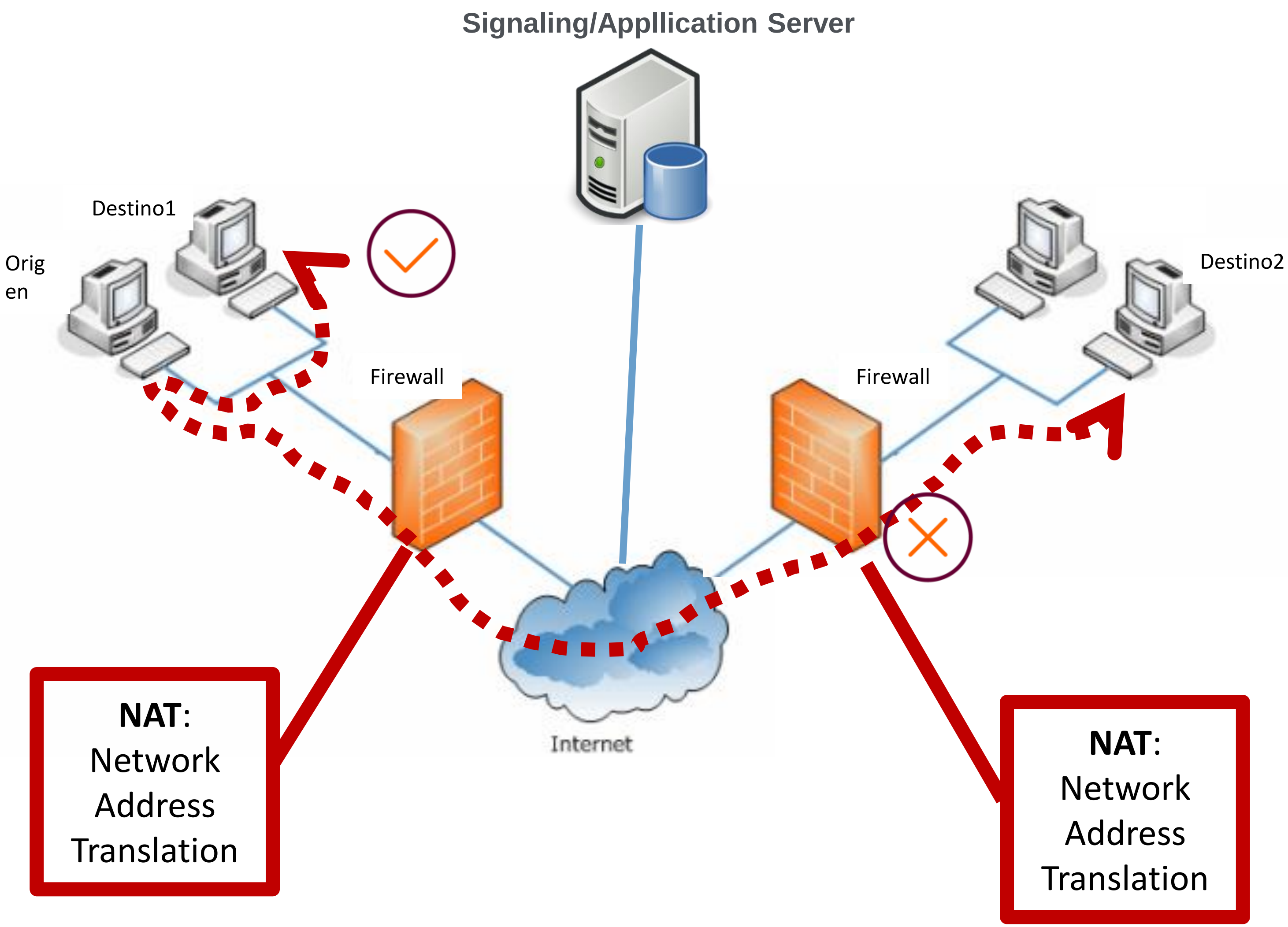
```
▼ v=0 (4 more lines)
o=root 1035137076 1035137076 IN IP4 201.217.144.18
s=Asterisk PBX 13.38.1
c=IN IP4 201.217.144.18
t=0 0
▼ m=audio 12818 RTP/SAVPF 8 0 126 (16 more lines)
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:126 telephone-event/8000
a=fmtp:126 0-16
a=maxptime:150
a=ice-ufrag:6340a96d093368b75187ecc50c4c5c4b
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:xp4J415iV2KcsvMlGwiAEd2N
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
```

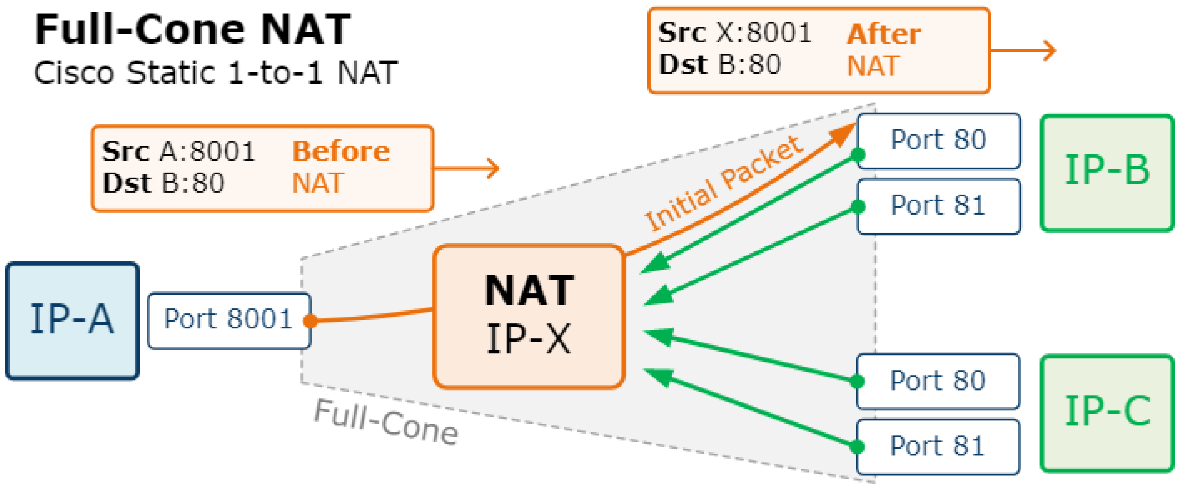


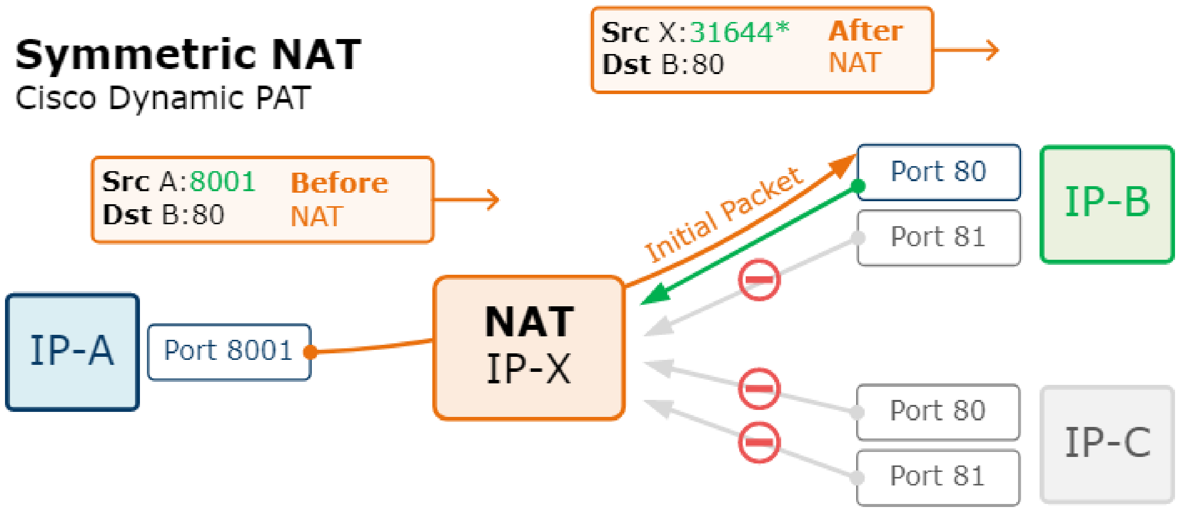


Además…

Origen y destino se deben "ver directamente" para poder intercambiar tráfico multimedia
… pero pueden estar en redes diferentes

Acerca de NAT





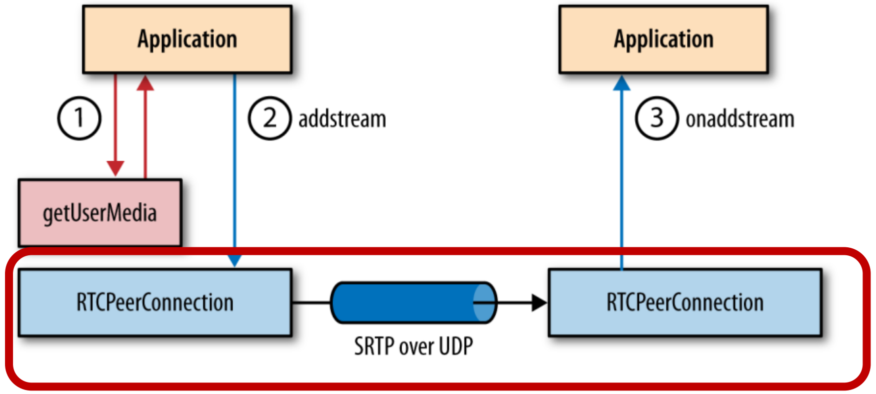
Full-cone NAT

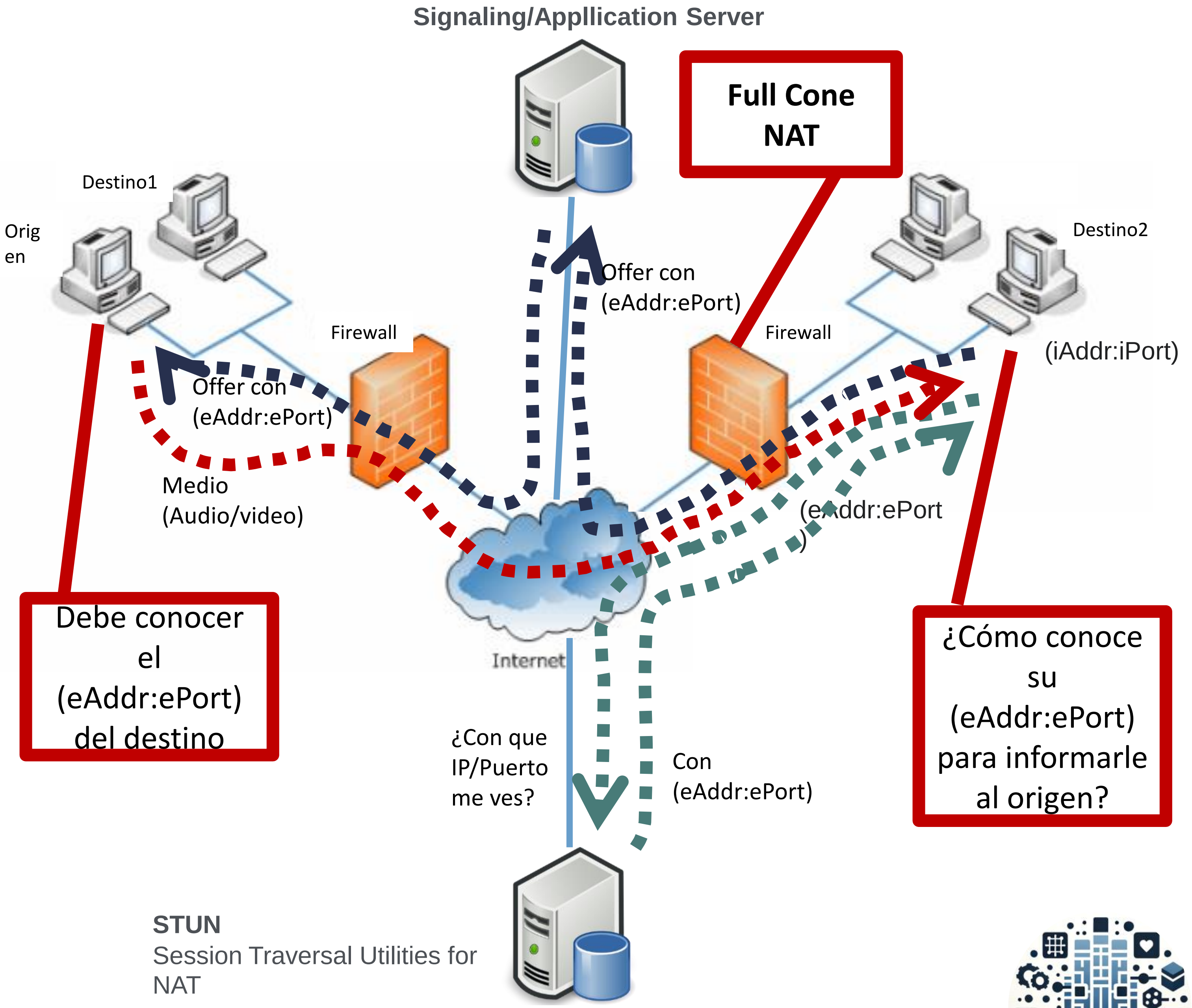
A una dirección interna (iAddr:iPort) se le asigna a una dirección externa (eAddr:ePort).
Todos los paquetes de iAddr:iPort se envían a través de eAddr:ePort.
Cualquier host externo puede enviar paquetes a iAddr:iPort, enviando paquetes a eAddr:ePort.

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

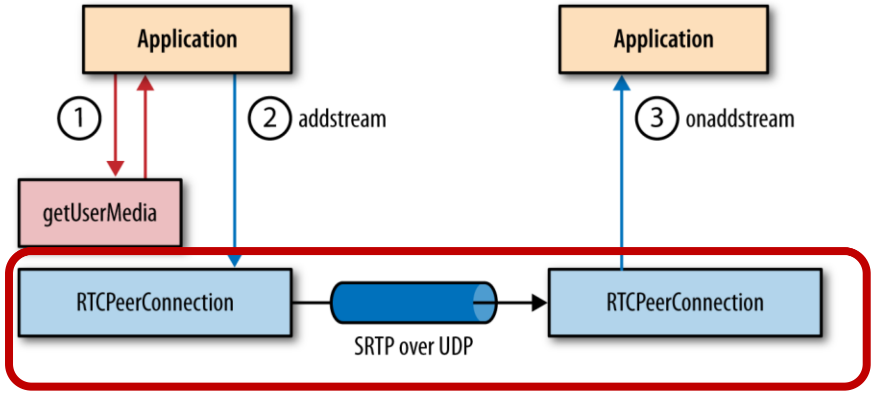
Symetric NAT

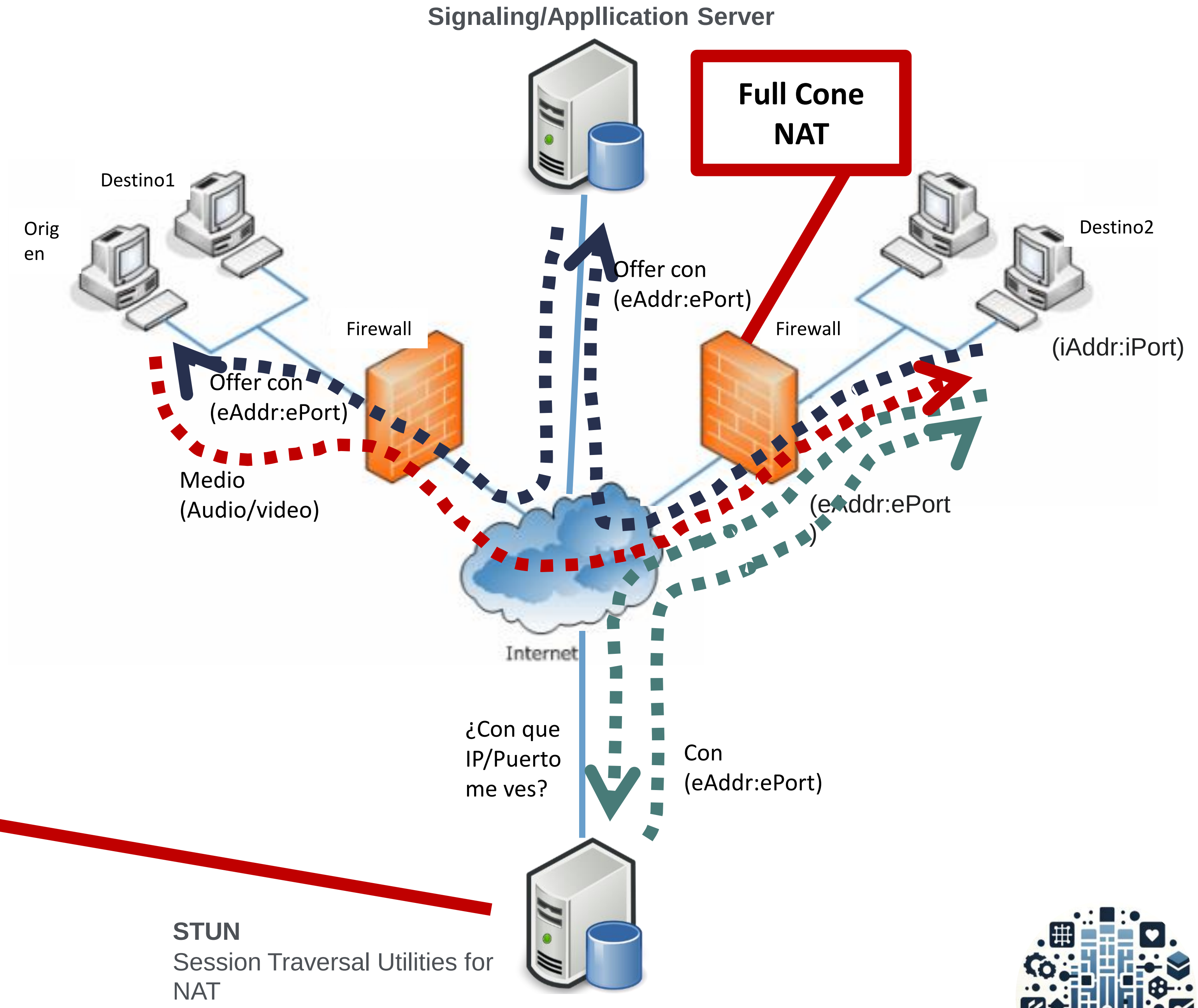
La combinación de una dirección IP interna más una dirección IP y puerto de destino se asigna a una única dirección IP y puerto de origen externo (eAddr:ePort)
Si el mismo host interno envía un paquete (incluso con la misma dirección y puerto de origen) a un destino *diferente*, se utiliza una asignación *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)



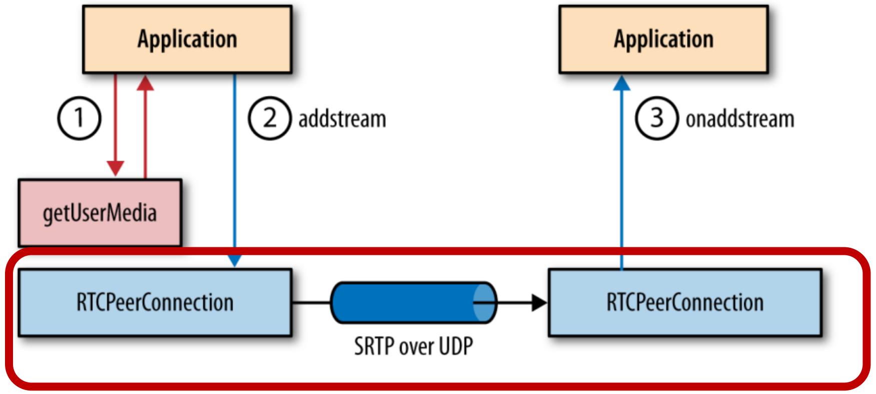


¿Cómo establecer sesiones de medios a través de un NAT?

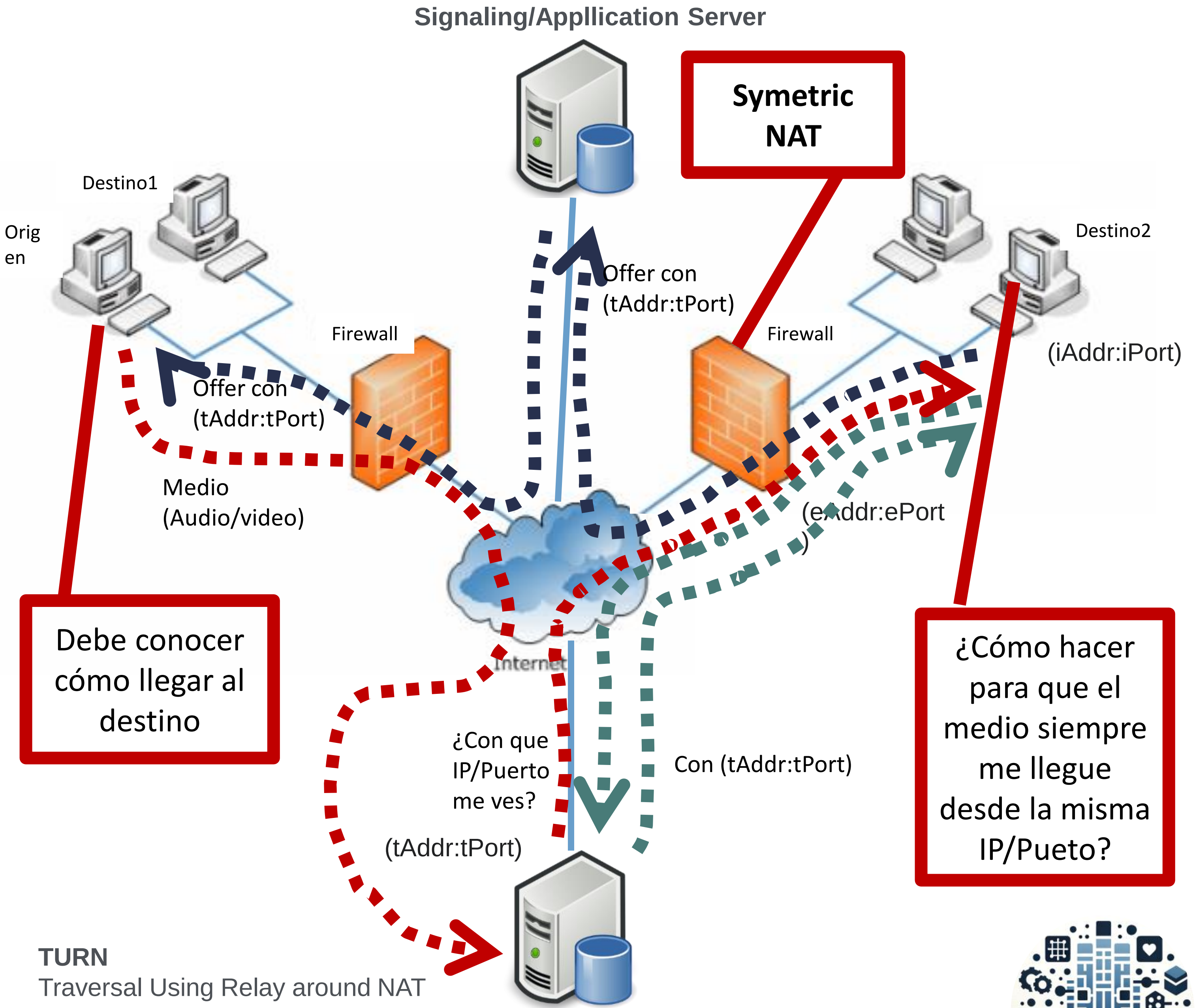




Funciona bien con un "Full Cone NAT"….
¿pero qué pasa si hay un **Symetric NAT**?



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

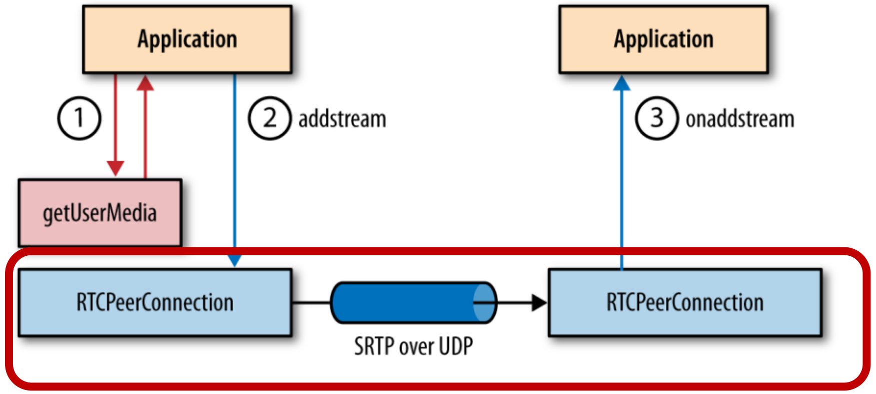


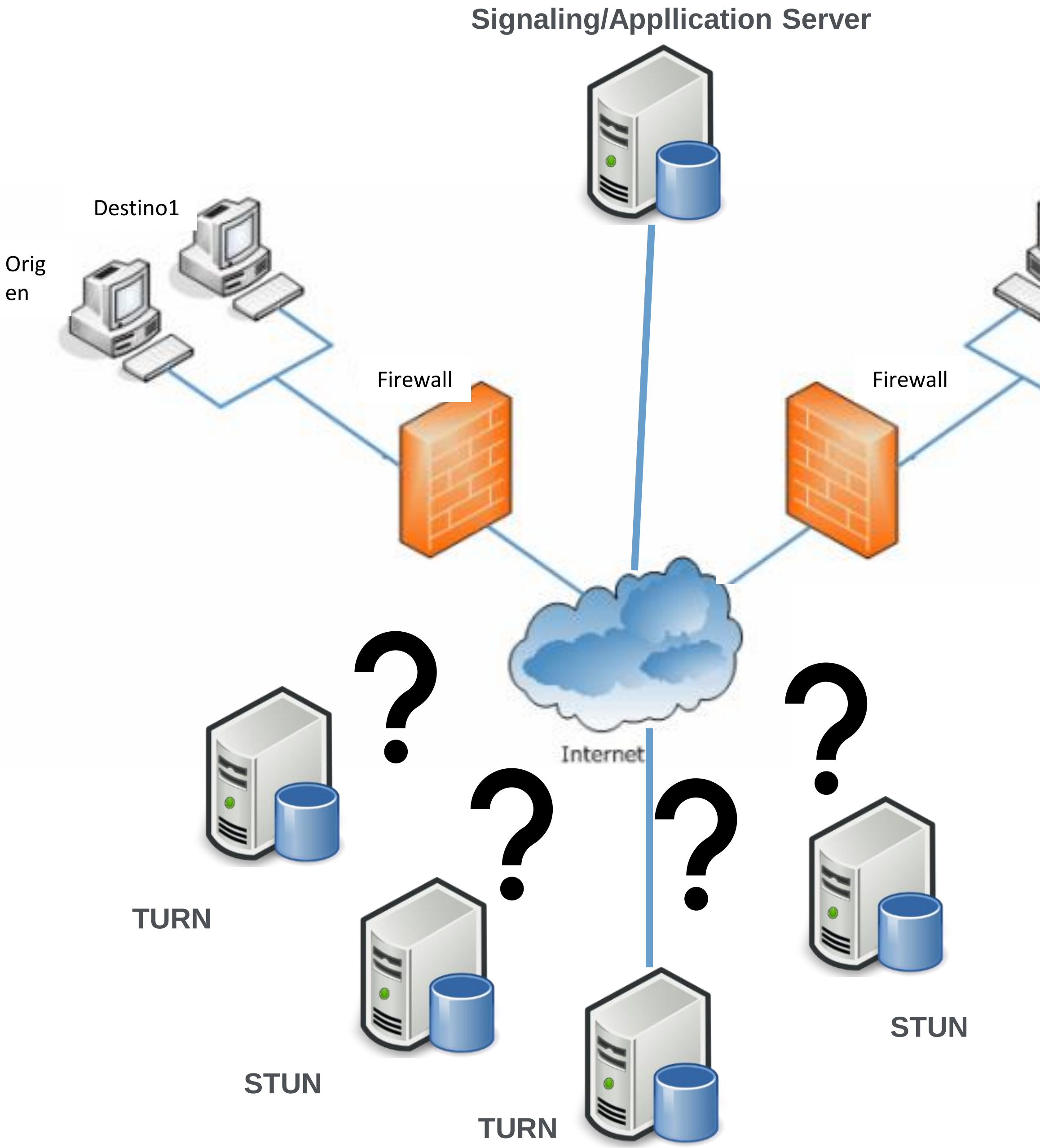
STUN: Session Traversal Utilities for NAT

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)

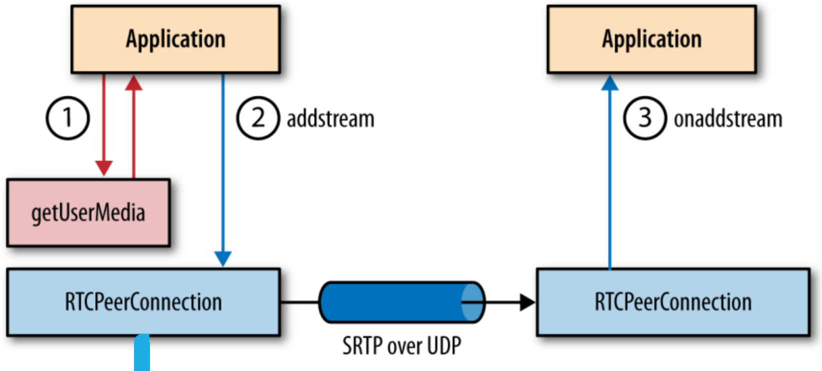
Se ordenan y se elige la “ruta más económica”, con preferencia 1) host → 2) reflexive → 3) relay

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_IfrRn4ry	succeeded	vpn (VPN)		udp	0x7e7d1efffe000000	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						
IIfrRn4ry	remote-candidate	192.168.0.190	16210	host	0x7effffff						
CPTi5gAVKy_IfrRn4ry	in-progress	wifi		udp	0x7e7f1efffe000000	0 / 0	8 / 0				
ITi5gAVKy	local-candidate	192.168.68.101	58811	host	0x7e7f1eff						
IIfrRn4ry	remote-candidate	192.168.0.190	16210	host	0x7effffff						
CPZSQ6AefB_CMMYZCLK	succeeded	wifi		udp	0x647f1effca000000	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	0x64fffffffcfa3c00	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	0x64fffffffcfc3c00	0 / 0	8 / 0	0 / 0			9:12:09
IyXG+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



▼ 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 12818 RTP/SAVPF 8 0 126 (16 more lines)
a=rtpmap:8 PCMA/8000
a=rtpmap:0 PCMU/8000
a=rtpmap:126 telephone-event/8000
a=fmtp:126 0-16
a=maxptime:150
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
```

https://soporteuc.isbel.com.uy/views/agen
[rid: 347, lid: 1, pid: 32260]

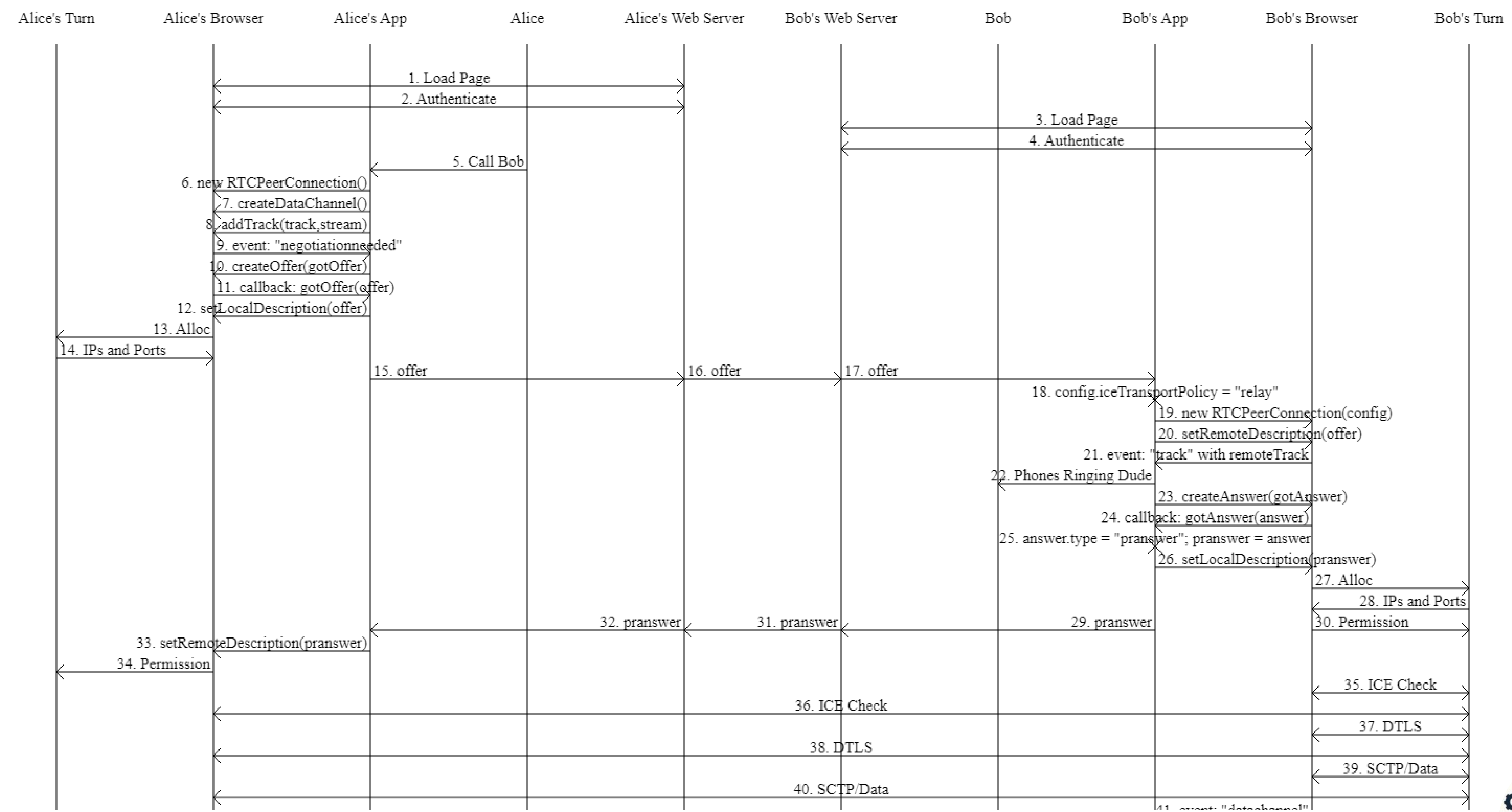
Time	Event
13/4/2023, 8:47:55	▶ transceiverAdded
13/4/2023, 8:47:55	▶ createOffer
13/4/2023, 8:47:55	negotiationneeded
13/4/2023, 8:47:55	▶ createOfferOnSuccess (type: "offer", 2 sections)
13/4/2023, 8:47:55	▶ setLocalDescription (type: "offer", 2 sections)
13/4/2023, 8:47:55	setLocalDescriptionOnSuccess
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: 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

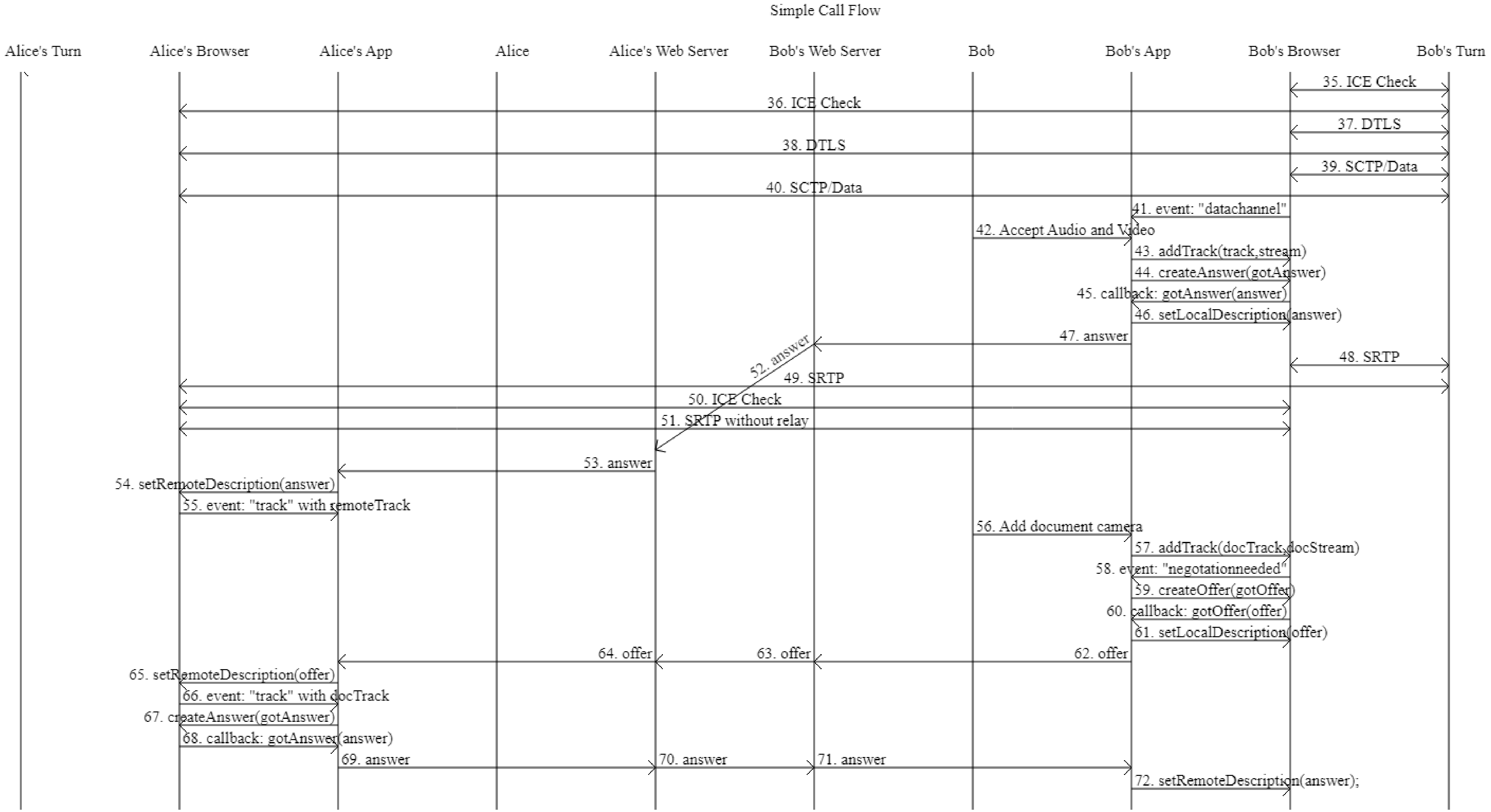
Recepción de capacidades remotas (códecs, etc.) y acuerdo de infraestructura de servidores de medios a usar

Establecimiento de la comunicación

Fin de la comunicación



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



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

Por suerte…

todo resulta ser muy sencillo……

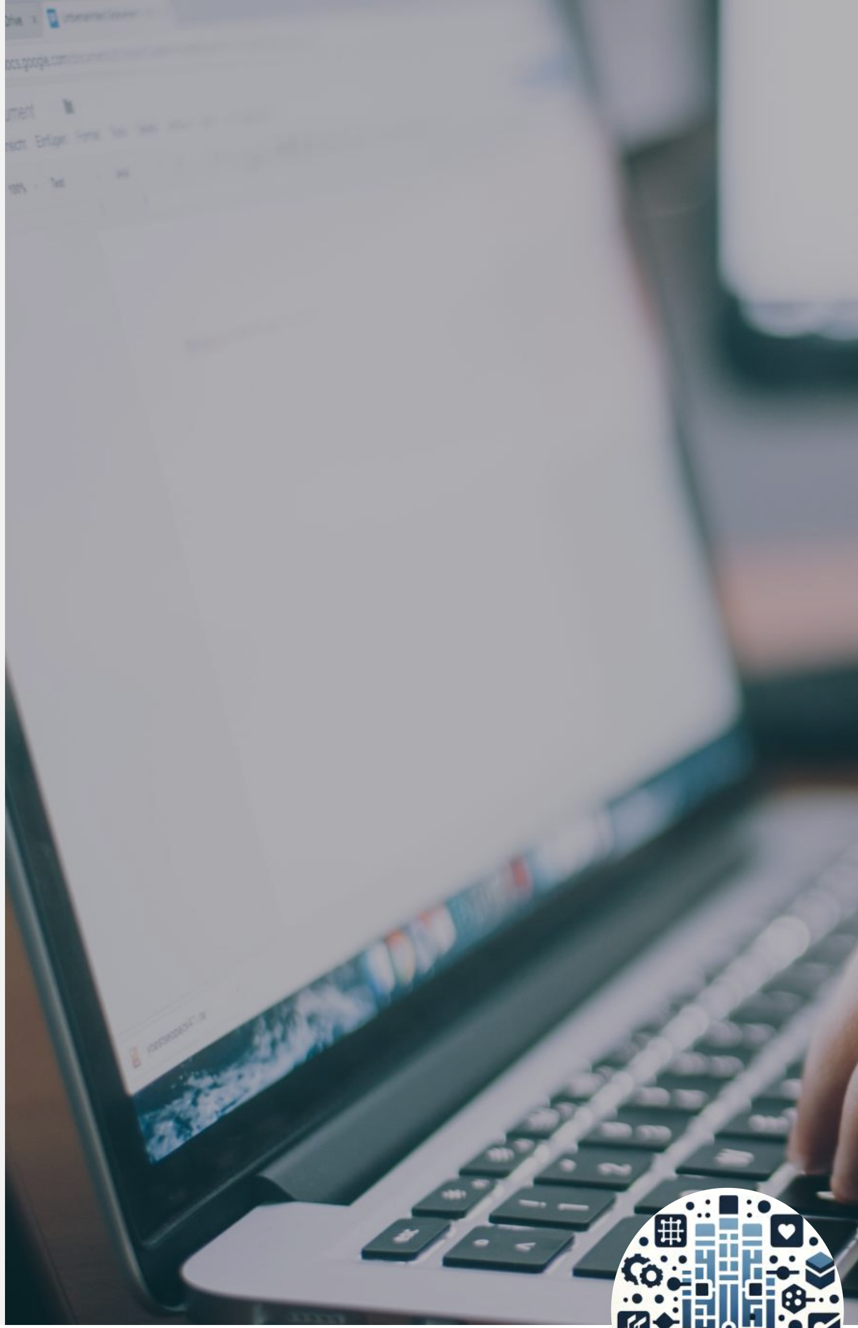
Pero calma…

Hay herramientas para dar soporte…

Soprte de WebRTC

Herramientas para dar soporte a aplicaciones WebRTC

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



WebRTC STUN Check

ICE servers

stun:stun.l.google.com:19302

STUN or TURN URI:

TURN username:

TURN password:

Add Server | Remove Server

ICE options

IceTransports value: ◉ all ○ relay

Gather IPv6 candidates: ☑

Gather RTCP candidates: ☑

ICE Candidate Pool: 10 0 10

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					

Gather candidates

webrtc-internals: conjunto de herramientas propias del browser

1

Desplegar los logs

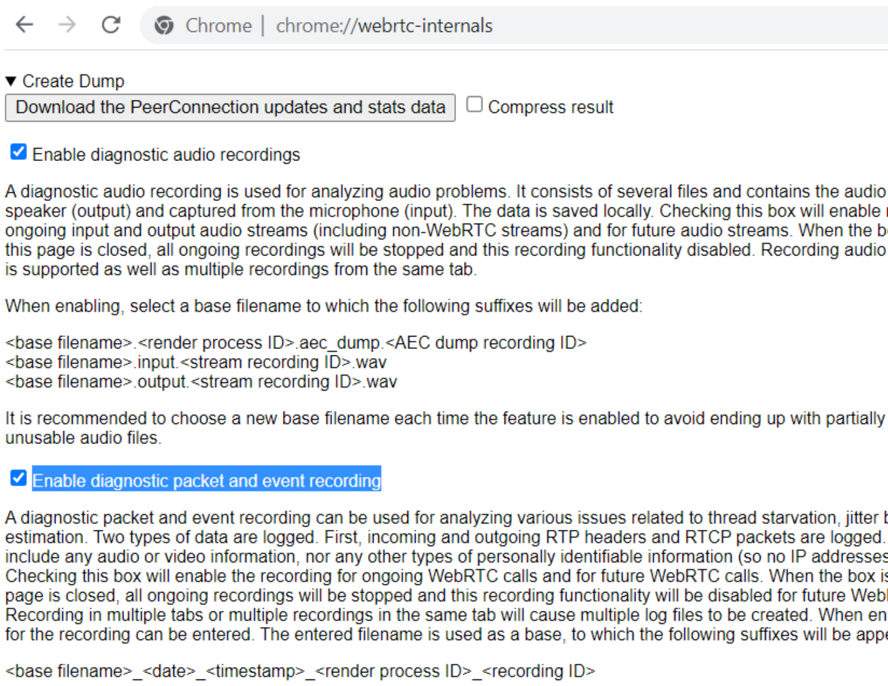
En una nueva pestaña de crome, escribir
chrome://webrtc-internals/

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”.

.



3

Ingresar a la aplicación

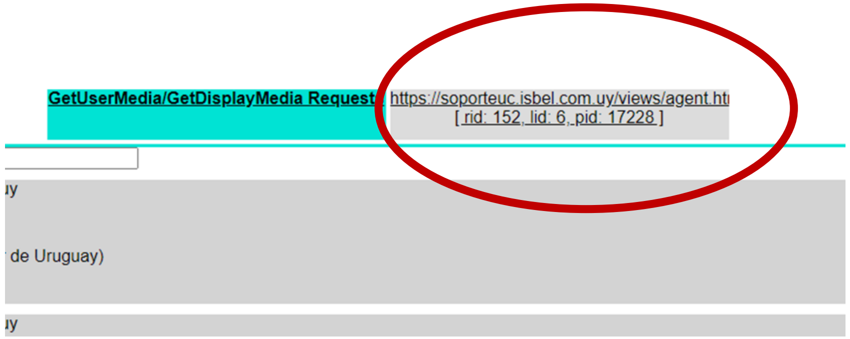
Entrar en uContact, loguear, etc.

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

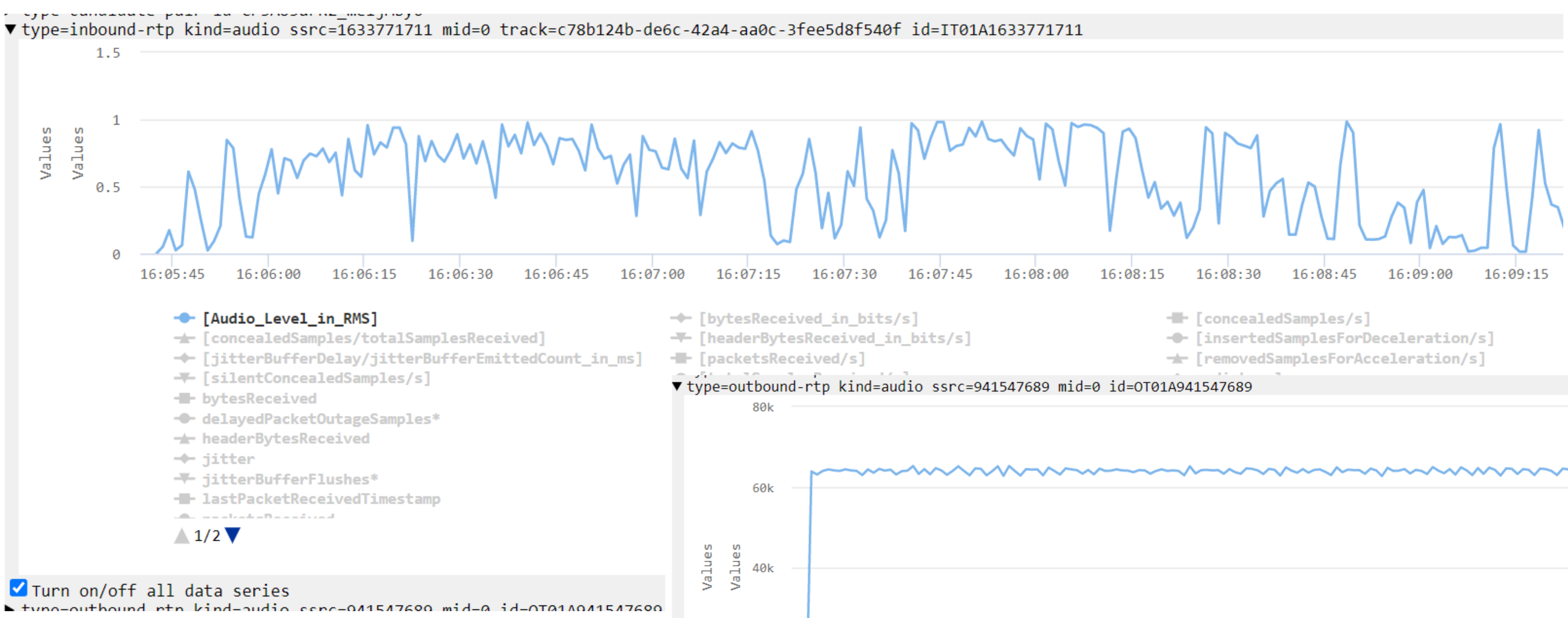
A tener en cuenta

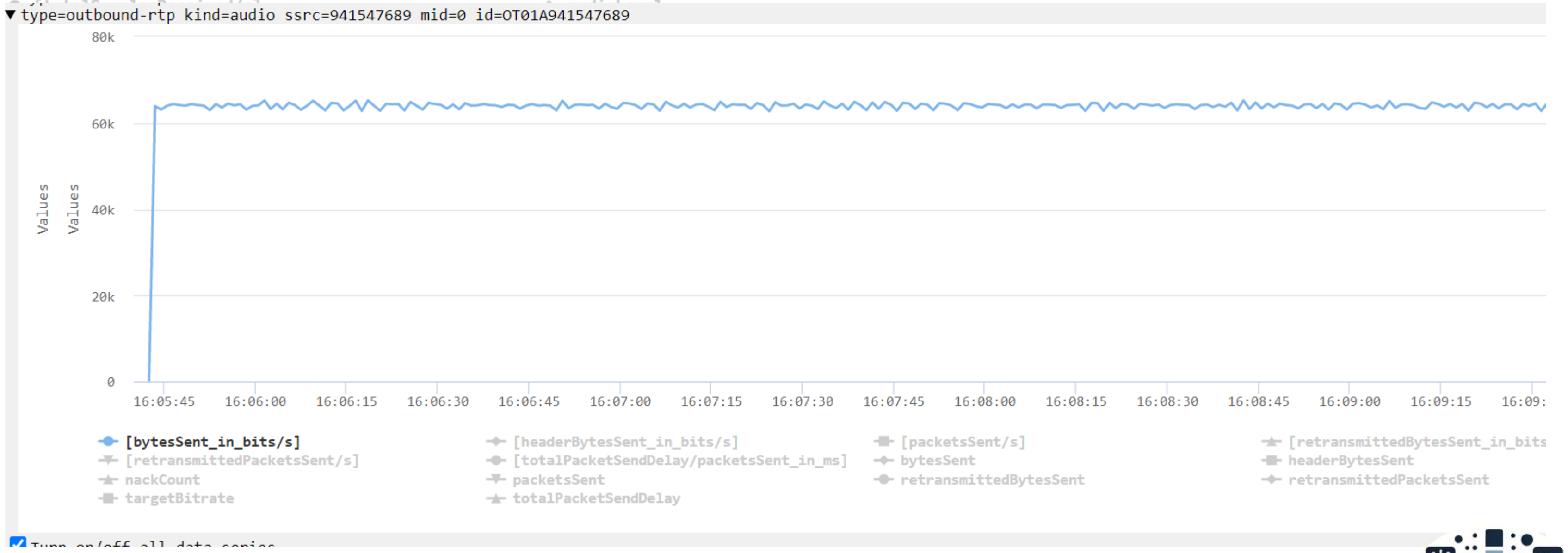
- Este procedimiento deja archivos de AUDIO entrante y saliente separados y varios logs.
- Los audios se pueden escuchar directamente con cualquier reproductor (son WAV).
- Los logs se pueden abrir con cualquier editor de texto, están en format JSON.
- Para visualizarlo en forma amigable:
    https://fippo.github.io/webrtc-dump-importer/

- Mas Info y detalles:
    https://testrtc.com/webrtc-internals-parameters/

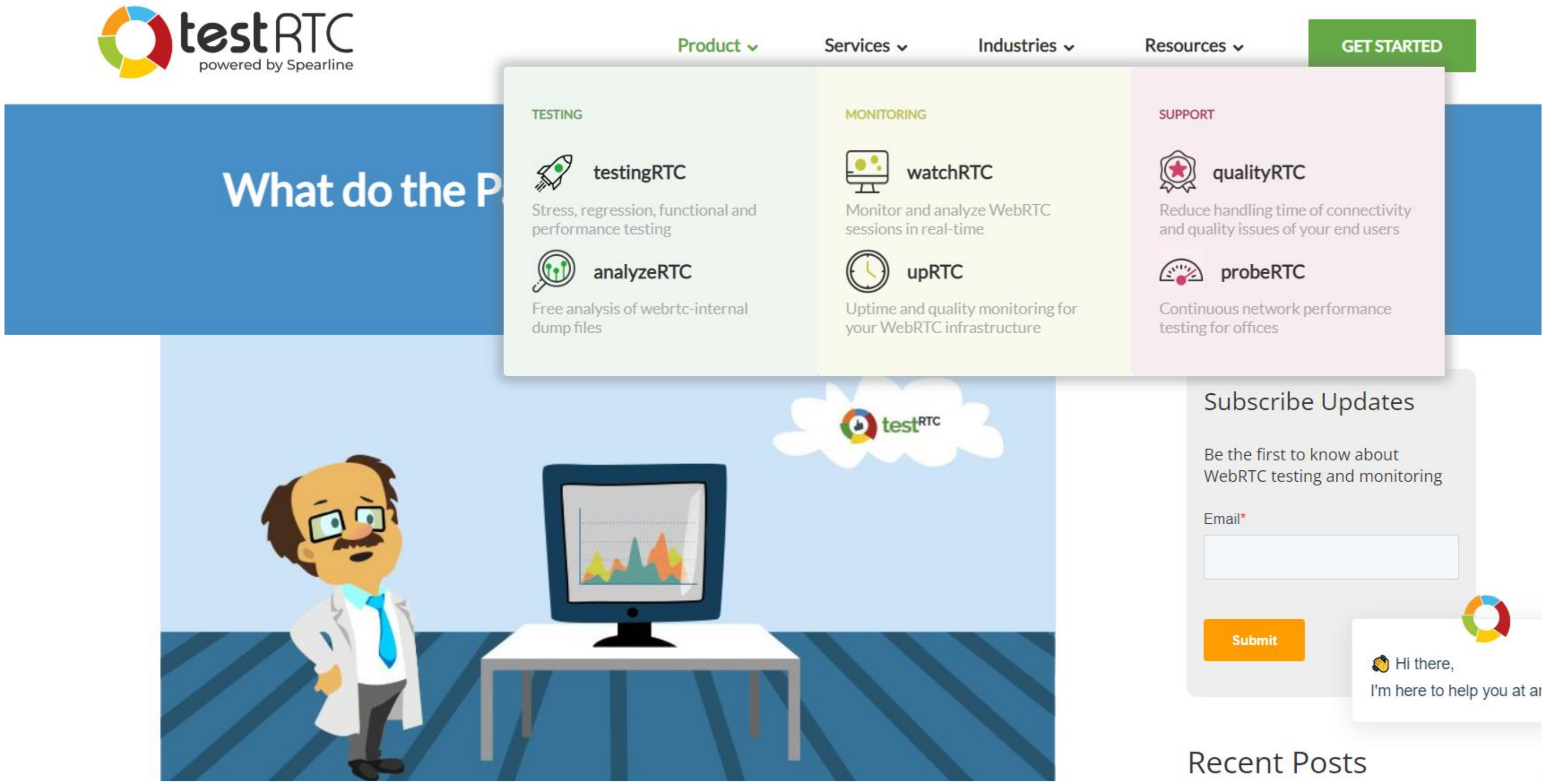
Fuente: https://webrtc.org/

Información: Audio grabado + gráficas estadísticas



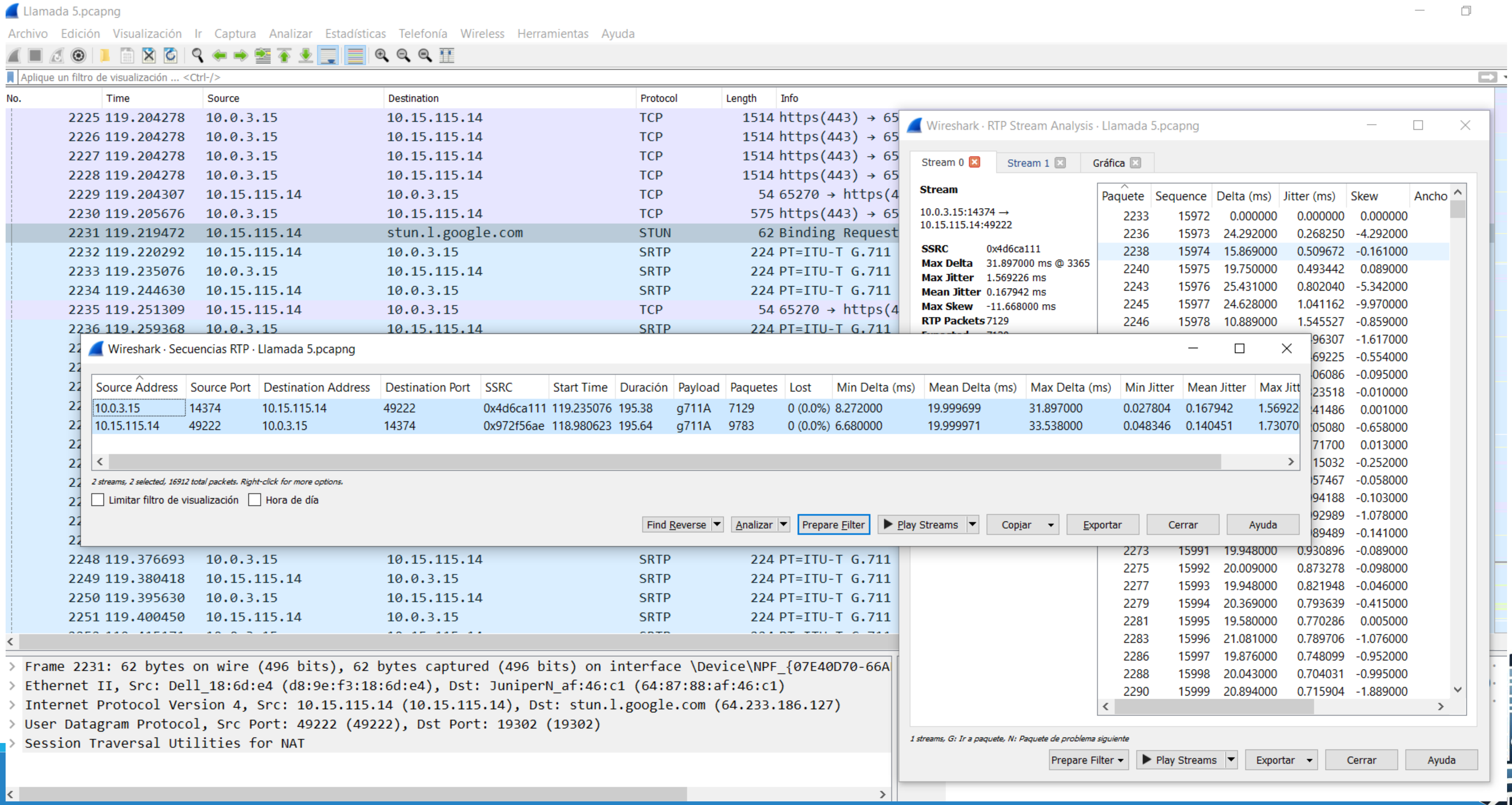


testRTC
powered by Spearline



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

Wireshark: captura y análisis de tráfico de red



Para desarrolladores: webrtc.org/

WebRTC

Search

HOME GUIDES SUPPORT

Filter

Overview

Get started

- Media devices
- Media capture and constraints
- Peer connections
- Remote streams
- Data channels
- TURN server
- Testing
- Unified Plan transition guide

Codelabs

- FirebaseRTC Code lab

Google is committed to advancing racial equity for Black communities. See how.

WebRTC > Guides

Was this helpful?

Getting started with WebRTC

Creating a new application based on the WebRTC technologies can be overwhelming if you're unfamiliar with the APIs. In this section we will show how to get started with the various APIs in the WebRTC standard, by explaining a number of common use cases and code snippets for solving those.

WebRTC APIs

The WebRTC standard covers, on a high level, two different technologies: media capture devices and peer-to-peer connectivity.

Media capture devices includes video cameras and microphones, but also screen capturing "devices". For cameras and microphones, we use `navigator.mediaDevices.getUserMedia()` to capture `MediaStreams`. For screen recording, we use `navigator.mediaDevices.getDisplayMedia()` instead.

The peer-to-peer connectivity is handled by the `RTCPeerConnection` interface. This 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!