WebRTC arquitectura para múltiples conexiones

Autores: Navia Peña Samuel Douglas, Cardenas Morales Luis Fernando

Introducción

WebRTC es una de las tecnología muy utilizada ya a su facilidad y forma de presentar las conexiones entre usuarios en su caso usuarios WebRTC mediante intercambio de datos multimedia entre navegadores, tanto que llegó a ser un referente a al hota de hablar de comunicación a tiempo real, esta está implementada de casi todos los navegadores, por lo cual no se necesita usará complementos adicionales, con esta herramienta se pueden llegar a crear varias arquitecturas MCU(Multipoint Control Unit), SFU(Selective Forwarding Unit) siendo quizás las más características de arquitecturas a la hora de hablar del multicast (one to many), en el presente documento se hablara de las ventajas que tienen estos dos modelos, al igual de algunas implementaciones observadas que usan algunos servicios populares, y algo de MCU mas al lado de lo teórico, para eso se hará el uso de la herramiento de TestRTC que resumiendo nos permite sacar analiticas de los servicios simulando el ingreso de usuarios mediante navegadores.

Marco teórico

WebRTC:llega a agregar funciones de comunicación en tiempo real a la aplicación que funciona además de un estándar abierto.Admite los datos de video, voz y genéricos que se envían entre pares, lo que permite a los desarrolladores crear potentes soluciones de voz y video comunicación. El proyecto WebRTC es de código abierto y es compatible con Apple, Google, Microsoft y Mozilla, entre otros. (WebRTC, 2022)

WebRTC solo admite la comunicación uno a uno, pero se puede usar en situaciones de red más complejas, como cuando varios pares se comunican directamente entre sí o a través de una unidad de control multipunto (MCU), un servidor que puede manejar grandes cantidades de participantes y realizar un reenvío selectivo de transmisiones, así como mezclar o grabar audio y video.(Lacy, 2022)

MCU:Las arquitecturas MCU se han utilizado durante mucho tiempo, desde la antigua era H.323. Básicamente, recibe un flujo de cada participante y, después de decodificarlo, compone un flujo nuevo para enviarlo a todos los participantes. Entonces, los usuarios envían una transmisión y reciben una.SFU:Las arquitecturas SFU son relativamente recientes si se comparan con las MCU. Las SFU no se mezclan, solo transmiten la voz / video recibido de todos los participantes. Esto significa que los usuarios comparten una transmisión con su propia voz / video, pero reciben al menos n-1 transmisiones para ver a todos los participantes.(quobis European Technology, 2021)

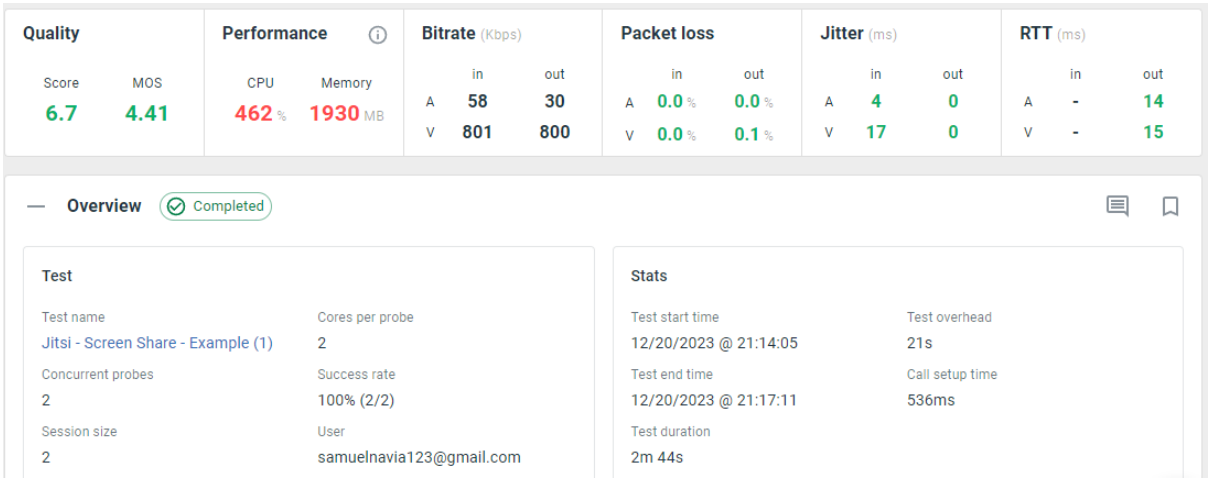
Con los conceptos ya claros podemos ver que la base para realizar WebRTC es la conexión con múltiples es primeramente conectar un peer to peer para luego básicamente lo que permite WebRTC con su api, cabe aclara que WebRTC no solo es para el manejo de video/audio sino también tiene más funciones que no se llegan a tocar, WebRTC como protocolo permite hacer esta conexión peer to peer mediante la conexión de dos agentes WebRTC, luego de esta conexión hay que utilizar una de las dos arquitecturas habladas MCU o SFU para poder conectar, por lo que vamos a realizar tales conexiones y verificar el nivel de estrés que puede realizar a las implementaciones a la hora de utilizar múltiples conexiones.

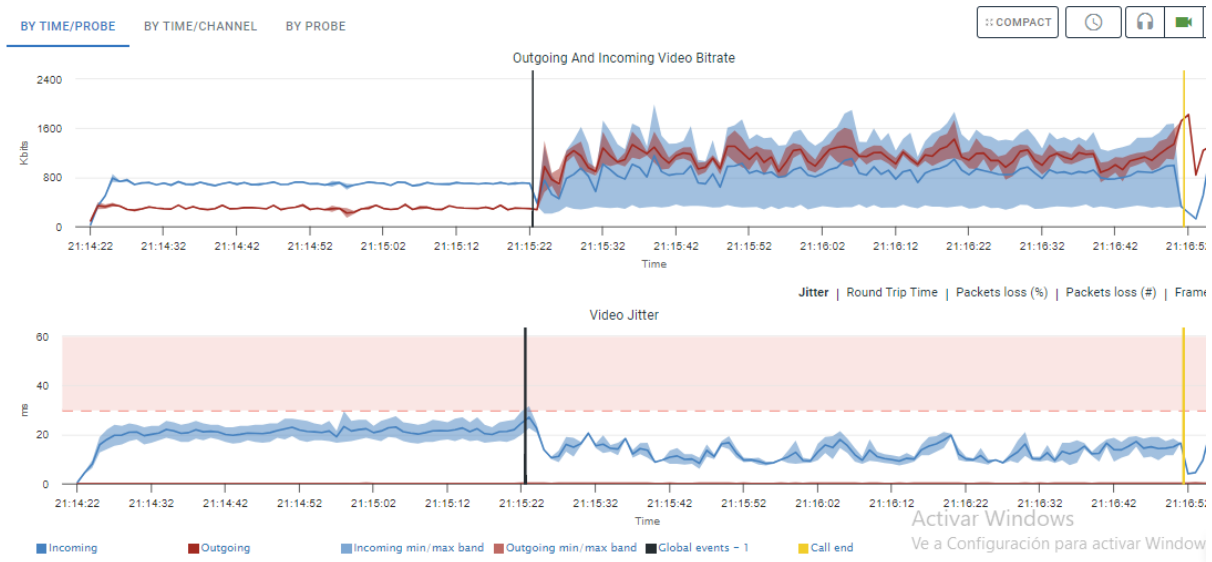
TestRTC: es una parte de cyaras suite de pruebas para probar WebRTC, esta herramienta le ofrecen una ventanilla única para todas la aplicaciones y servicios basados en WebRTC, es una potente herramienta de prueba webRTC que proporciona pruebas rápidas, fáciles y eficaces, depuración, optimización y validación de la aplicación, testRTC simula a los usuarios reales automatizando navegadores web de diferentes lugares que ejecutan diferentes condiciones de red. escalamiento fácil, ya que utiliza código flexible y confiable. (Cyaras, n.d.)

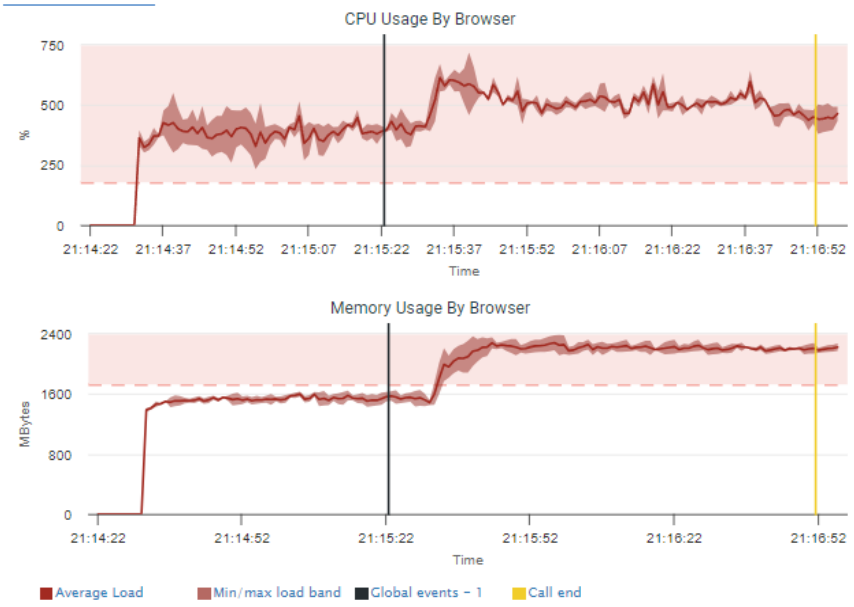
Experimentos

Siguiendo en esta parte consiste en trabajar con los modelos, lastimosamente no se encontró un modelo comercial para hacer las pruebas, el servicio junto los tests.

JITSU




BY TIME/PROBE
BY TIME/CHANNEL
BY PROBE
COMPACT
Outgoing And Incoming Video Bitrate
Kbits
2400
1600
800
0
21:14:22
21:14:32
21:14:42
21:14:52
21:15:02
21:15:12
21:15:22
21:15:32
21:15:42
21:15:52
21:16:02
21:16:12
21:16:22
21:16:32
21:16:42
Time
Jitter | Round Trip Time | Packets loss (%) | Packets loss (#) | Frame
Video Jitter
ms
60
40
20
0
Time
Incoming
Outgoing
Incoming min/max band
Outgoing min/max band
Global events - 1
Call end
Activar Windows
Ve a Configuración para activar Window


CPU Usage By Browser
%
750
500
250
0
21:14:22
21:14:37
21:14:52
21:15:07
21:15:22
21:15:37
21:15:52
21:16:07
21:16:22
21:16:37
21:16:52
Time
Memory Usage By Browser
MBytes
2400
1600
800
0
21:14:22
21:14:52
21:15:22
21:15:52
21:16:22
21:16:52
Time
Average Load
Min/max load band
Global events - 1
Call end

JANUS

Quality		Performance		Bitrate (Kbps)			Packet loss			Jitter (ms)			RTT (ms)		
Score	MOS	CPU	Memory		in	out		in	out		in	out		in	out
7.0	4.41	142 %	1282 MB	A	8	30	A	0.0 %	0.0 %	A	555	0	A	-	96
				V	623	1,568	V	0.0 %	0.0 %	V	147	0	V	-	96

Overview Completed

Test

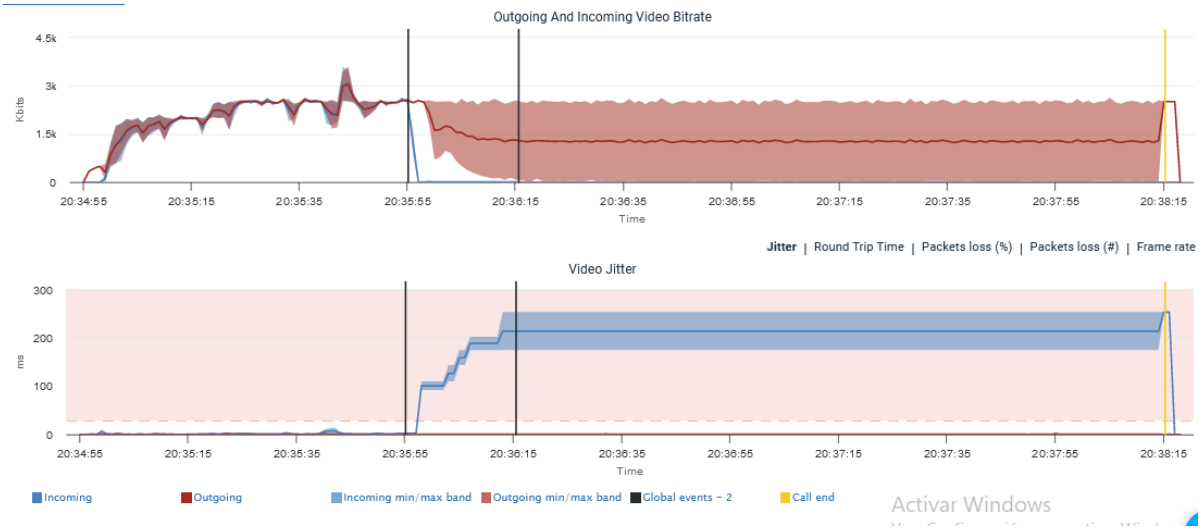
Test name	Cores per probe
Janus example - call drop	2
Concurrent probes	**Success rate**
2	100% (2/2)
Session size	**User**
2	yosoyam4@gmail.com

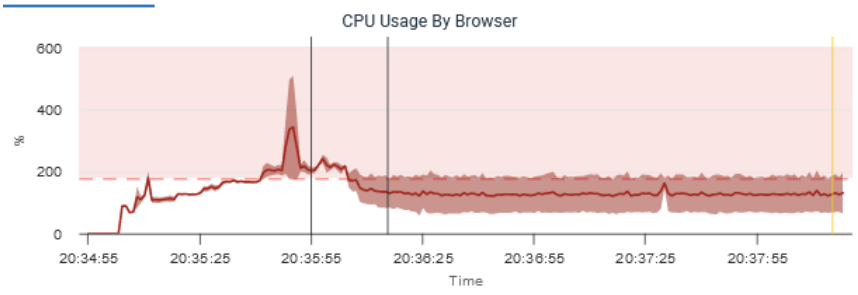
Stats

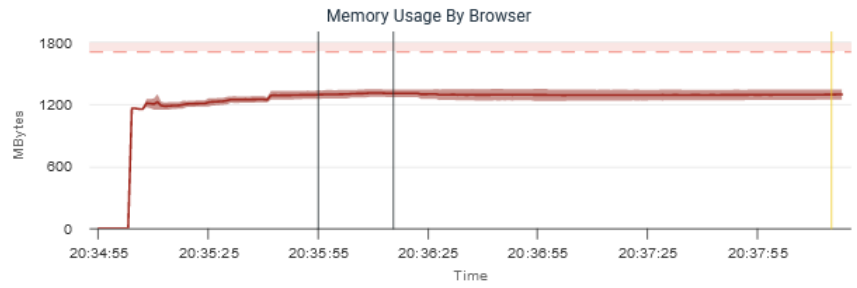
Test start time	Test overhead
12/20/2023 @ 20:34:37	27s
Test end time	**Call setup time**
12/20/2023 @ 20:38:35	353ms
Test duration	
3m 32s	

BY TIME/PROBE | BY TIME/CHANNEL | BY PROBE

:: COMPACT







Resultados y conclusiones

ambos servicios utilizan teoricamente segun su documentación SFU, entonces teóricamente este no procesa ningun recurso no necesario (que algun otro usuario este pidiendo), sino que lo dirige a los usuarios uno por uno , por lo cual no se ve grandes diferencias, pero observamos que el segundo llega a ser más eficiente que el primero, esto posiblemente por el estado de la red mia, la de la empresa, y quizás la forma que se implementaron, aunque janus tenga un jitter mayor este llega a manejarlo mejor por lo quer la implementación que se usa es mejor, también hay que notar que jitsi utiliza entrada y salida simétrico siendo 801 y 800 los valores en KBPS casi siendo una velocidad 1 a 1, en contraste con lo que saca janus de 639 y 703 los valores KBPS siendo una velocidad de 9 a 10 esto indica que ambos llegan a ser muy buenos escalando en cuanto a los servicios que utilizan, hablando del consumo de los navegadores observamos que JITSU llega a utilizar muchos mas recursos que lo que es JANUS llegando a los 1920MB con uso del CPU muy alto estos datos solamente aclaran lo ya sabido que JITSU esta manejando de manera supoptima los recursos pidiendo mas de lo que se necesita, esto puede llegar a dar ventajas ya que la calidad de los recursos sera mayor, pero muy sujetos a la velocidad de conexion que se tiene con el servicio, a diferencia de JANUS que intenta mantener 1282 MB lo que aclara que este esta utilizando algun delimitador para que todos los servicios lleguen a tener una misma calidad en los servicios , aunque no se logró encontrar un servicio de MCU vamos usar el conocimiento dado por la página (WebRTC Multiparty Video, 2016) para tomar como base en el articulo UCM significa que cada navegador envía el supuesto video/audio y al centro lo uno para poder pasar una sola secuencia a todos los demas usuarios de la red, uno por cada usuario que usa que da, la logica de estos servicios para llegar a superar la mayor cantidad de usuarios es limitar la input y output de cada usuario lo que puede llegar a ser más eficiente que otros servicios SFU.

Con los datos dados podemos observar que la mejor implementación para el envío de pantalla, video, audio es SFU quizás pidiendo que el servidor trabaje más pero mejorando la escalabilidad de servicios para la conexión de varios servicios, pero si de latencia se quiere hablar la mejor solución es MCU ya que este no llega a tener tanto procesamiento como SFU y mejora la latencia pero hay que tomar en cuenta desde el inicio de la construcción del servicio la cantidad Máxima de usuarios los cuales queremos que ingresen al servicio, y así prever, limitar el (out/in)put de cada navegador, lo que llega a ser muy difícil el escalamiento del servicio en un futuro.

Referencias

Cyaras. (n.d.). *Cyara testingRTC — Cyara*. Cyara. Retrieved December 20, 2023, from https://cyara.com/products/testingrtc/

Lacy, S. (2022, October 2). . . - YouTube. Retrieved December 18, 2023, from https://web.dev/articles/webrtc-basics?hl=es-419#network_topologies

Quobis European Technology. (2021, May 3). *SFU vs MCU: ¿cuál es la mejor forma de gestionar una multiconferencia?* Quobis. Retrieved December 18, 2023, from https://quobis.com/es/2021/05/03/sfu-vs-mcu-cual-es-la-mejor-forma-de-gestionar-una-multiconferencia/

WebRTC. (2022, October 2). .*Comunicación en tiempo real para la Web*. WebRTC introducción. Retrieved December 18, 2023, from https://webrtc.org/?hl=es-419

WebRTC Multiparty Video. (2016, March 7). *How Different WebRTC Multiparty Video Conferencing Technologies Look Like on the Wire*. testRTC. Retrieved December 20, 2023, from https://testrtc.com/different-multiparty-video-conferencing/