Redes Multimedia – Prácticas 2020
Práctica 3: VoIP

Turno y pareja: Martes (2461) y 06

Integrantes:

Pablo Díez del Pozo

Alejandro Alcalá Álvarez

Contenido

1 Introducción

El objetivo de la práctica es que el alumno entienda la arquitectura de una infraestructura de comunicaciones VoIP y sea capaz de construir los elementos básicos de una infraestructura VoIP. Para ello se plantean los siguientes sub-objetivos:

1. Conocer los elementos básicos de una arquitectura VoIP.

2. Entender las diferencias entre protocolos de señalización (SIP) y de transporte de medios (RTP), así como entender y manejar los elementos básicos de SDP.

3. Desplegar una infraestructura VoIP usando como base el servidor Yate (yate.null.ro) y softphones (Yate y Zoiper).

4. Entender el funcionamiento interno del protocolo SIP y como se establecen y liberan comunicaciones multimedia entre agentes de usuario remotos.

5. Entender conceptos básicos de rutado de llamadas en una centralita de VoIP (PBX). 6. Ser capaz de configurar un softphone SIP y los elementos de una infraestructura VoIP.

2 Realización de la práctica

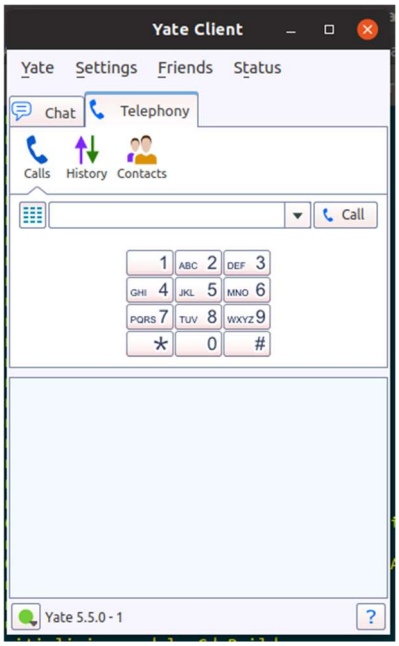
1. Cada pareja procederá a configurar el servidor Yate en su ordenador de trabajo. Para ejecutar este software deberá seguir la documentación disponible en el *site* de Yate (yate.ro). A la hora de ejecutar el servidor y el cliente Yate se recomienda hacer uso de los scripts **run y run-qt4** que proporciona Yate y ejecutar por terminal para poder ver la salida de error. Esta salida nos mostrará información con descripciones de errores de configuración. Para realizar la configuración debemos modificar una serie de ficheros que encontraremos en la carpeta conf.d de Yate.

Debido a que el material que nos proporcionan los profesores de Yate ya esta compilado y listo para ejecutar de run y run-qt4. El comando run sirve para ejecutar el servidor de Yate, donde tenemos varias opciones de iniciar, se puede iniciar el servidor de la siguiente manera:
- Ejecutarlo en modo debug con el siguiente flag : ./run -vvvvvv
- Ejecutarlo en modo demonio con el siguente flag: ./run -d

Para ver todas las acciones que realiza el servidor lo vamos a ejecutar en modo debug par ver todas las acciones que realizamos y los mensajes que recibimos del cliente.

```
aalcala@aalcala-SATELLITE-PRO-C50-A-1HQ:~/Escritorio/yate-5.5.0-1.comptlado/yate$ ./run -vvvvvv
Yate (7075) is starting Sun Mar 22 10:13:03 2020
2020-03-22_10:13:03.800944 <NOTE> Failed to open config file './conf.d/yate.conf', using defaults (2: No such file or directory)
2020-03-22_10:13:03.838186 <ALL> Plugin::Plugin("gvoice",false) [0x7fb23bd54460]
Loaded module GVoice
2020-03-22_10:13:03.854131 <ALL> Plugin::Plugin("mux",true) [0x7fb23bb4e600]
Loaded module MUX
2020-03-22_10:13:03.866393 <ALL> Plugin::Plugin("msgsniff",false) [0x7fb23b944220]
Loaded module MsgSniffer
2020-03-22_10:13:03.872796 <ALL> Plugin::Plugin("enumroute",false) [0x7fb23b73e200]
2020-03-22_10:13:03.885813 <ALL> Plugin::Plugin("ilbcwebrtc",false) [0x7fb23b537580]
Loaded module iLBC - based on WebRTC iLBC library version 1.1.1
2020-03-22_10:13:03.897230 <ALL> Plugin::Plugin("tonedetect",false) [0x7fb23b3202e0]
Loaded module ToneDetector
2020-03-22_10:13:03.911156 <ALL> Plugin::Plugin("filetransfer",false) [0x7fb23b117820]
Loaded module File Transfer
2020-03-22_10:13:03.922549 <ALL> Plugin::Plugin("isaccodec",false) [0x7fb23af066e0]
Loaded module iSAC floating point - based on WebRTC iSAC library version 4.3.0 (SPL version 1.2.0)
2020-03-22_10:13:03.948416 <ALL> Plugin::Plugin("javascript",true) [0x7fb23accafe0]
Loaded module Javascript
2020-03-22_10:13:03.986840 <ALL> Plugin::Plugin("jingle",false) [0x7fb23a857a40]
Loaded module YJingle
2020-03-22_10:13:04.000979 <ALL> Plugin::Plugin("cdrcombine",false) [0x7fb23a3c7900]
Loaded module CdrCombine
2020-03-22_10:13:04.009751 <ALL> Plugin::Plugin("rmanager",false) [0x7fb23a1c15c0]
Loaded module RManager
2020-03-22_10:13:04.021535 <ALL> Plugin::Plugin("pbx",false) [0x7fb239fb1480]
Loaded module PBX
2020-03-22_10:13:04.034354 <ALL> Plugin::Plugin("conf",false) [0x7fb239daa480]
Loaded module Conference
2020-03-22_10:13:04.041022 <ALL> Plugin::Plugin("extmodule",false) [0x7fb239b9b940]
Loaded module ExtModule
2020-03-22_10:13:04.051263 <ALL> Plugin::Plugin("socks",true) [0x7fb239987960]
Loaded module YSOCKS
```

Después de iniciar el servidor, tenemos que iniciar el cliente, donde nos moveremos al directorio de clients y ejecutaremos el comando: ./run-qt4



Al ejecutar este comando nos saldrá esta interfaz de usuario, en la cual tenemos que configurar un usuario para poder utilizar el servicio que nos proporcionara Yate. Los usuarios que van a tener acceso a este cliente están registrados en el archivo regfile.conf que esta en el directorio conf.d , que está ubicado en la carpeta raíz de Yate.

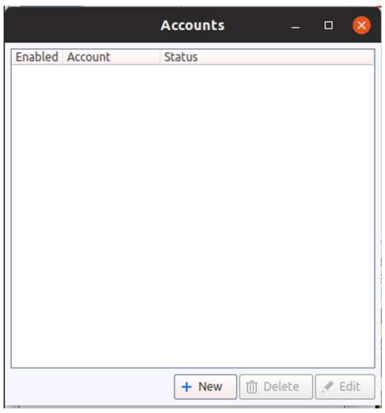
```
[001]
password=001

[002]
password=002
```

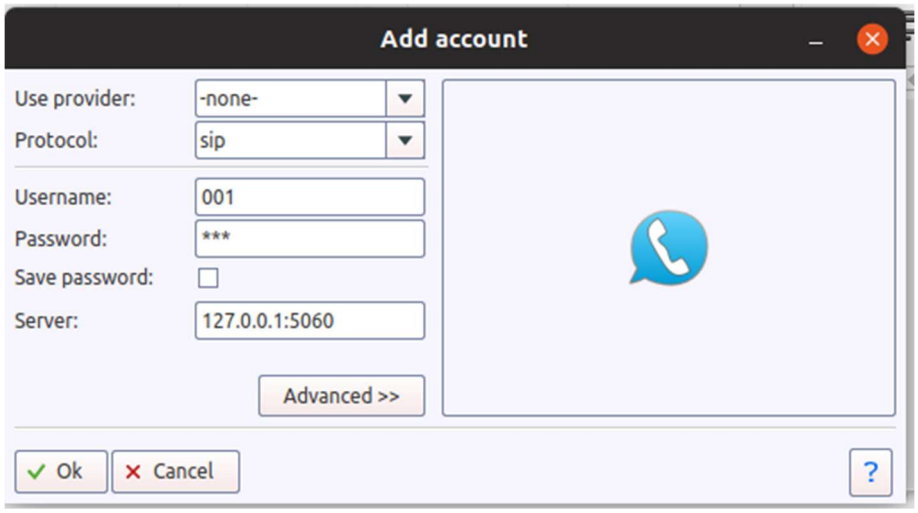
En el archivo ya están preconfigurados dos usuarios, pero nosotros podremos añadir los usuarios que queramos con un id y una contraseña. Ahora vamos a configurar el Yate Client

con algunos de estos usuarios, a continuación mostraremos los pasos que vamos a seguir para este proceso:

- En la ventana principal, seleccionamos la pestaña de Settings y a continuación pincharemos en la opción Accounts del menú desplegable que nos aparece por pinchar en la pestaña de Settings.



- Después, pincharemos en la opción de New para introducir el usuario que queremos conectar al servidor de Yate. Al pinchar en New nos saldra otra ventana donde tenemos que introducir el usuario y contraseña con el cual queremos conectarnos al servidor y la dirección IP del servidor de Yate.



Cuando damos al botón de OK, manda al servidor unas peticiones SIP que vamos a ver a continuación.

1	0.000000000	127.0.0.1	127.0.0.1	SIP	443 Request: REGISTER sip:127.0.0.1:5060 (1 binding) |
2	0.010491041	127.0.0.1	127.0.0.1	SIP	322 Status: 100 Trying |
3	0.015901524	127.0.0.1	127.0.0.1	SIP	522 Status: 401 Unauthorized |
4	0.040236545	127.0.0.1	127.0.0.1	SIP	634 Request: REGISTER sip:127.0.0.1:5060 (1 binding) |
5	0.046582195	127.0.0.1	127.0.0.1	SIP	322 Status: 100 Trying |
6	0.051915646	127.0.0.1	127.0.0.1	SIP	468 Status: 200 OK (1 binding) |

En la imagen superior podemos ver los mensajes que intercambia el cliente con el servidor. Estos mensajes son a través del protocolo SIP, donde el cliente en primera instancia hace un REGISTER en el servidor y el servidor en última instancia da el OK a que se registre ese usuario en el servidor con un código 200. Después de toda esta configuración ya podríamos usar Yate

para llamar a otro usuario que este registrado en el servidor que nos hemos conectado de Yate.

2. (1 punto) A continuación se procederá a configurar dos usuarios (uno para cada miembro de la pareja) en el servidor Yate. Note que Yate permite diferentes modos de configurar usuarios (base de datos, LDAP, fichero local o cualquier otro método que podamos implementar). En este caso esta configuración se realizará usando un fichero de texto.

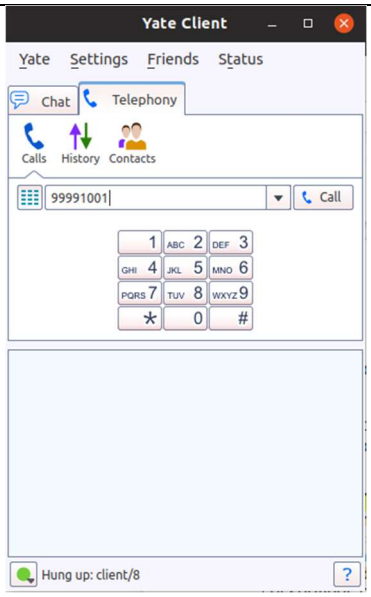
   Asociado a esta configuración, la pareja procederá a configurar el *softphone* Yate client contra el servidor local usando la información de autenticación que acaba de configurar en Yate. Como el servidor y el cliente están en la misma máquina la dirección de servidor que usaremos será 127.0.0.1:5060. Puede comprobar que la configuración es correcta llamando desde el *softphone* a uno de los teléfonos de prueba configurados en Yate, p.ej. El 99991001[1].

   Al llamar a ese número de teléfono se debe establecer la llamada automáticamente y se deberá oír un tono de marcado. El fichero de configuración utilizado debe ser adjuntado a la hora de entregar la práctica[2].

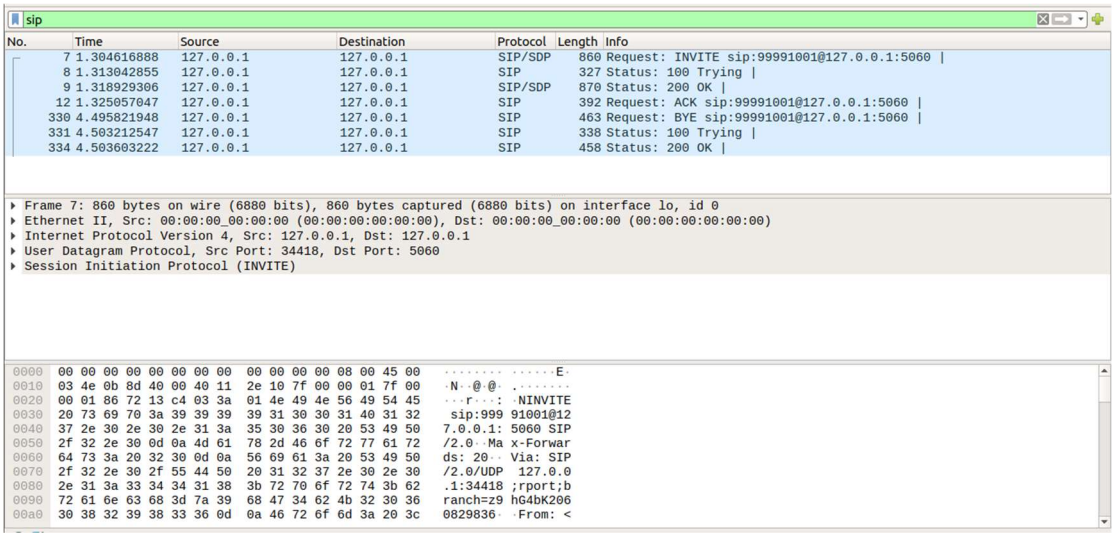
Para ver si hemos hecho la configuración bien, desde Yate Client vamos a hacer una serie de llamadas a los teléfonos de pruebas que están definidos en el archivo regexroute.conf que está en el directorio llamado conf.d, donde le tenemos que quitar la terminación de .sample y reiniciar el servidor para que el servidor coja la funcionalidad de esos teléfonos de prueba.
- El primer paso es introducir en el cliente el número de teléfono que es el 9999 +{1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008}, donde cada uno de estos tiene un tono definido. Por último, pincharemos en el botón de CALL.

1 Para que esto funcione es necesario renombrar el fichero conf.d/regexroute.conf.sample a conf.d/regexroute.conf antes de arrancar el servidor Yate.

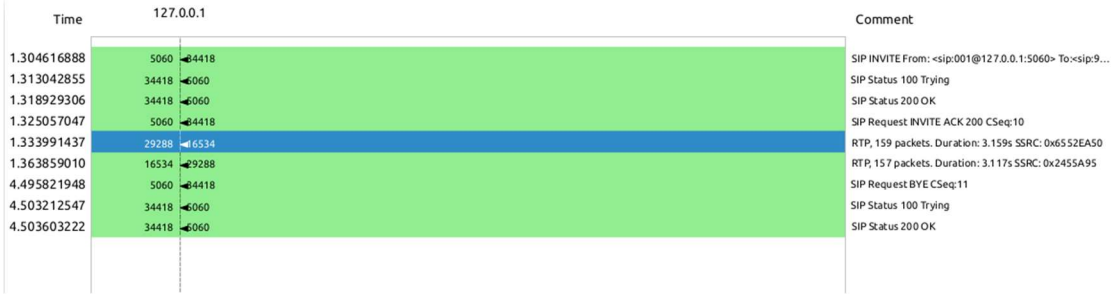
2 La ruta de los ficheros de configuración de Yate se encuentra en la carpeta yate/conf.d. Para que los ficheros sean válidos debe guardarse una copia del fichero sin la extensión .sample.



- A continuación mostraremos como se comparta el servidor cuando hacemos algunas de estas llamadas.



Podemos observar todos los mensajes SIP para aceptar la llamada en el servidor, la comunicación entre cliente y servidor cuando esta en curso la llamada están empaquetados con el protocolo RTP. A continuación, mostraremos un diagrama de secuencia con los mensajes que se han enviado el cliente y servidor en esta llamada de VoIP al teléfono de prueba.



Ahora realizaremos la prueba llamando al usuario 001, es decir, llamarnos a nosotros mismos para sacar el diagrama de secuencia de esa llamada y como realiza el intercambio de paquetes para esta llamada.

sip

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	SIP/SDP	851	Request: INVITE sip:001@127.0.0.1:5060 |
2	0.008427941	127.0.0.1	127.0.0.1	SIP	323	Status: 100 Trying |
3	0.014463706	127.0.0.1	127.0.0.1	SIP/SDP	854	Request: INVITE sip:001@127.0.0.1:34418 |
4	0.040264112	127.0.0.1	127.0.0.1	SIP	308	Status: 100 Trying |
5	0.060750901	127.0.0.1	127.0.0.1	SIP	417	Status: 180 Ringing |
6	0.070599900	127.0.0.1	127.0.0.1	SIP	448	Status: 180 Ringing |
7	4.235897011	127.0.0.1	127.0.0.1	SIP	396	Request: CANCEL sip:001@127.0.0.1:5060 |
8	4.235952312	127.0.0.1	127.0.0.1	SIP	434	Request: BYE sip:001@127.0.0.1:5060 |
9	4.244245668	127.0.0.1	127.0.0.1	SIP	323	Status: 100 Trying |
10	4.244388723	127.0.0.1	127.0.0.1	SIP	334	Status: 100 Trying |
11	4.244791535	127.0.0.1	127.0.0.1	SIP	459	Status: 487 Request Terminated |
12	4.244903622	127.0.0.1	127.0.0.1	SIP	407	Status: 200 OK |
13	4.245041502	127.0.0.1	127.0.0.1	SIP	433	Status: 481 Call/Transaction Does Not Exist |
14	4.250283742	127.0.0.1	127.0.0.1	SIP	400	Request: CANCEL sip:001@127.0.0.1:34418 |
15	4.250344389	127.0.0.1	127.0.0.1	SIP	439	Request: BYE sip:001@127.0.0.1:34418 |
16	4.256046911	127.0.0.1	127.0.0.1	SIP	384	Request: ACK sip:001@127.0.0.1:5060 |
17	4.276233520	127.0.0.1	127.0.0.1	SIP	308	Status: 100 Trying |
18	4.276275376	127.0.0.1	127.0.0.1	SIP	320	Status: 100 Trying |
19	4.276460718	127.0.0.1	127.0.0.1	SIP	428	Status: 487 Request Terminated |
20	4.276493413	127.0.0.1	127.0.0.1	SIP	376	Status: 200 OK |
21	4.276543697	127.0.0.1	127.0.0.1	SIP	402	Status: 481 Call/Transaction Does Not Exist |
22	4.280848388	127.0.0.1	127.0.0.1	SIP	370	Request: ACK sip:001@127.0.0.1:34418 |

A continuación, mostramos el diagrama de secuencia de la llamada que hemos realizado al usuario 001.

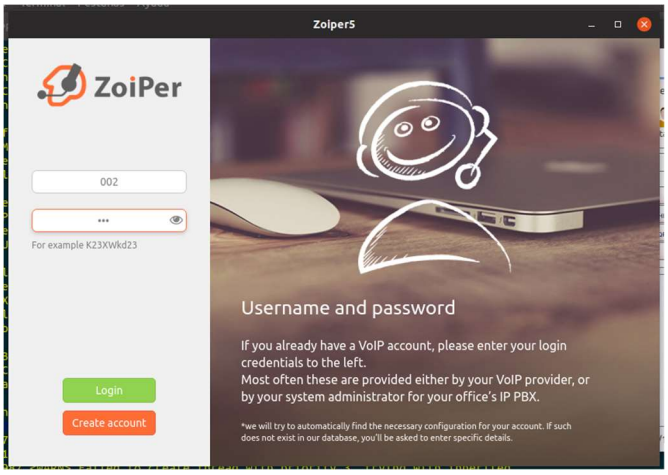
3. (1 punto) Instalar y configurar el softphone Zoiper en un teléfono móvil o en un PC para conectarse con el servidor Yate desplegado. Debe configurar la cuenta indicando que el usuario a usar es usuario@ip_servidor_yate:5060. La IP del servidor debe obtenerla en el equipo del servidor ejecutando el comando ifconfig (si usamos Linux) o ipconfig /all (si usamos Windows). La IP será la de la interfaz WiFi o Ethernet dependiendo de si el equipo está conectado por WiFi o por cable.

   Para la configuración de Zoiper indicaremos que no usamos ningún proxy. Zoiper hace una serie de pruebas para comprobar si se soportan determinados protocolos. Para cada prueba se muestra en rojo, amarillo o verde el resultado. Si la comunicación se puede establecer de manera correcta se mostrará en color verde la opción SIP sobre UDP que es la que usaremos por defecto en esta práctica.

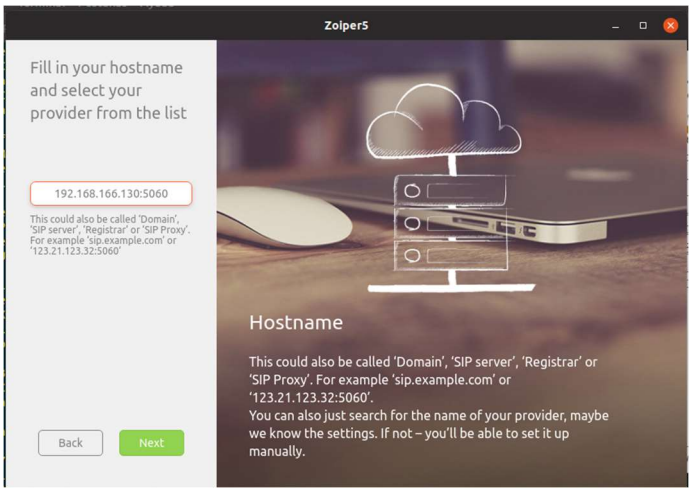
   Se debe documentar el proceso con un pequeño texto y alguna captura de pantalla.

Para configurar el softphone Zoiper, primero tenemos que descargar e instalarlo para poder acceder a su funcionalidad. Después de la instalación, debemos configurar la cuenta para poder conectarnos con el servidor. Tenemos que indicar que nuestro servidor se esta ejecutando en un maquina virtual creada para dar más realidad a esta práctica, esta maquina se esta corriendo con un Ubuntu 18.04.

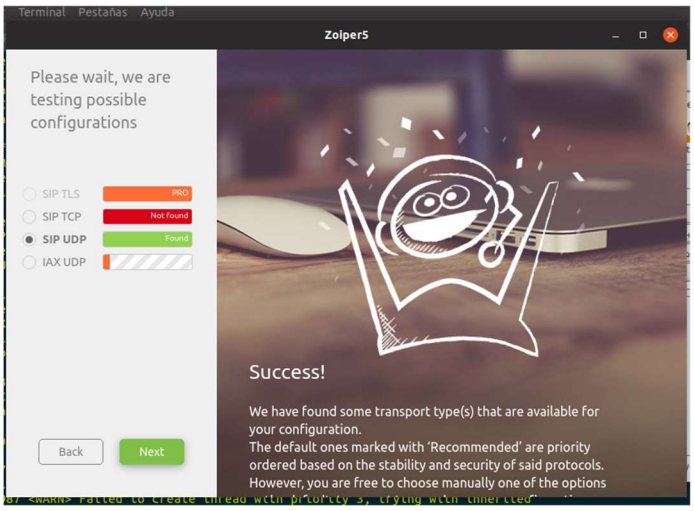
- Primero, tenemos que introducir nuestro usuario y la contraseña de ese usuario.



- Segundo, introduciremos la dirección del servidor Yate y el puerto que tiene ese puerto.



- Tercero, Zoiper nos da las posibles configuraciones que se puede utilizar para la comunicación , pero la comunicación que mejor rendimiento tiene según Zoiper es SIP UDP, por lo tanto es la elegida para hacer la comunicación entre los clientes y el servidor.



Por último, después de configurar Zoiper ya esta listo para poder llamar a cualquier usuario que este dentro del servidor que se esta ejecutando en la dirección IP.

4. (4 puntos) Una vez configurado inicialmente el servidor y los softphones, vamos a proceder a hacer un análisis básico del funcionamiento real del protocolo SIP usando Wireshark. Para ello dispondremos de un ordenador que ejecutará el servidor Yate y un softphone Yate client (al que llamaremos A a partir de ahora) configurado con uno de los usuarios creados. Por otro lado dispondremos de un móvil u ordenador con el softphone Zoiper (al que llamaremos B a partir de ahora) configurado con el otro usuario. A continuación se realizarán una serie de experimentos, **para cada uno de ellos deberá realizar una captura de Wireshark.** A la hora de capturar tenga en cuenta que hay una parte del tráfico que circula en local (Yate client ↔ Yate) y otra que circula a través de la interfaz WiFi o Ethernet (Zoiper ↔ Yate). Por este motivo a la hora de capturar el tráfico en el PC **debemos capturar a la vez de la interfaz de localhost y de la interfaz WiFi o Ethernet a la vez para ver todo el tráfico. Wireshark permite capturar de más de una interfaz a la vez y no es necesario arrancar dos instancias de Wireshark.** Los experimentos a realizar son:

   4.1. Registrar los softphones con el servidor (para ello, ciérrelos previamente y capture el proceso de registro que se lleva a cabo cuando los iniciamos).
   4.2. Llamar desde el softphone B al A. El usuario A deberá aceptar la llamada y verificar que se ha establecido la llamada de voz. Pasados unos segundos el usuario A podrá colgar la llamada.
   4.3. Igual que en el caso anterior, el usuario B llamará al A, pero en este caso, el usuario A rechazará la llamada.
   4.4. Apagamos el softphone A y volvemos a intentar la llamada desde B.

   Para cada uno de estos casos se analizará la captura de Wireshark y se realizarán las siguientes tareas:

   Para cada uno de estos experimentos se analizará la captura de Wireshark y se realizarán las siguientes tareas:

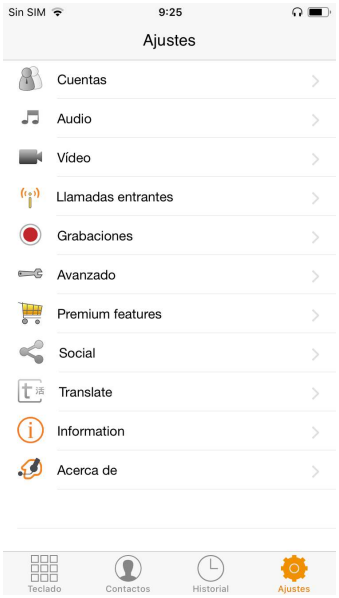
4.4.A) Dibujar un diagrama de secuencia en el que figuren el UA del usuario A, el UA del usuario B y el proxy SIP, y todo el intercambio de mensajes SIP que ha habido entre cada componente.

Para ello tenemos que registrar los dos clientes al servidor, con el cliente Yate ya lo tenemos configurado y toda la comunicación la hace por local, debido a que el servidor y el cliente A están en el mismo ordenador. La comunicación que hace entre ellos es la siguiente:

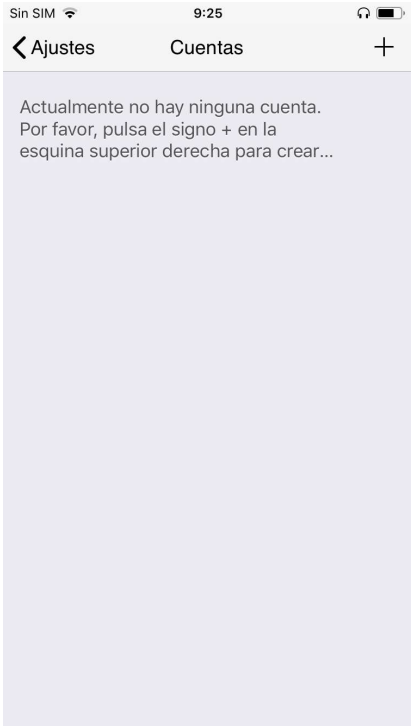
Podemos observar que toda la comunicación SIP lo hace con localhost porque solo hay comunicación con la IP: 192.168.1.104 que es la del ordenador del servidor.

A continuación, mostraremos como configurar el cliente Zoiper en el móvil (iPhone 6) para poder conectarnos al servidor Yate.

- Primero, nos descargamos la aplicación del App Store y cuando este descargada e instalada, abrimos la aplicación y nos vamos a la sección de Ajustes:



- Segundo, pulsamos en cuenta y nos aparecerá esta pantalla, donde le daremos al más y seleccionaremos la opción de Configuración manual.



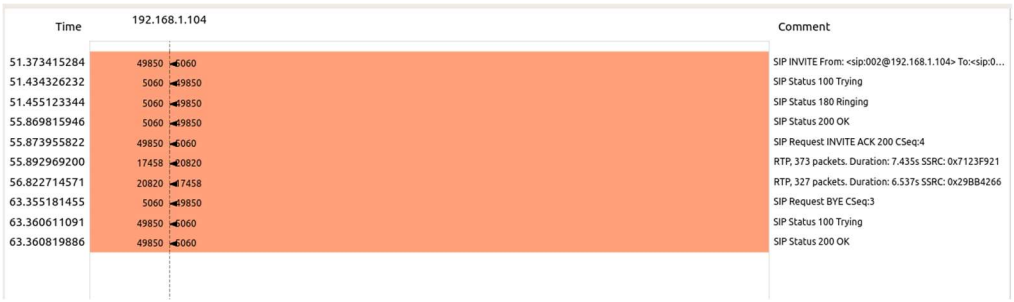
- Tercero, después de de pulsar a Configuración manual nos saldrá la siguiente pantalla. Donde rellenaremos los siguientes datos con el usuario, contraseña y el servidor en donde se esta ejecutando el Yate.

- Por último, nos aparece este mensaje para ver que estamos registrados en el servidor.

A continuación, mostraremos el flujo de información que tiene el servidor con el cliente. Ahí vemos que la información va por la red Wifi esta comunicación.

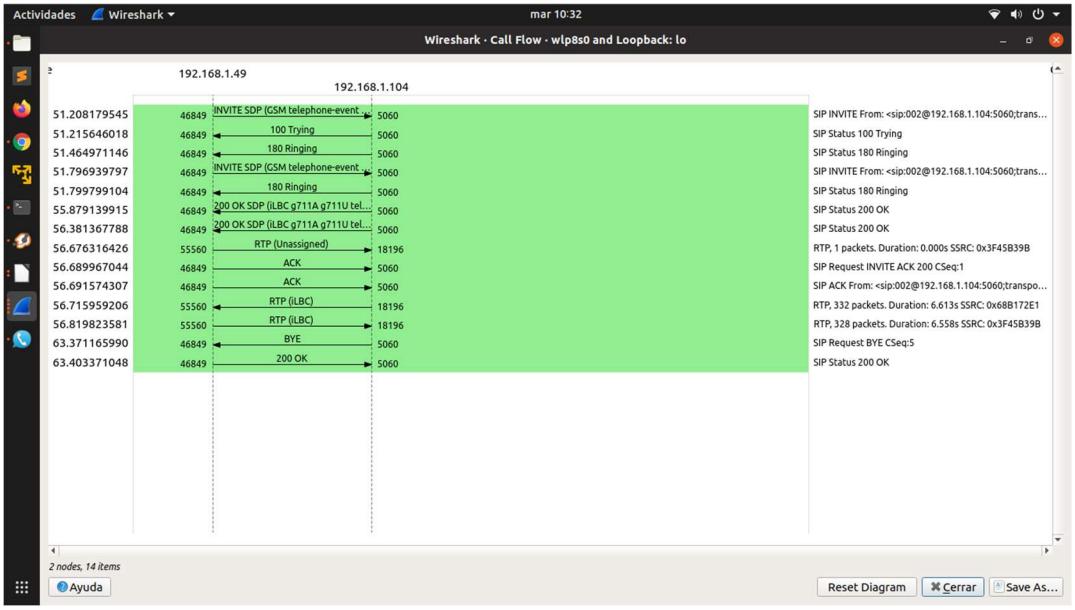
Ahora vamos a ver los diagramas de las llamadas que se hacen y las acciones de los dos clientes durante la llamada:

- Cliente A:



Aquí podemos ver toda la comunicación que hace el cliente A con el servidor a la hora de conectarse la llamada con el cliente B, donde el cliente B nos llama y el cliente A coge la llamada. Por último, podemos observar como envia un mensaje de BYE para cortar la comunicación de la llamada.

- Cliente B:



Aquí podemos observar como es la comunicación de la llamada entre el Cliente B y el Cliente A, donde podemos observar como el Cliente B envía paquetes RTP al Cliente A donde esta contenida la pulsos de voz. Podemos observar como el Cliente A cuelga la llamada.

Ahora vamos a observar el siguiente caso que el Cliente A cuelga una llamada al Cliente B sin empezar a conversar.

-Cliente B:

- Cliente A:

Como podemos observar en los diagramas cuando un cliente cuelga a otro manda un mensaje SIP de Busy Here, diciéndole que esta ocupada y no puede coger la llamada.

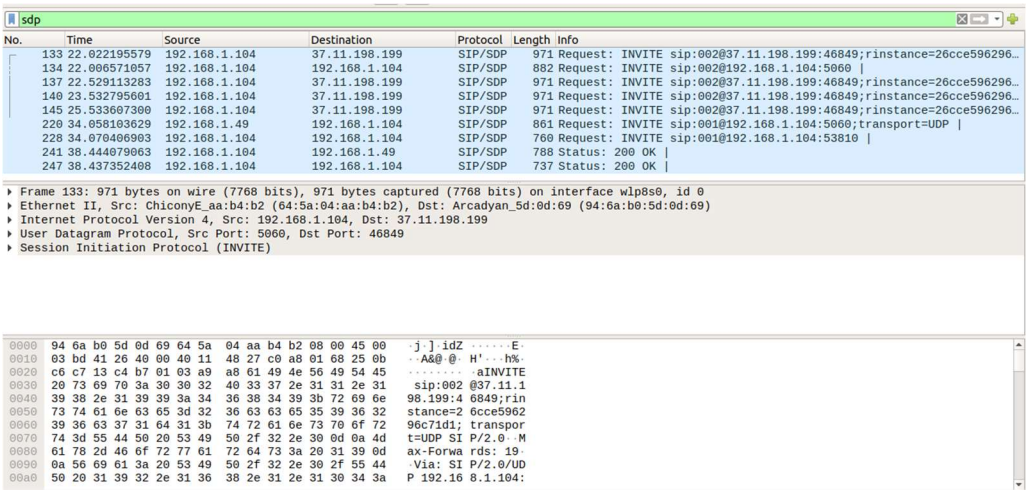
Ahora vamos a probar el caso de que no este conectado el Cliente A, por lo tanto cerramos el Cliente A y empezamos a capturar tráfico, y a continuación, llamaremos con el Cliente B al Cliente A para ver que mensajes SIP envía entre ellos.

Como podemos observar enviamos un mensaje Not Found, debido a que el servidor no encuentra ningún usuario que este conectado con esa id, por lo tanto no se puede establecer como queremos la llamada.

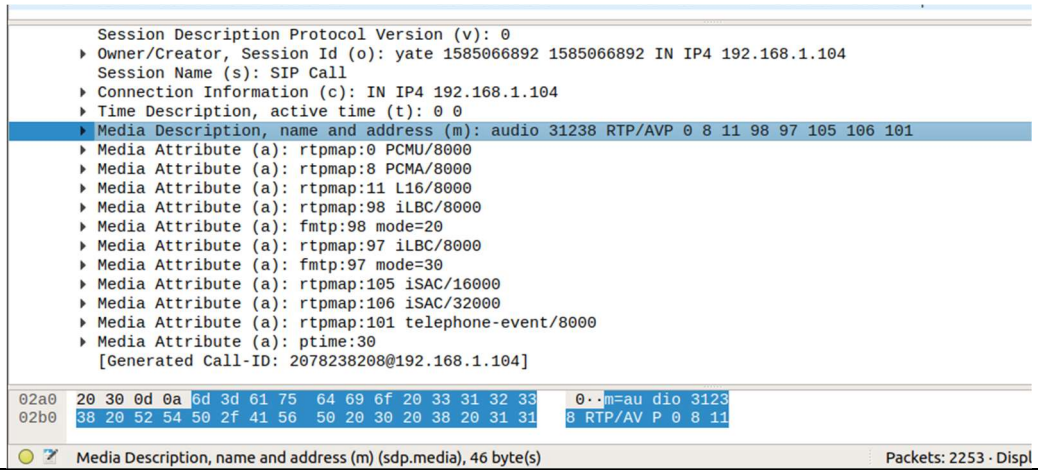
4.4.B) Recuperar el contenido del mensaje SDP (embebido en algunos mensajes SIP) que se intercambia cada UA y de ahí inferir los tipos de datos/códecs multimedia que cada UA ofrece, y determinar el tipo de códec y tasa binaria finalmente usados para los datos de audio intercambiados. Para esta última parte será necesario analizar el tráfico RTP asociado (si lo hubiera).

Para coger el contenido del mensaje SDP tenemos que hacer una llamada entre el Cliente A y Cliente B, debido a que estos mensajes se envían con el método INVITE de SIP de la solicitud. Por lo tanto, lo que hemos hecho es que el Wireshark capture de las dos trazas a las que están conectados los dos clientes y filtrar por los paquetes SDP, como podemos ver en la imagen siguiente.

Nos



ofrecen varios tipos de codecs para el envio del audio y son los siguientes:



El tipo de codec que se esta utilizando es el G.711, debido a que se puede ver en la herramienta de Telephony que tiene Wireshark, donde nos muestran los diagramas que hace una llamada de VoIP.
El codec G.711 envía a una tasa binaria de 87,2 Kbps. Con un tamaño de paquetización de 20 ms, enviando 50 paquetes por segundo y un tamaño de paquetización de 160 bytes.

4.4.C) Determinar direcciones IP y puertos de origen y de destino de los flujos RTP intercambiados, si los hubiera.

Esta información es obtenida con la traza que sacamos de una llamada de Wireshark, y tendríamos que mirar en la cabecera del mensaje donde podremos obtener las direcciones IP y puertos.
Las IP's que son origen y destino en todos los mensajes RTP que se envían entre los dos clientes son: 192.168.1.104 y 192.168.1.49, estas IP's corresponde a las dos direcciones IP's que tienen los clientes.
Los puertos origen y destino más utilizados son los siguientes: 55560, 21112, 32434 y 28234.

Utiliza el protocolo iLBC que es el codec Internet Low Bitrate Codec que envía a una tasa binaria de 13,33 o 15,2 Kbps dependiendo el tamaño de paquetización y el tiempo de paquetización.

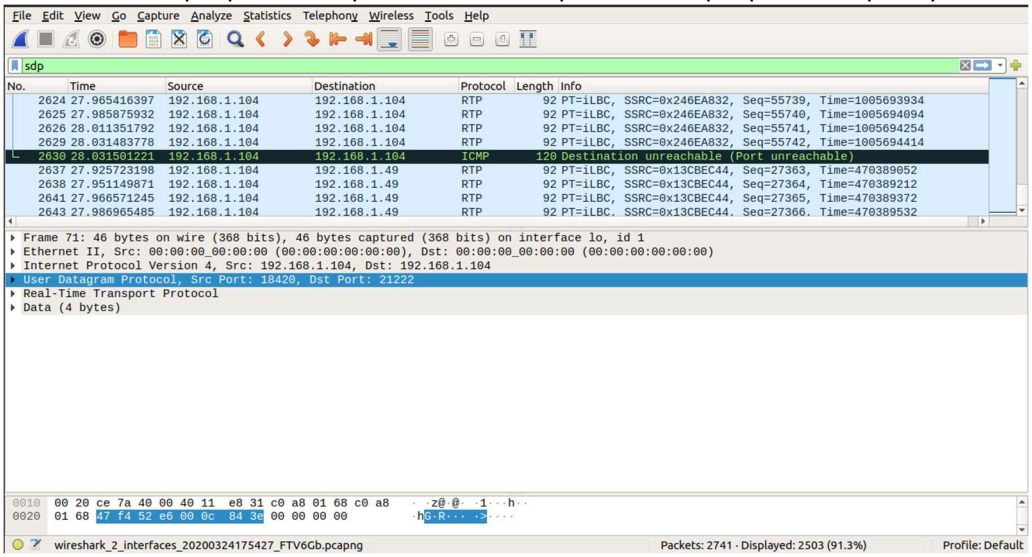
Para el caso de una llamada establecida correctamente (4.2) se realizarán además las siguientes tareas:

4.4.D) Haciendo uso de Wireshark reconstruir una llamada establecida y escuchar el audio transportado a partir del tráfico capturado. Indicar los pasos necesarios para escuchar el audio. Wireshark solo es capaz de reproducir llamadas que hagan uso de códecs libres. Si su llamada usa un códec que necesite licencia no podrá escuchar nada. Típicamente las llamadas se establecerán con el códec G.711 que es libre y puede ser reproducido por Wireshark.

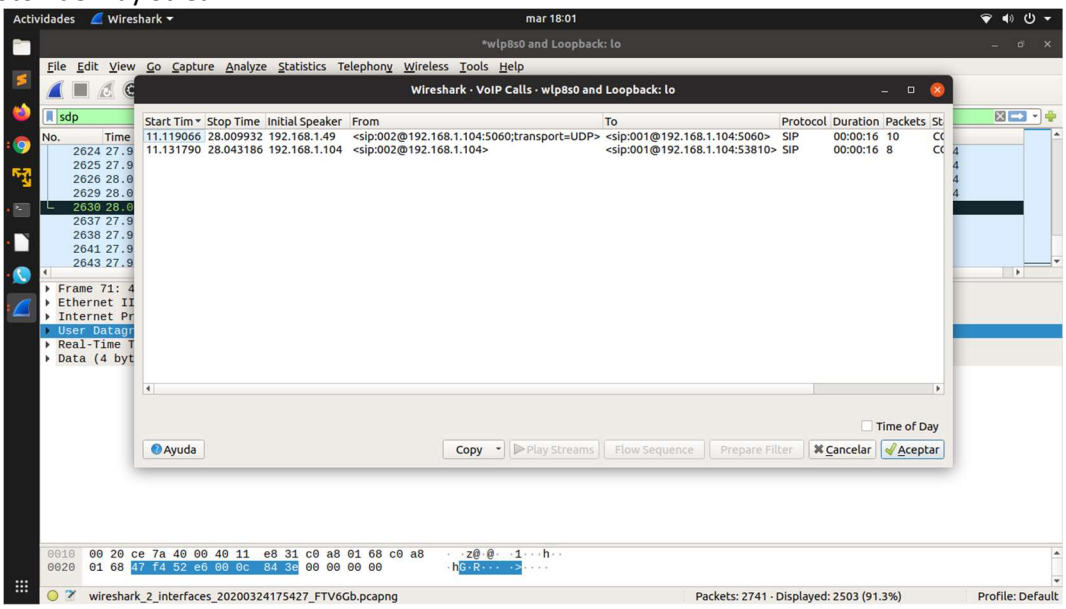
Los pasos necesarios para reconstruir una llamada que hemos capturado por Wireshark hay que seguir los siguientes pasos:

- Primero tenemos que parar la captura e irnos a la pestaña de que pone Telephony.

-



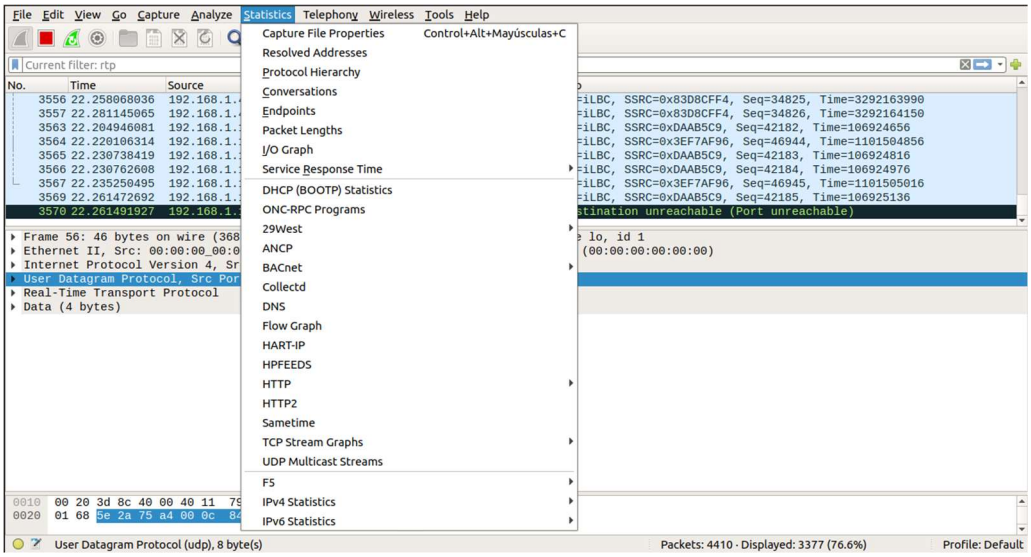
Después, nos saldrá un despegable donde tendremos que pinchar sobre la opción VoIP Calls, donde al clickear en esta opción nos saldrá esta ventana que nos saldrá todas las llamadas que hemos tenido y para poder reconstruirlas solo tendremos que pinchar sobre ellas y dar al botón de Play Stream.



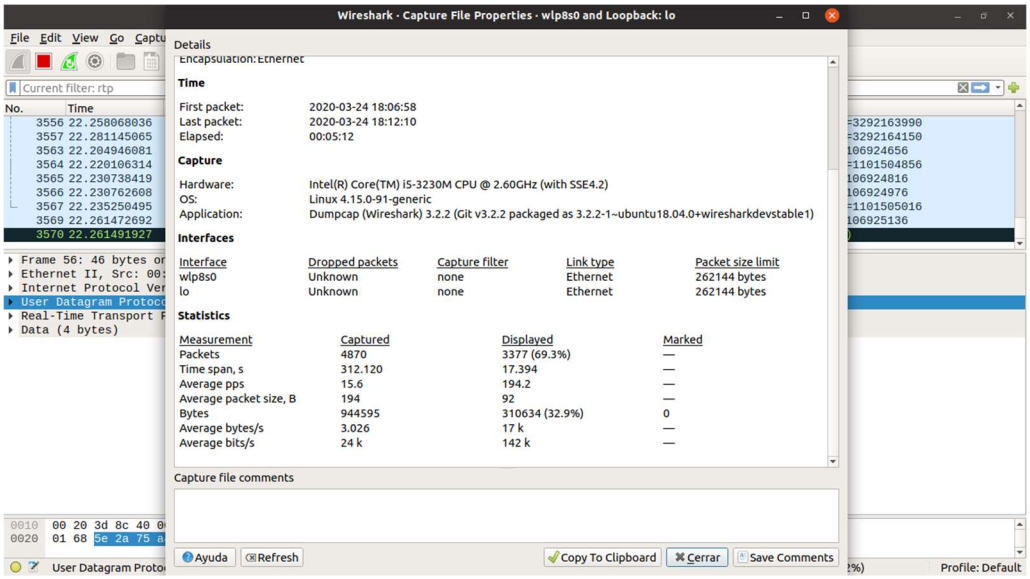
- Y por último, cuando pinchamos nos saldrá otra pantalla donde solo tenemos que pulsar el botón de play para reproducir la llamada.

4.4.E) Haciendo uso de las herramientas de Wireshark, consiga una estimación del jitter y la tasa de pérdida de paquetes para una llamada establecida correctamente. Para ello se recomienda usar las opciones contenidas en el apartado “Telephony” de Wireshark. ¿En qué se diferencia la estimación del jitter de la que se realizaba en la práctica anterior?

- Primer paso es capturar el tráfico de la llamada y en la parte superior tendremos que dar a Statistic y después seleccionar la opción de Capture File Properties, o también, hacer la combinación de letras Control+Alt+Mayuscula+C.



Después de pinchar en esta opción o abrir la pestaña por el atajo de teclas, nos saldrá esta pantalla con todo lo referente a las medidas de calidad que hace Wireshark.



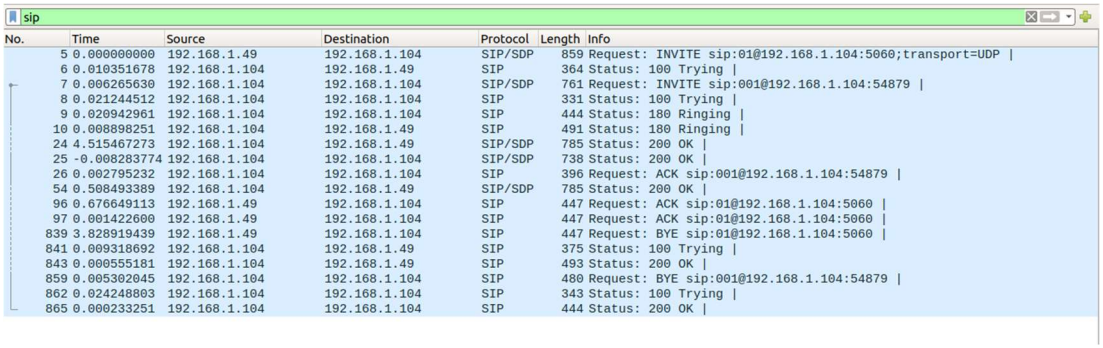
La diferencia con la práctica 2 es que el jitter se calculaba a partir de conseguir a la tasa máxima a la que podemos enviar y luego ejecutar el clienteTren2 con una tasa que sea menor a que esa tasa máxima. Pero en está práctica encontramos el jitter sin ninguna prueba anterior, debido a que siempre enviamos a la misma tasa, es decir, la que marca el codec que estamos utilizando para la llamada.

5. (2 puntos) A continuación vamos a analizar ciertas funciones de rutado de llamadas. El rutado de llamadas es una funcionalidad que realiza internamente el servidor SIP y que permite, por ejemplo, asociar una extensión corta a un usuario para que al llamarle sea suficiente con escribir la extensión corta en vez de toda la URI SIP. Para ello se realizarán las siguientes tareas:
   5.1. Se asociará a cada uno de los dos usuarios creados anteriormente una extensión corta (01 al primero y 02 al segundo). Para ello se podrá usar o bien la configuración de rutado usando expresiones regulares (regexroute.conf) o usar javascript (javascript.conf). Adjunte la expresión regular o el fichero javascript usado para rutar la llamada.

Para configurar rutas cortas hemos utilizado el cambiar el fichero regexroute.conf, donde le hemos añadido dos expresiones regulares para poder hacer la marcación corta con el usuario 1 y el usuario 2 registrados en el servidor Yate. Para realizar estos cambios tenemos que ir al directorio llamado conf.d/ y alli modificar el fichero regexroute.conf. Para ello le añadiremos las siguientes lineas:
- ^01$=return;called=001
- ^02$=return;called=002

   5.2. Realice una llamada de prueba entre los dos usuarios mientras captura el tráfico con Wireshark. Analice la parte del tráfico que proviene del softphone Zoiper. Explique qué cambia en los paquetes SIP intercambiados con respecto al apartado 3

Como podemos observar cambia la dirección del INVITE, donde pone la abreviatura que hemos puesto y luego hace el INVITE con el usuario real, y cuando queremos el ACK lo hace a la abreviatura y el mensaje de BYE lo hace al usuario sin la abreviatura.



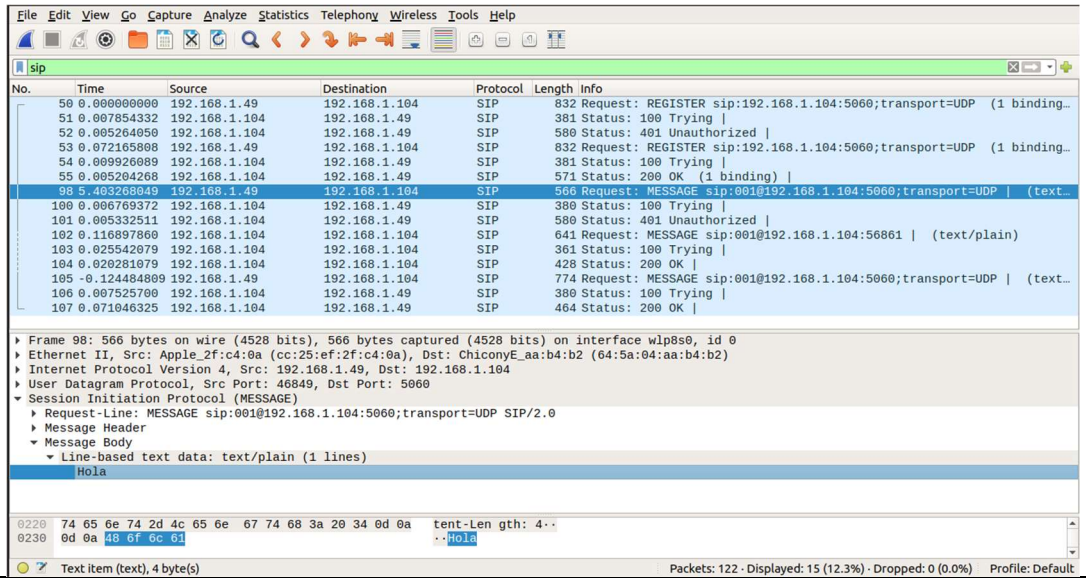
sip

No.	Time	Source	Destination	Protocol	Length	Info
5	0.000000000	192.168.1.49	192.168.1.104	SIP/SDP	859	Request: INVITE sip:01@192.168.1.104:5060;transport=UDP |
6	0.010351678	192.168.1.104	192.168.1.49	SIP	364	Status: 100 Trying |
7	0.006265630	192.168.1.104	192.168.1.104	SIP/SDP	761	Request: INVITE sip:001@192.168.1.104:54879 |
8	0.021244512	192.168.1.104	192.168.1.104	SIP	331	Status: 100 Trying |
9	0.020942961	192.168.1.104	192.168.1.104	SIP	444	Status: 180 Ringing |
10	0.008898251	192.168.1.104	192.168.1.49	SIP	491	Status: 180 Ringing |
24	4.515467273	192.168.1.104	192.168.1.49	SIP/SDP	785	Status: 200 OK |
25	-0.008283774	192.168.1.104	192.168.1.104	SIP/SDP	738	Status: 200 OK |
26	0.002795232	192.168.1.104	192.168.1.104	SIP	396	Request: ACK sip:001@192.168.1.104:54879 |
54	0.508493389	192.168.1.104	192.168.1.49	SIP/SDP	785	Status: 200 OK |
96	0.676649113	192.168.1.49	192.168.1.104	SIP	447	Request: ACK sip:01@192.168.1.104:5060 |
97	0.001422600	192.168.1.49	192.168.1.104	SIP	447	Request: ACK sip:01@192.168.1.104:5060 |
839	3.828919439	192.168.1.49	192.168.1.104	SIP	447	Request: BYE sip:01@192.168.1.104:5060 |
841	0.009318692	192.168.1.104	192.168.1.49	SIP	375	Status: 100 Trying |
843	0.000555181	192.168.1.104	192.168.1.49	SIP	493	Status: 200 OK |
859	0.005302045	192.168.1.104	192.168.1.104	SIP	480	Request: BYE sip:001@192.168.1.104:54879 |
862	0.024248803	192.168.1.104	192.168.1.104	SIP	343	Status: 100 Trying |
865	0.000233251	192.168.1.104	192.168.1.104	SIP	444	Status: 200 OK |

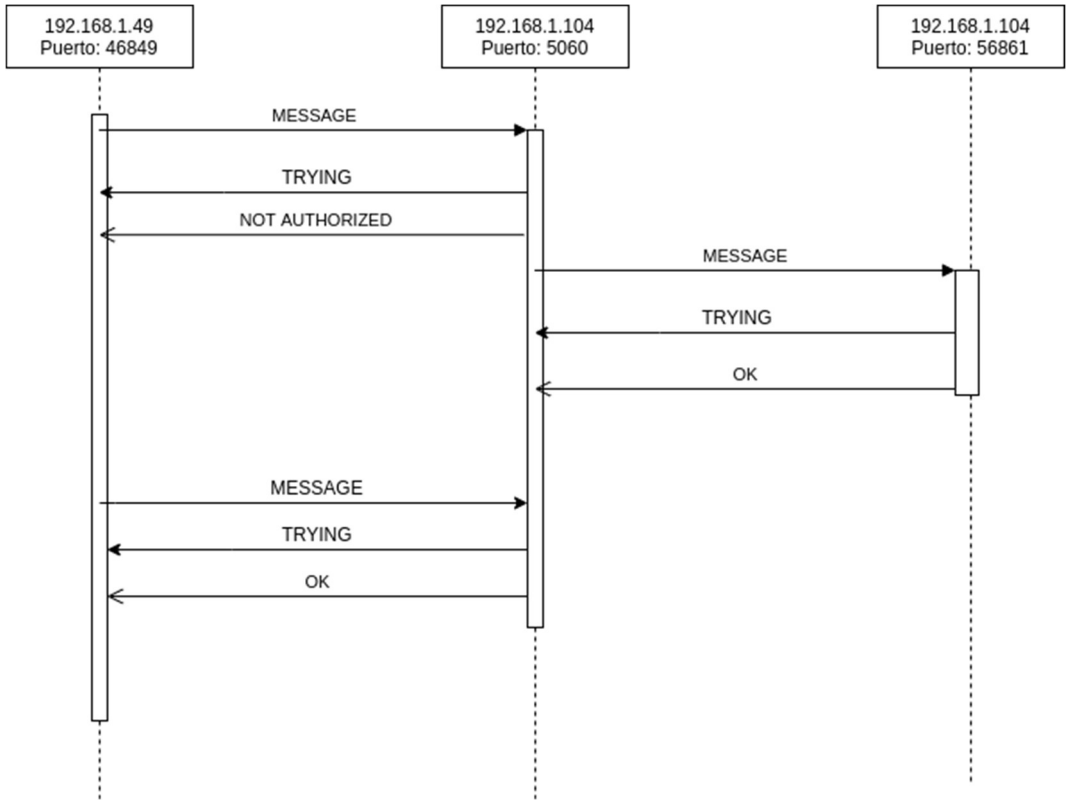
El protocolo SIP no permite únicamente establecer sesiones de audio y vídeo sino que permite también el envío de mensajes instantáneos. A continuación se configurará Yate para soportar esta funcionalidad. Debe investigar en la documentación y realizar los siguientes ejercicios:

6. (2 puntos) Configurar Yate para que permita el intercambio de mensajes usando SIP. Enviar un mensaje desde B a A y capturar el tráfico asociado. El softphone Yate no permite mostrar mensajes que se han recibido y no es necesario ver el texto en el programa para realizar los siguientes ejercicios. Si quiere visualizar los mensajes puede instalar otro softphone con soporte para texto como Jami (https://jami.net/).

No hay que hacer nada al servidor Yate, debido a que ya soporta mensajes de texto, por lo tanto no se realizaría ningún cambio. A continuación, mostraremos la traza que se realiza con el envío de mensajes de texto.



6.1. Dibujar un diagrama de secuencia en el que figuren el UA del usuario A, el UA del usuario B y el proxy SIP, y todo el intercambio de mensajes SIP que ha habido entre cada componente.



Aquí podemos observar el intercambio de mensajes entre los dos clientes y el servidor, para el envío de los mensajes de textos entre en los clientes

6.2. Recuperar el contenido del mensaje SIP que se intercambia cada UA y analizar sus contenidos ¿Qué diferencias se observan respecto al caso del establecimiento de una llamada de VoIP? ¿Observa alguna diferencia en el uso del protocolo SDP?

Las diferencias es que no se tiene que establecer una comunicación entre ellos y todos esos mensajes pasan por el servidor. No hace falta el uso de SDP. El mensaje que queremos enviar esta codificado en el cuerpo de la cabecera SIP que mandamos al servidor.

3 Conclusiones

Lo que hemos aprendido en esta práctica es saber cómo instalar y configurar un servidor Yate para poder realizar llamadas VoIP y envío de mensajes de dos clientes que utilicen este servidor. También hemos sabido configurar un softphone para que se conecte al servidor y así poder realizar llamadas entre los usuarios conectados al servidor. En el servidor también hemos podido configurar la marcación rápida para los distintos usuarios que hay en el servidor. Por último, hemos sabido manejar Wireshark para ver todo el tráfico en la red que hacen los clientes para realizar las llamadas VoIP. También hemos podido restaurar llamadas a través de un plugin de Wireshark y poder reproducirlas si están tenían un CODEC gratuito.