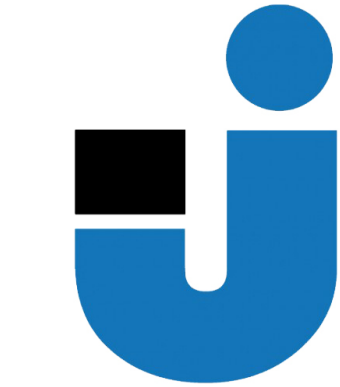


Universidad Nacional
ARTURO JAURETCHE

Redes de Computadoras II

TP Final

Emiliano Salvatori Nicolás Monasterio
Guillermo Cruz Mauricio Mamani

Junio del 2020

Índice

1. Introducción

La incorporación de contenido multimedia siempre fue una idea que estuvo presente en el mundo de la informática, desde los primeros tiempos de los años 70; pero los altos costos y las limitaciones técnicas constituían una gran dificultad que atentaba contra este deseo. Los principales desafíos técnicos para afrontar este reto consistían en: por un lado, tener una capacidad de procesamiento suficiente para el tratamiento de audio y video, y por el otro, un ancho de banda que soporte una alta tasa de datos de transferencia.

Se tuvo que esperar sobre el final de la década de los 90 para poder contar con la tecnología suficiente (por ejemplo la apertura multimedia dada por Apple en 1991, la difusión de Internet y su salto cuantitativo en el ancho de banda, el desarrollo de protocolos y el uso de sus estándares como por ejemplo TCP/IP, HTTP, UDP, etc) para afrontar el gran desafío de transmitir datos en tiempo real de contenido multimedia.

Fue recién con la aparición de Real Networks en 1995 que se comenzó a esgrimir una solución para las transmisiones multimedia sobre internet. En este año, Real Networks logró transmitir por primera vez el audio de un partido de baseball entre los Yankees y los Seattle Mariners. Fue tal el impacto que causó este acontecimiento que el *streaming* comienza a pensarse en sus inicios, como tecnología dedicada a la transmisión de audio; y muy posteriormente, para video.

Esto significó un cambio de paradigma en Internet, dado que ampliaba sus capacidades al permitir acceder a contenidos audiovisuales sin necesidad de una descarga previa. Junto con los enormes desarrollos en áreas tales como telefonía móvil, dispositivos fotográficos, instrumentos de audio, y mayores accesos a servicios de banda ancha, las telecomunicaciones y la era digital acapararon todo el globo. Dado este avance tecnológico que posibilitó la transmisión de datos de forma más rápida, acceso a computadoras hogareñas con mayor poder de procesamiento y sistemas operativos más especializados, se logró de manera progresiva el tan aclamado sueño de los años 70: la transmisión de contenido multimedia a gran escala.

Pero no fue hasta mediados de la década del 2000 que todo ello se materializó en una idea iconoclasta: Youtube. La página presentó por primera vez la posibilidad de compartir audio y video para cualquier internatua; de una forma fácil y sencilla, la página permitía la búsqueda de cualquier contenido multimedia, con una interfaz accesible y minimalista y con un atractivo que hasta la fecha parecía imposible: no era necesario ninguna descarga previa del contenido para poder interactuar con el. El usuario podía controlar el flujo de datos con los comandos típicos de un video VHS, mediante las opciones de PLAY, PAUSE, o STOP.

Todo ello fue y sigue siendo aún posible (más aún hoy en día con el constante surgimiento nuevas redes sociales tales como Facebook, Instagram, Twich.tv, etc), no sólo gracias al avance en hardware, sino también gracias al desarrollo de mejoras constantes en lo que se refiere a protocolos de comunicación entre computadoras. Algunos de los protocolos que permiten que todo esto, que hace tan sólo unas décadas atrás parecía imposible se vuelva realidad, son los protocolos denominados como *Protocolos de Tiempo Real*, tales como RTP, RTCP o RTSP.

En los siguientes apartados se intentará dar un acercamiento al desempeño de estos tres protocolos, y más específicamente a RTSP, los cuales no solo permiten la

difusión de contenido audiovisual, sino también el control del usuario sobre ellos. Cabe aclarar que, para el correcto abordaje sobre el protocolo RSTP será necesario plantear de forma conjunta el protocolo que se utiliza comúnmente junto con este, el denominado Protocolo de Tiempo Real (Real Time Protocol) y el Protocolo RTP de Control.

2. Protocolos RTP y RTCP

2.1. Introducción al Protocolo de Tiempo Real (RTP)

El protocolo de Transporte en Tiempo Real (por sus siglas en inglés RTP) es un protocolo de Capa de Transporte para tráfico multimedia comúnmente utilizado en Internet. RTP, detalla un formato estándar de paquete para transmitir audio y video sobre la red. Es utilizado en sistemas de transmisión de datos multimedia (en conjunto con el protocolo RTCP), en conferencias de video, o en telefonía IP.

El estándar de RTP actualmente define el siguiente par de protocolos de forma simultánea: RTP propiamente dicho, y RTCP. El primero es utilizado para el intercambio de datos multimedia (audio y video), mientras que el segundo, es la parte encargada de controlar el flujo de datos y que de forma periódica, obtiene información de control sobre la calidad de la transmisión asociada con ellos.

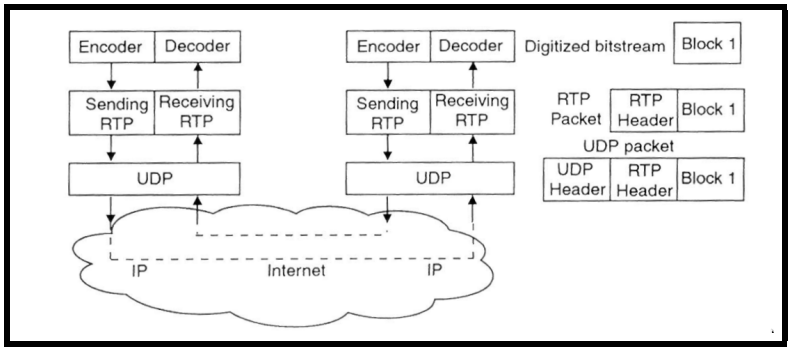
RTP se basa en un envío ”poco fiable”, es decir de mejor esfuerzo, en tiempo real de datos sobre UDP. Se suele usar junto con RTSP para realizar las tareas de control del flujo de datos mediante sesiones. En contra de lo que su nombre pueda sugerir, la utilización del protocolo RTP no asegura que la transmisión sea en tiempo real, pero sí aumenta la sincronización y el control sobre los datos transmitidos (de audio y video). Es decir, que se pueden enviar el audio y el video por ”flujos” separados y mediante RTP, sincronizarlos posteriormente en el destino.

RTP en sí mismo no proporciona mecanismos de entrega en tiempo u otras como por ejemplo, Calidad de Servicio (denominadas por sus siglas en inglés QoS); para ello se ampara en servicios de bajo nivel para su implementación. Tampoco garantiza o previene una transmisión de datos en orden (lo que significa que los paquetes RTP pueden viajar desordenados en todo el camino), y por lo tanto, la confiabilidad de la red subyacente se vuelve incierta. Pero, por otro lado, el número de serie en RTP permite al receptor reorganizar la secuencia de paquetes en el lado del remitente, y también puede ser utilizado para determinar la ubicación del paquete apropiado como por ejemplo, en la decodificación de video.

El protocolo consta de dos partes estrechamente vinculadas: RTP transmite datos con propiedades de tiempo real, y el Protocolo de Control de RTP monitorea la calidad del servicio, encargándose de transmitir todo lo que sucede entre los participantes de la sesión. RTP se emplea casi siempre bajo el protocolo de transporte UDP/IP y dependiendo de las circunstancias en que la aplicación lo requiera, también puede utilizarse TCP. Tanto RTP como RTCP usan puertos de la capa de aplicación cuando utilizan UDP.

La siguente figura ilustra de forma esquemática el uso de RTP. El audio y video es digitalizado utilizando un codec particular; una vez realizado esta codifica-

ción/decodificación, cada bloque de bits es encapsulado bajo un paquete RTP y luego bajo un paquete UDP.



El cuerpo de datos perteneciente a RTP provee soporte para aplicaciones con características de tiempo real para transmisión continua de audio y video, lo que permite la reconstrucción temporal de paquetes, la detección de pérdidas, seguridad e identificación de contenido. Pero hay que tener en cuenta, como se dijo anteriormente, que RTP no reserva ancho de banda para lo que comúnmente se denomina como "Calidad de Servicio".

Quien provee esto último es el protocolo RTCP, el cual proporciona soporte para conferencias en tiempo real en grupos de cualquier tamaño en internet. Este protocolo ofrece información constante sobre la calidad de la transmisión, la cual obtiene de los participantes a los que se les transmite los datos multimedia. RTCP también mantiene información de todos los que intervienen en la transmisión.

RTP está diseñado para soportar una gran variedad de aplicaciones. Provee mecanismos flexibles que permiten que estas puedan ser desarrolladas sin tener que ajustarse al protocolo, sino todo lo contrario, será el protocolo quien se adapte a las aplicaciones. Lo hace gracias a un mecanismo por el cual RTP define un perfil y uno o más formatos para la carga útil del paquete. El perfil provee un rango de información que asegura el común acuerdo entre los campos de la cabecera de RTP para esa clase de aplicación. Esta asimismo, puede definir, extender o modificar el protocolo para que se adapte a una clase especial de aplicaciones. En la especificación sobre el formato de la carga útil, se explica cómo la información que sigue a la cabecera de RTP debe ser interpretada.

2.1.1. Mezcladores y Traductores

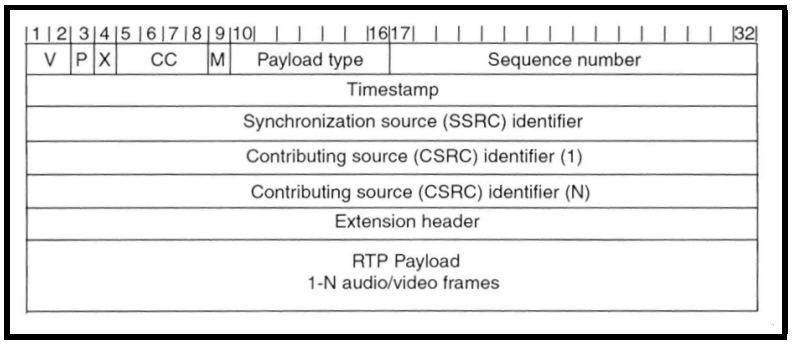
Además de definir cómo se deben comportar tanto el remitente como el receptor, RTP también define otros dos roles dentro de un sistema de comunicación en tiempo real, aquellos denominados **Mezcladores** (*mixers*) y **Traductores** (*translators*). Estos dos roles, se sitúan entre las dos cabeceras de la comunicación y actúan sobre los paquetes de tipo RTP a medida que van intercambiándose.

El traductor es utilizado para traducir de una carga útil a otra. Por ejemplo: supongamos que uno de los participantes de una videoconferencia tiene muy poco acceso

al ancho de banda requerido para la transmisión. En este caso, el *traductor* puede convertir el flujo de datos a un formato de audio y video que requiera menos ancho de banda. Por otro lado, el *mezclador* es útil para combinar múltiples fuentes de datos en una sola. Un ejemplo de ello es cuando se combinan distintas imágenes de distintos usuarios en una sola, simulando una escena de grupo, dentro de una videoconferencia.

2.2. Formato de Cabecera RTP

La siguiente imagen, muestra el formato de la cabecera de los paquetes para RTP. Los primeros 12 bytes están siempre presentes, mientras que la parte denominada como *Contribuiting source Identifier* (Identificador de Fuente colaboradora, o por sus siglas en inglés CSRC) se utiliza en determinadas circunstancias.



Luego de esta cabecera, pueden existir partes opcionales. Finalmente, la carga útil del paquete RTP, cuyo formato está determinado por la aplicación como se comentó más arriba, sigue a la cabecera. Esta cabecera contiene sólo los campos que son probables que sean utilizados por la mayoría de las aplicaciones multimedia.

- **Version (V)**: *Versión* 2 bits.

  Estos bits identifican la versión del protocolo RTP, cuyo valor es el 2 para la versión RTP al momento de escribir este trabajo.

- **Padding (P)**: *Relleno* 1 bit.

  Si el bit de *padding* se encuentra activado, se indica que el paquete contendrá uno o más octetos de relleno sobre el final, los cuales no serán parte de la carga útil. El último octeto contiene un contador de cuántos de estos octetos deberán de ser ignorados. Los datos que transporta pueden ser rellenados hasta completar un bloque de cierto tamaño requerido por algún algoritmo de encriptación.

- **Extension (X)**: *Extensión* - 1 bit.

  Si este se encuentra activado, entonces se indica que un encabezado fijo deberá de ir seguido de un encabezado de extensión.

- **CSRC Count(CC)**: *Contador para la Fuente colaboradora* - 4 bits.

  Este campo contiene el número de identificadores de CSRC que siguen al encabezado.

- **Marker(M)**: *Marcador* - 1 bit.

  La interpretación del marcador está definida por el perfil. Su objetivo es permitir que eventos importantes, como por ejemplo, los límites de trama, sean marcados en la secuencia de paquetes. Un perfil PUEDE definir bits marcadores adicionales o especificar que no hay bits marcadores al cambiar el número de bits en el campo de tipo de carga útil.

- **Payload Type (PT)**: *Identificación de Carga útil* - 7 bits

  Este campo identifica el formato de la carga útil del paquete RTP y determina su interpretación por parte de la aplicación que lo utilice. Un perfil puede especificar una asignación estática predeterminada de códigos de tipo para formatos de carga útil. Un posible uso de este campo podría ser el habilitar una aplicación para que cambie de un esquema de código a otro basado en información sobre la calidad de la red sobre la que transmite y que pueda obtener de alguna otra aplicación. Los tipos de carga útil pueden ser: PCM, MPEG1/MPEG2, video JPEG, etc. Más tipos de carga útil pueden ser agregados configurando un perfil y un esquema de formato de carga útil.

  Cabe aclarar, que el protocolo establece que el receptor en caso de recibir tipos de Carga útil que no entiende, deben ser descartados.

- **Sequence Number** : *Número de Secuencia* - 16 bits

  El número de secuencia aumenta en uno por cada paquete de datos RTP enviado, y el receptor lo utiliza para detectar la pérdida de paquetes y restaurar la secuencia de paquetes enviados. El valor inicial de este número debe ser aleatorio (y por lo tanto, impredecible) para hacer que los ataques mediante texto plano o el cripto-análisis por ejemplo, sean más difíciles de realizar, incluso si la fuente no se cifra de acuerdo con los métodos propuestos en el documento RFC, ya que los paquetes pueden ser transportados a través de un traductor que si lo hace.

  Hay que resaltar que el protocolo RTP no toma ninguna acción en caso de detectar la pérdida de algún paquete. Esto se lo delega a la aplicación, para que decida qué hacer. Por ejemplo, una aplicación de video puede solicitar nuevamente el paquete si encuentra que hubo alguna pérdida. Otra aplicación puede, sin embargo, decidir modificar el parámetro de decodificación en orden de reducir la necesidad de ancho de banda. Todas estas decisiones requieren de inteligencia adicional para ser resueltas, por lo que el protocolo la delega a las aplicaciones.

- **Timestamp** : *Marca temporal* - 32 bits.

  La marca de tiempo refleja el instante de muestreo del primer octeto en el paquete de datos RTP. El instante de muestreo debe derivarse de un reloj que se incremente de forma monótona y linealmente en el tiempo para permitir la sincronización y los cálculos de fluctuación de fase en los paquetes.

  El valor inicial deberá de ser aleatorio. RTP no establece ninguna unidad de medida para el tiempo, ya que distintas aplicaciones requerirán diferente gra-

nularidad temporal; y esta granularidad será necesaria especificarla en el perfil RTP o en el formato de la Carga útil para la aplicación.

- **Synchronization source (SSRC)**: 32 bits.

  Este campo identifica la fuente de sincronización en la misma sesión establecida por RTP. Es elegida al azar para evitar que dos fuentes en la misma sesión RTP tengan el mismo identificador SSRC. Este indica dónde fueron combinados los datos, o si sólo hay una fuente. La fuente emisora puede estar en el mismo nodo o en diferentes nodos, como por ejemplo, durante una conferencia.

- **Contribuiting Source (CSRC)** : *Lista de Fuentes colaboradoras*

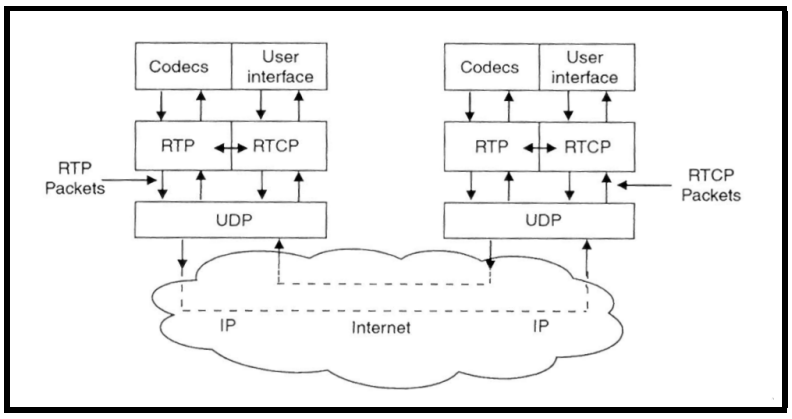
  Puede tener entre 0 y 15 items, de 32 bits cada uno. La Lista de Fuentes Colaboradoras identifica las fuentes que contribuyen en la carga útil que contiene este paquete. El número de identificación viene dado por el campo CC. En caso de que existen más de 15 fuentes colaboradoras, sólo se identifican 15.

  Es tarea de los denominados *mezcladores* el ir insertando los identificadores CSRC, utilizando los SSRC de las fuentes colaboradoras.

  Un mezclador puede utilizarse por ejemplo, en una conferencia para combinar datos que son recibidos desde distintas fuentes y enviarlos, como un único flujo, para reducir el ancho de banda requerido.

2.3. Protocolo de Control para RTP

RTCP es un protocolo de control diseñado para trabajar conjuntamente con RTP y se encuentra especificado en el documento RFC 3550. La relación entre estos dos protocolos se puede ver en la siguiente imagen:



Como se dijo, RTCP es un protocolo para el control de RTP. Este provee control ”fuera de banda” para flujos multimedia. Cabe destacar que RTCP no transmite datos por si mismos, pero colabora con RTP en empaquetar y enviar datos multimedia (audio o video). Transfiere de forma periódica datos de control entre los participantes de una

sesión multimedia; teniendo como tarea el proveer de información acerca de la calidad del servicio entregado por RTP y el de sincronizar el audio y el video.

En una sesión de RTP, los participantes de forma periódica, envían paquetes RTCP a todos los miembros que se encuentren en la misma sesión usando IP *multicasting*. Esto lo hace mediante la recolección de datos, como ser: número de bites transferidos, número de paquetes transmitidos, la cantidad de paquetes perdidos, la latencia de la red, etc. Muchas aplicaciones utilizan la información suministrada por RPCT para mejorar la calidad del servicio que brindan. Cabe resaltar que RTCP por sí solo no provee datos de encriptación o de autenticación. Para esto se puede utilizar SRTP (Protocolo seguro de Transporte en Tiempo Real).

La recolección de datos y las eventuales estadísticas que realiza RTCP, son útiles para aquellas aplicaciones adaptativas, que utilizan esta información para enviar los datos en alta o baja calidad dependiendo de la congestión en la que se encuentre la red. Estas incrementarán el ratio de compresión cuando haya poco ancho de banda habilitado y lo reducirá en caso contrario. También esta información es útil a fin de realizar diagnósticos para encontrar problemas que puedan ir surgiendo a medida que transcurre la transmisión.

RTCP provee una forma para realizar un correlato y una sicronización para distintos flujos que provengan del mismo emisor. Cuando ocurre una colisión de tipo SSRC, es necesario modificar el valor de SSRC para un flujo determinado y esto es realizado por el protocolo de Control de RTP.

Para aquellas aplicaciones que implican tener flujos multimedia separados, se utiliza un sistema de reloj en común para realizar la sincronización. El sistema que inicia la sesión provee esta funcionalidad y los mensajes provistos por RTCP activan a todos los demás sistemas para utilizar el mismo reloj del anfitrión.

También es utilizado para proporcionar información sobre el número de miembros de una sesión.

La documentación provista por RFC, define cinto tipo de paquetes para RTCP para manejar la información de control. Estos cinco tipos son los siguientes:

- **Receiver Report (RR)**: *Informe del Receptor*

  Este informe se genera a partir de aquellos participantes que no están siendo parte de los transmisores activos de una comunicación. Este contiene informe, comentarios sobre la calidad de la recepción de los datos que están siendo transmitidos, incluyendo la mayor cantidad de paquetes recibidos, el número de paquetes perdidos, congestión entre llegadas y marcas de tiempo para poder calcular el retardo que existe entre el viaje de ida y vuelta entre emisor y receptor.

- **Sender Report (SR)**: *Informe del Remitente*

  Este tipo de informe es generado por aquellos participantes que están siendo parte de los transmisores activos en la comunicación. Además de los datos provistos por los Informes del Receptor, este contiene además una sección sobre la información enviada, reportes sobre la sincronización intermedia, como también sobre los contadores generados en la transmisión y el número de bytes enviados.

- **Source Description Items (RR)**: *Items para la descripción de la fuente*

  Este paquete contiene información sobre la fuente emisora. En los paquetes de datos de tipo RTP, la fuente es identificada mediante un número aleatorio de 32 bits. Estos identificadores no son convenientes para la lectura humana, por lo que RTCP provee una descripción denominada *Nombres canónicos* como identificadores globales y únicos para cada participante de la sesión en curso. Esta información se provee en forma de texto plano y puede incluir datos tales como: nombres de usuario, números telefónicos, direcciones de correo electrónico, etc.

- **BYE**:

  Indica el fin de la participación en la sesión RTP en curso.

- **Application specific functions (APP)**: *Funciones específicas de aplicación*

  Este tipo de paquete está destinado al uso experimental para las nuevas aplicaciones y las nuevas funcionalidades que estén en desarrollo.

Los paquetes RTCP son enviados periódicamente a través de los participantes de la comunicación. Cuando el número de participantes se incrementa, es necesario encontrar un equilibrio entre obtener información de control actualizada y limitar el tráfico de control RTP. A fin de ser escalable a grandes grupos multicast, RTCP debe prevenir el control de tráfico en situaciones donde los recursos de la red pueden ser escasos. Este limita el control de tráfico en general al 5 % de todo el ancho de banda, siendo posible mediante el ajuste permanente que se realiza en sus paquetes, según la velocidad promedio de entre todos los participantes de una sesión.

3. Protocolo RTSP

RTSP es un protocolo de flujo de datos en tiempo real no orientado a conexión, que se utiliza para definir cómo se hará el envío de información entre el cliente y el servidor. Este protocolo trabaja a nivel de aplicación y controla que la entrega de datos se realice correctamente, pues el tipo de contenido con el que se trabaja normalmente al hacer streaming es muy sensible a la sincronía temporal (o a la falta de ella). Así pues se podría considerar que el RTSP actúa como si de una especie de mando a distancia para servidores multimedia se tratase.

RTSP define diferentes tipos de conexión y diferentes conjuntos de requisitos, para intentar conseguir siempre un envío de flujo de datos a través de redes IP lo más eficiente posible. Además establece y controla uno o más flujos sincronizados de datos como audio y video. Para ello, se definió el uso de sesiones, mediante un identificador único.

RTSP es independiente del protocolo de transporte y puede funcionar tanto sobre UDP, RDP o TCP. Durante una sesión, un cliente puede abrir y cerrar conexiones fiables de transporte con el servidor mediante peticiones RTSP. Los flujos controlados por RTSP pueden usar RTP como se ha mencionado anteriormente, pero el modo de operación de RTSP es independiente del mecanismo de transporte usado para transmitir el continuo flujo de datos. Sin embargo, en la mayoría de casos se utiliza TCP para el control del reproductor y UDP para la transmisión de datos con RTP.

Comencemos por ver cómo se definen las comunicaciones en el documento de RFC, bajo RTSP.

3.1. Puertos bajo RTSP

Como se comentó anteriormente, los flujos enviados bajo este protocolo, se dice que son ”fuera de banda” porque el flujo de datos multimedia en el envío, viajan por un puerto y bajo un protocolo distinto (RTP sobre UDP), y en cambio, la información de la sesión, así como los mensajes de petición y respuesta son transportados por RTSP sobre TCP. Los puertos utilizados a menudo pueden estar descritos en la URL, o bien, se utilizan los asignados por defecto:

- Puerto nº 554: RTSP sobre UDP o TCP
- Puerto nº 8554: RTSP sobre UDP o TCP (alternativos)
- Puerto nº 322: RTSPS (TCP/TLS)

Veamos ahora cómo son los comándos que RTSP tiene disponibles para interactuar con el contenido multimedia.

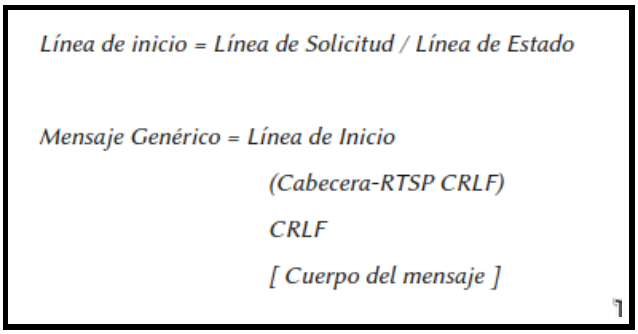
3.2. Métodos RTSP

Los métodos proporcionados por el protocolo para los mensajes de tipo *request* son los siguientes:

- **DESCRIBE**: El cliente recupera del servidor la descripción de una presentación u objeto que se encuentra identificado en el URL request.
- **GET_PARAMETER**: Recupera el valor de un parámetro de una presentacion o un flujo especificado en la URL.
- **OPTIONS**: No todos los métodos son obligatorios de implementar. Esta opción, permite que el cliente o el servidor informen a la contraparte sobre qué métodos soportan.
- **RECORD**: El cliente inicia la grabación de un rango acordado en la descripción de la presentación.
- **PAUSE**: El cliente detiene momentáneamente el flujo RTP sin liberar recursos.
- **PLAY**: El cliente pide al servidor comenzar o continuar con el flujo RTP
- **PLAY_NOTIFY**: Es emitido por el servidor para informar al cliente de algún evento de sincronización, sobre un flujo que este ocurriendo en tiempo real en una sesión.
- **REDIRECT**: El servidor indica que el cliente debe conectarse en otra localización

- **SETUP**: El cliente le pide al servidor asignar recursos para un flujo y una sesión RTSP.
- **SET_PARAMETER**: Configura el valor de un parámetro de una presentación o un flujo especificado en la URL.
- **TEARDOWN**: El cliente le pide al servidor que detenga el flujo RTP y libere los recursos asociados a la sesión.

Todo ello se hace mediante mensajes de tipo *request* (Mensaje de Solicitud) y *response* (Mensaje de Respuesta). En la imagen a continuación, se visualiza la sintáxis utilizada para los mensajes bajo RTSP.



3.3. Sintaxis de mensajes tipo *Request* y *Response*

3.3.1. Sintaxis del tipo Solicitud

Un mensaje de tipo *request* o de Solicitud, tiene el siguiente formato:

- **Línea de Solicitud**: contiene el método, el identificador del recurso y la versión del protocolo.
- **Cabecera**: Puede estar compuesta por cero, una o más líneas. Y entre ellas puede contener: Cabecera general, Cabecera de Solicitud o Cabecera de cuerpo de mensaje. Por ejemplo: Accept-Language
- Un retorno de carro y salto de línea (CRLF) que indica el final de la sección
- **Cuerpo de Mensaje**: (Opcional) que consiste en una o más líneas. La dimensión del cuerpo está especificada por el campo de cabecera de mensaje Content-Length y el tipo por el campo Content-Type.

3.3.2. Sintaxis del tipo Respuesta

Un mensaje de tipo *Response* o de Respuesta, tiene el siguiente formato:

- **Línea de estado**: Consiste en la versión del protocolo, un código numérico y una frase explicativa.

- **Cabecera de Respuesta**: Proporcionan información que no puede incluirse en la línea de estado. Por ejemplo, información sobre el servidor o sobre el archivo informado en la URL del mensaje de solicitud.
- Un retorno de carro y salto de línea *CRLF* que indica el final de la sección.
- **Cuerpo del Mensaje** (opcional): Algunos métodos RTSP para mensajes de requerimientos, tienen respuestas que pueden tener un cuerpo de mensaje. Por ejemplo GET_PARAMETER, SET_PARAMETER y respuestas con códitos del tipo *4xx* y *5xx* también pueden incluir un cuerpo de mensaje.

3.4. Funcionamiento de RTSP

En primer lugar, el cliente establece una conexión TCP con el servidor RTSP en el puerto 554, este le devolverá un archivo con metainformación que luego el cliente le proporciona al *reproductor* o *control*. Mediante los comandos OPTIONS, DESCRIBE y SETUP se configura una sesión con un ID único, por el cual identificará a todos los flujos que pertenezcan a dicha sesión.

Antes de conectarse al servidor, el cliente corrobora qué operaciones soporta, y ello lo hace mediante el método OPTIONS. El campo de cabecera *Cseq* indica un número de secuencia con el cual el mensaje de respuesta del servidor se corresponde, es decir, el valor es el mismo para cada mensaje de solicitud o respuesta.

```
C-->S OPTIONS rtsp://media.ejemplo:1234 RTSP/1.0
      CSeq: 1

S-->C RTSP/1.0 200 OK
      Server: MajorKernelPanic RTSP Server
      Cseq: 1
      Content-Length: 0
      'Public: DESCRIBE,SETUP,TEARDOWN,PLAY,PAUSE'
```

Luego de esto, el cliente pide la descripción para la transmisión. El servidor, por su lado, le proporciona informacion en el formato del protocolo SDP que sirve para describir sesiones. También podría utilizar formatos como XML o MHEG

```
C-->S DESCRIBE rtsp://media.ejemplo:1234 RTSP/1.0
      CSeq: 2

S-->C RTSP/1.0 200 OK
      Server: MajorKernelPanic RTSP Server
      Cseq: 2
      Content-Length: 467
      'Content-Type: application/sdp'
      <CRLF>
      <SDP data. . . . . >
```

Finalmente el cliente ya conoce la información necesaria para configurar el protocolo que debe utilizar y los puertos (que son los 5067 y 5068) indicados en el campo de cabecera *Transport*. El mensaje de respuesta incluye los puertos del servidor (siendo el 6023 y 6024) en el campo *Transport* y el campo session contiene el ID bajo el cual estará toda la transmisión.

```
C-->S SETUP rtsp://media.ejemplo:1234/trackID=1 RTSP/1.0
      CSeq: 3
      Transport: RTP/AVP;unicast;client_port=5067-5068

S-->C RTSP/1.0 200 OK
      Server: MajorKernelPanic RTSP Server
      Cseq: 3
      Content-Length: 0
      Transport:RTP/AVP/UDP;unicast;destination=hostCliente;
      ___ client_port=5067-5068;'server_port=6023-6024'
      'Session: 1185d20035702ca'
       Cache-Control: no-cache
```

En los siguientes tres ejemplos, se muestra cómo el cliente controla el flujo RTP con los métodos PLAY para reproducir el contenido, PAUSE para detener el contenido, y TEARDOWN para dar finalización a la sesión y posteriormente liberar recursos.

```
C-->S PLAY rtsp://media.ejemplo:1234 RTSP/1.0
      CSeq: 4
      Session: 1185d20035702ca

S-->C RTSP/1.0 200 OK
      Server: MajorKernelPanic RTSP Server
      Cseq: 4
      Content-Length: 0
      RTP-Info: url=rtsp://media.ejemplo:1234/trackID=1;seq=0
      Session: 1185d20035702ca
```

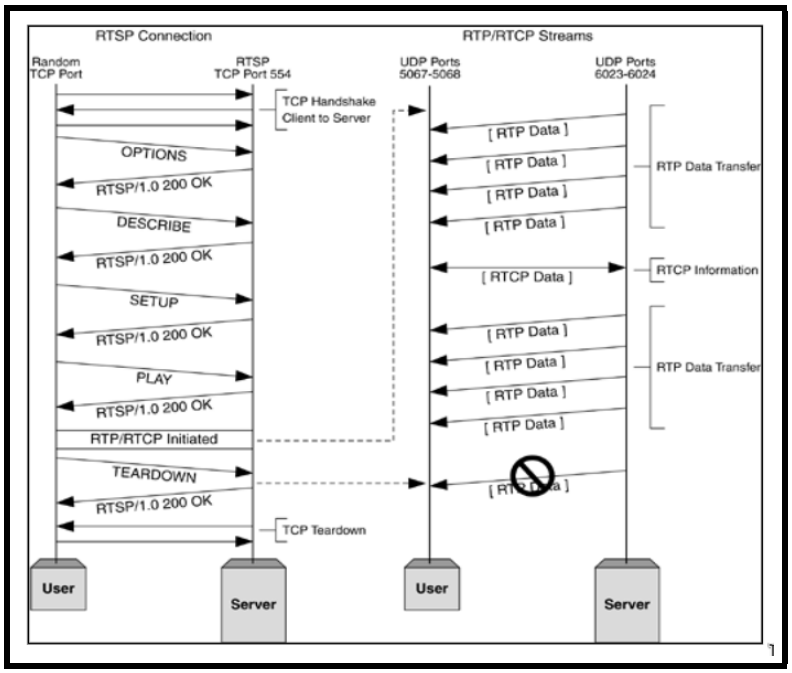
```
C-->S PAUSE rtsp://media.ejemplo:1234 RTSP/1.0
       CSeq: 5
       Session: 1185d20035702ca

S-->C RTSP/1.0 200 OK
       Server: MajorKernelPanic RTSP Server
       Cseq: 5
       Content-Length: 0
```

```
C-->S TEARDOWN rtsp://media.ejemplo:1234 RTSP/1.0
       CSeq: 6
       Session: 1185d20035702ca

S-->C RTSP/1.0 200 OK
       Cseq: 6
```

En la figura que se muestra a continuación, se puede visualizar un resumen de lo expuesto anteriormente. Se puede ver cómo RTSP conforma un protocolo ”fuera de borda” ya que establece dos conexiones en paralelo: una donde transmite los mensajes de control y otra conexión destinada al flujo de datos multimedia.



La imagen muestra cómo es un archivo RTSP de descripción. Este documento se puede definir como un XML, el cual es muy parecido a un archivo de tipo HTML. La etiqueta *group* incluye 2 pistas de audio y un archivo de video.

También tiene la propiedad de idioma establecida en Inglés (en=English) y un pedido de sincronización entre la pista de audio y el video (determinada por la palabra *lipsync*).

La etiqueta *switch* indica que se puede seleccionar entre dos pistas de audio de distinta calidad (notar las palabras *hifi* y *lofi*) y los esquemas de codificación están dados por los parámetros *e* y *pt*.

Para localizar el archivo en el servidor RTSP se indica la dirección en el parámetro *srs* (La URL puede incluir el número de puerto).

```
<title>Twister/title>
<session>
  <group language=en lipsync>
    <switch>
      <track type=audio
             e="PCMU/8000/1"
             src= "rtsp://audio.example.com/twister/
                   audio.en/lofi">
      <track type=audio
             e="DVI4/16000/2" pt="90 DVI4/8000/1"
             scr="rtsp://audio.example.com/twister/
                   audio.en/hifi">
    </switch>
    <track type="video/jpeg"
           scr="rtsp://video.example.com/twister/video">
  </group>
</session>
```

3.5. Campos generales en las cabeceras RTSP

Los siguientes campos pueden incluirse tanto en solicitudes como en las respuestas de los mensajes RTSP

- **Accept-Ranges**: Indica el formato de marca de tiempo que el recurso soporta (NPT,SMPTE,AbsoluteTime)
- **Cache-Control**: Se usa para definir mecanismos de cacheo a lo largo de la cadena de solicitudes/respuestas
- **Connection**: Se incluye para especificar que se desea luego de que el mensaje sea recibido. Por ejemplo Connection:close
- **CSeq**: Indica número de secuencia para un par solicitud/respuesta.
- **Date**: Indica la fecha en que el mensaje fue creado.
- **Media-Properties**: Indica qué propiedades acepta el recurso. Por ejemplo: comenzar el clip desde un tiempo aleatorio o cambiar velocidad.
- **Media-Range**: Utilizado en las respuestas de los métodos SETUP, PLAY y PAUSE para indicar el rango de tiempo del recurso o bien informar en qué parte del rango está el recurso al momento en que se recibe la solicitud
- **Pipelined-Requests**: Se utiliza en SETUP requests para encadenar varias solicitudes (paralelo) de configuración, sin necesidad de esperar por el ID de la sesión para responderlas, pues el campo aloja un ID propio que es alcanzado por la conexión persistente.

- **Proxy-Supported**: Contiene una lista de etiquetas de configuración aceptadas por un proxy
- **Range**: Rango de tiempo para realizar una acción o bien definir duración. Por ejemplo: Range smpte 00:00:00 - 00:56:14
- **RTP-Info**: Utilizado solo cuando el stream es vía RTP. Contiene parámetros como url, RTP timestamp(para mapear con marca de tiempo RTSP)y número de secuencia(seq)
- **Scale**: Indica la velocidad en que el recurso es reproducido. Por ejemplo Scale = 1, indica velocidad normal ; Scale=0.5 indica la mitad de la velocidad normal. También admite valores negativos.
- **Seek-Style**: Sirve para indicar al cliente que envía un PLAY request desde una posición aleatoria del recurso, sobre que punto elegirá el servidor para comenzar el flujo. Por ejemplo el usuario selecciona, a través de la interfaz del reproductor, comenzar el recurso en el segundo 45, pero el servidor comienza el video en el segundo 30.
- **Server**: Indica el software utilizado para crear o manejar una solicitud.
- **Session**: Identifica toda la cadena de solicitudes/respuestas. El servidor crea un ID y lo devuelve en un SETUP response. Todas las siguientes solicitudes/respuestas DEBEN incluir este campo.
- **Speed**: Se usa para indicar al server un rango de velocidad de entrega de datos(bitrate) para transmitir por la red.
- **Supported**: Contiene una lista de etiquetas de configuración soportadas por el cliente o el servidor
- **Timestamp**: Indica en qué momento exacto el agente-usuario envía una solicitud.
- **Transport**: Indica qué protocolo de transporte se utilizará y sus parámetros tales como ip destino, multicast o unicast, ttl o puerto destino.
- **User-Agent**: Contiene información acerca del agente que origina una solicitud o el que produce una respuesta; su uso tiene fines estadísticos. Ejemplo: *User-Agent: PhonyClient/1.2 Via*, este campo debe ser utilizado por proxies para indicar el protocolo intermediario y receptor, en mensajes de solicitud y respuesta. Este campo está pensado para dar seguimiento a mensajes re-enviados para evitar bucles.

3.6. Códigos de estado para mensajes de respuesta

El código de estado está formado por 3 dígitos que entiende el agente-usuario, seguido de una frase de descripción para el entendimiento del usuario humano. El primer dígito del código de estado define la clase de respuesta, mientras que los últimos dos no tienen ninguna categorización. Existen cinco valores para el primer dígito:

- **1xx**: Informativo. Solicitud recibida, continuando proceso.
- **2xx**: Éxito; la acción fué satisfactoriamente entendida y aceptada
- **3rr**: Redirección. Es necesario tomar medidas adicionales para completar la solicitud
- **4xx**: Error en el cliente. La solicitud tiene mala sintaxis o no puede ser cumplida
- **5xx**: Error en el servidor. El servidor falló para cumplir una solicitud aparentemente correcta.

4. Laboratorio

4.1. Descripción

A continuación se detalla parte del laboratorio realizado para capturar paquetes de tipo RTSP mediante el software Wireshark. El mismo consistió en crear un servidor en un Sistema Operativo tipo Ubuntu, el cual se levantó en una máquina virtual mediante el software Virtualbox, sobre Windows 10 como sistema anfitrión. A través del servidor, el cual utiliza los protocolos descriptos en el presente trabajo, se alojó un archivo de tipo MP3 el cual pudo ser compartido entre varios dispositivos que pudieron reproducirlo a demanda bajo el reproductor VLC.

Como último paso, se configuró una máquina, con la cual capturar paquetes de tipo RTSP mediante el uso de Wireshark a medida que reproducía el archivo de audio.

4.2. Objetivos realizados

1. Crear un servidor de streaming en tiempo real en una máquina virtual con Ubuntu como huésped, corriendo sobre Windows 10 como anfitrión.
2. Reproducir en Ubuntu, el contenido de audio con el reproductor VLC, conectándose al servidor live555 RTP.
3. Reproducir desde el SO anfitrión (Windows 10) el contenido multimedia.
4. Acceder, mediante una misma conexión LAN al servidor live555 bajo distintos dispositivos, como por ejemplo: Pc, Notebook, teléfonos móviles.
5. Reproducir en múltiples host, los cuales no se encuetran bajo una misma red, el contenido multimedia.
6. Capturar paquetes de tipo RTP, RTCP, y RTSP del lado del cliente mediante Wireshark.

4.2.1. Creación del servidor de streaming

Bajo el Sistema Operativo Windows 10 como anfitrión, se instaló el software para virtualización denominado Virtualbox.

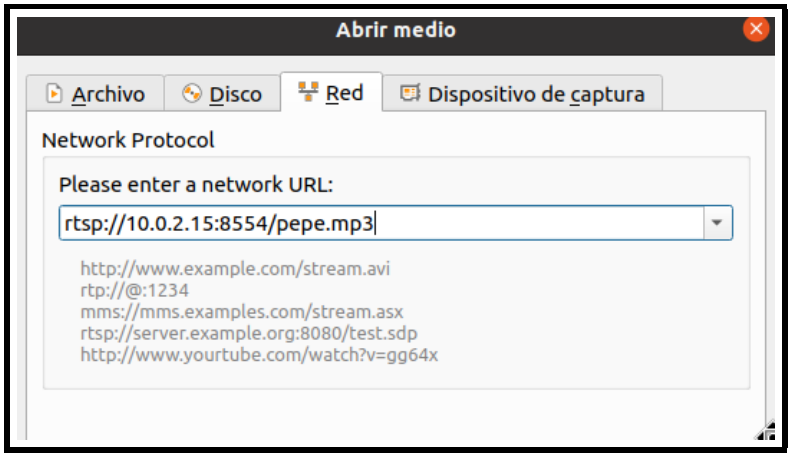
Se procedió a instalar el Sistema Operativo Ubuntu, como sistema huésped, en la version 20.04 LTS, de 64 bytes. El servidor de reproducción en tiempo real elegido fue Live555 Media Server, que utiliza los protocolos RTP/RTCP/RTSP.

Una vez instalado el servidor en el huésped Ubuntu, se ingresa un archivo de tipo MP3, denominado "pepe.mp3", el cual será el contenido utilizado para poder compartir con los demás dispostivos.

4.2.2. Configuración de VLC

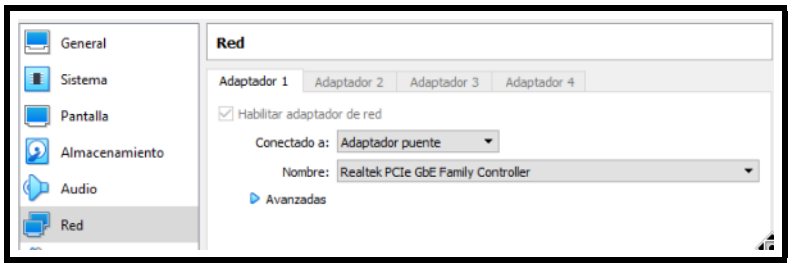
En el sistema huésped Ubuntu, se instala también el reproductor VLC desde los repositorios oficiales.

Una vez instalado, se procede a ingresar la URL desde la cual se reproducirá el archivo MP3.



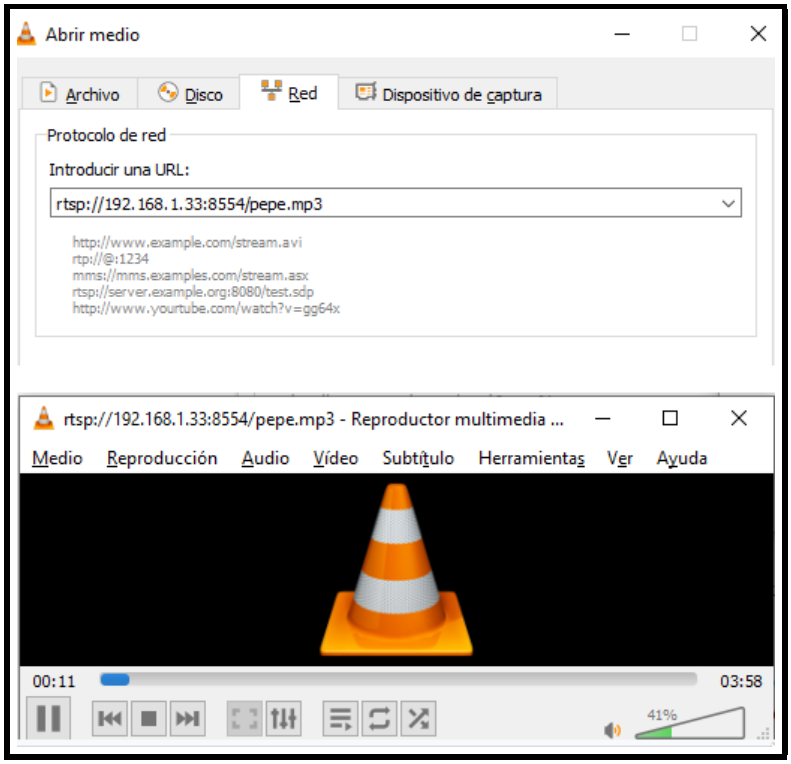
4.2.3. Conexión al servidor mediante Windows

Se prueba acceder al contenido mediante el sistema anfitrión Windows. Para que este pueda reproducirlo de forma satisfactoria, es necesario cambiar el adaptador que utiliza Virtualbox para que tome la dirección IP de la placa de red virtualizada. El mismo se conecta utilizando el modo puente.



Desde el lado del huésped, se abre el puerto nº 8554 para que sea accedido desde el anfitrión. Una vez realizado esto, la conexión se establecerá mediante telnet.

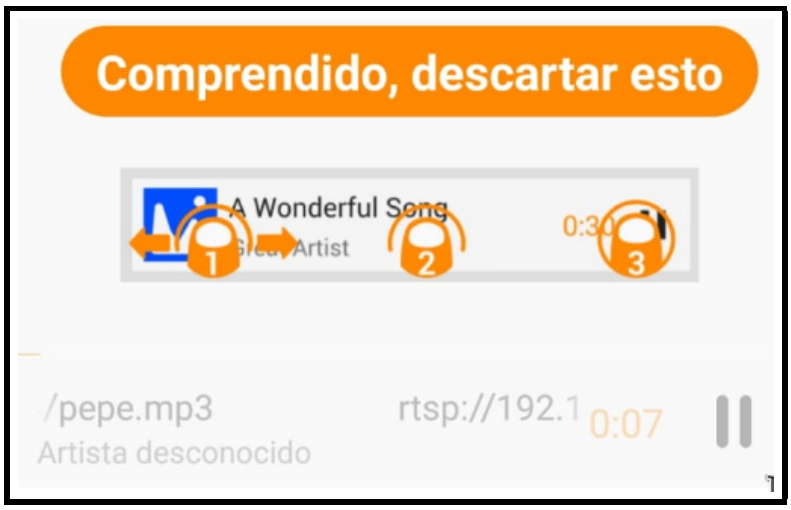
Una vez habilitado este puerto, puede verse a continuación cómo es posible reproducir el contenido del archivo MP3, suministrado por el servidor Live555, mediante una URL:



4.2.4. Reproducción desde distintos dispositivos

Bajo una misma red LAN hogareña, se conectan tres dispositivos distintos, a saber: una Notebook, un teléfono móvil y una tablet con sistema operativo Android. A todos se les proporciona el reproductor VLC para reproducir y se puede visualizar la correcta reproducción del archivo multimedia, de nombre ”pepe.mp3”.

A continuación, se puede ver la reproducción de la canción, mediante el teléfono móvil.



4.2.5. Reproducción desde distintos host en otras redes

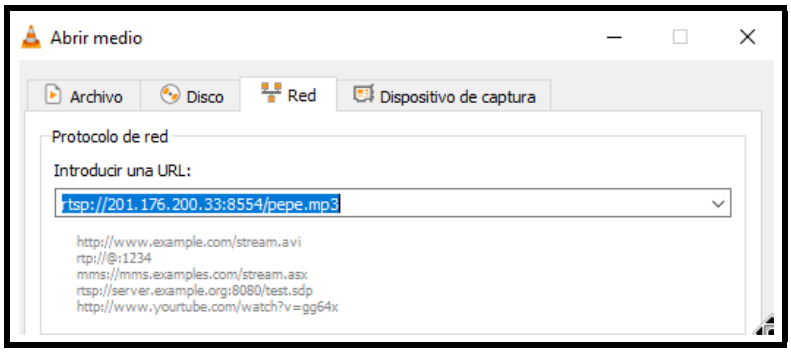
Se procedió a compartir el contenido del servidor entre los distintos integrantes del presente trabajo, los cuales nos encontrábamos bajo redes hogareñas disímiles. Para ello fue necesario que cada uno de los integrantes pudiesen tener el mismo reproductor VLC instalado en sus dispositivos.

Para que cada dispositivo pudiése obtener el mismo contenido, fue necesario configurar en el modem-router del ISP (Movistar) los puertos que utiliza el servidor y sus servicio, para que de esta forma pudiese atender las solicitudes entrantes desde internet. Se necesitó tener un IP pública para brindar a los distintos host la ruta correspondiente del servidor Live555.

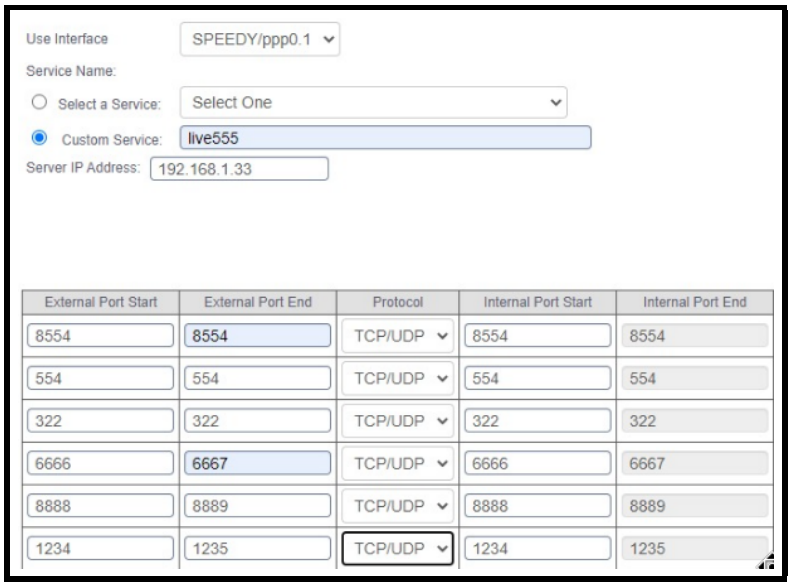
Como se sabe, Movistar modifica esta dirección cada vez que se reinicia el modem o se cierra la sesión PPOE.

A continuación se muestran las modificaciones en la configuración del modem-router para que las solicitudes que ingresen a el, sean redirigidas a la IP del servidor (cuya dirección es 192.168.1.33) en los puertos especificados en la imagen, dependiendo el tipo de archivo solicitado, vídeo,audio, y formatos se utilizarán los puertos correspondientes.

Se modifica la dirección IP del servidor:

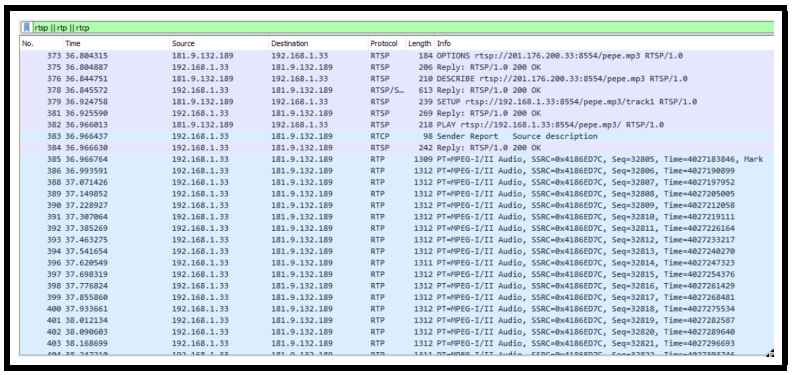


Se configuran los puertos del módem, de la siguiente manera:

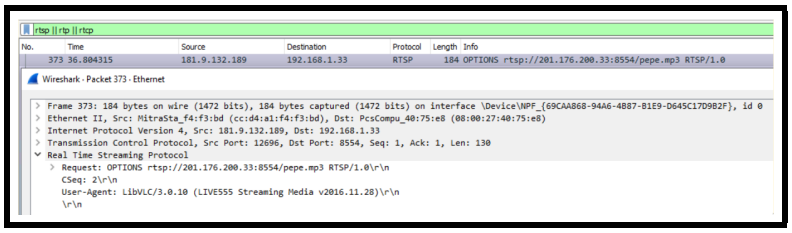


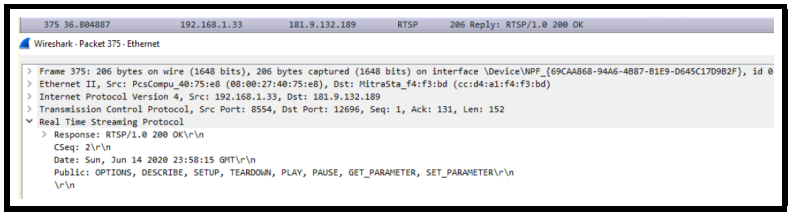
4.2.6. Capturas de paquetes en Wireshark

Dentro del software, para una mejor visualización del tráfico de la red, se aplican los filtro RTSP, RTP, RTCP. Con ello vemos la interacción de los 3 protocolos para llevar a cabo la transmisión del archivo MP3.

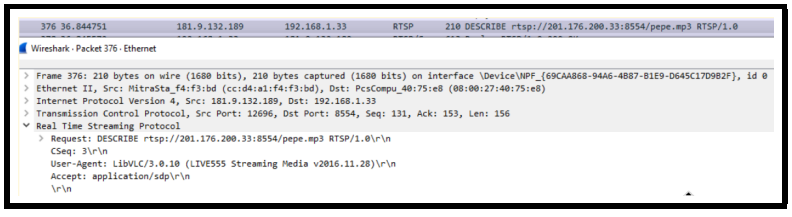


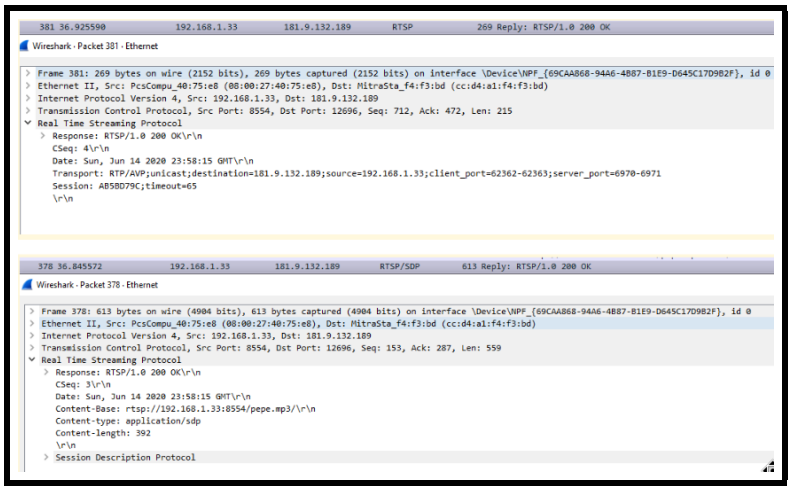
Se puede visualizar en las siguientes imágenes cómo se captura el tráfico entre el cliente enviándo una petición y el servidor contestando con un OK a la conexión solicitada.



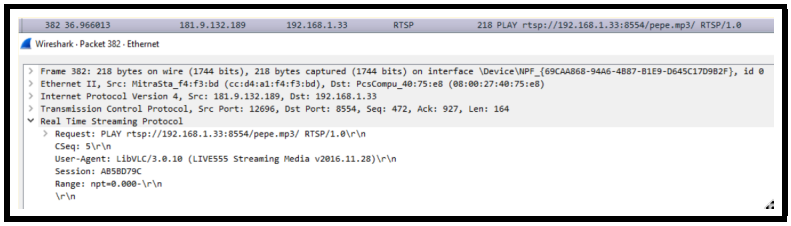


A continuación se establece el tipo de protocolo a utilizar (RTSP) en la transmisión y el archivo a reproducir.

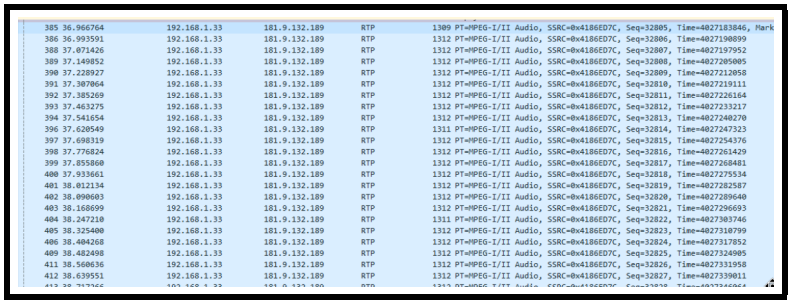




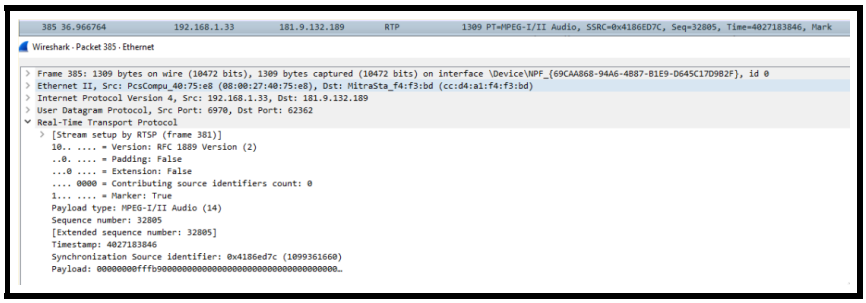
El cliente envía un mensaje de PLAY para que comience a reproducir el archivo multimedia.



A partir de ahora el servidor comienza a enviar el flujo de datos proveniente del archivo MP3 hacia el cliente. Esto lo hace mediante el protocolo RTP, como se explicó al comienzo del presente trabajo.



Se captura uno de los paquetes de tipo RTP para visualizar su contenido:



5. Conclusiones

Como conclusión del presente trabajo, se llevará a cabo una comparativa entre distintos protocolos de Tiempo Real y el conjunto de protocolos perteneciente a RTSP.

5.1. Tipos de Streaming

Presentamos en forma de cuadro expositivo, los distintos tipos de protocolos para streaming que se abordarán, teniendo como objetivo dar un paneo general sobre las particularidades de cada uno.

	Streaming				
	Streaming Tradicional		Streaming Alternativo		
			Progressive Download	HTTP Pseudostreaming	Dynamic Adaptive Streaming over HTTP
Protocolo de Aplicación	RTP-RTSP-RTCP		HTTP		
Protocolos de Transporte	UDP	TCP	TCP		
Protocolo de Red	IP				
Soporte Multicast	SI	No			
Soporte Unicast	Si				
Ante una perdidda de un paquete	El nivel de aplicación determina que hara con el paquete perdido	Retransmisión del mismo			
Capacidad de Multiplexación de varios Stream	SI		NO		
Problemas de NATEO y FIREWALL	Si	NO	NO		
Quality Service	SI		NO		
Permancia en cache	No		Si	SI	Solo en fragmentos de tamño reducido
Soportado por Browsers	No		SI		
	Necesario para la Reproduccción				
	Reproductor		Browser+Plugin / Reproductor		
Control sobre la transmisión	Si		No	SI	SI
Necesidad de Servidores	Servidores de Streaming		Servidores Web		
Inmediates	Se reproduce conforme llega				
	Solo usa el ancho de banda que necesita		Utilizan máximo ancho de banda para descarga		
Capacidad de Adaptación al ancho de banda	Si		NO	No	SI
Opción de modificar algortimo de compresión en curso	Si		NO		

5.1.1. Streaming Tradicional

Consideramos el **Streaming Tradicional** cuando por ejemplo, un usuario solicita reproducir vía navegador y mediante un hipervínculo de una página, algún archivo de audio o video. Como se realiza con cualquier petición para descargar un imagen o texto, el navegador establece primero una conexión TCP con el servidor que hospeda la información del link. A continuación, envía una solicitud del contenido del archivo especificado en el hipervínculo mediante un mensaje de petición GET. El servidor responde enviando el contenido del archivo en un mensaje de respuesta. Al recibirlo, el browser decide (a partir del campo *ContentType* de la cabecera del mensaje) invocar el reproductor de video y, al mismo tiempo, le pasa el contenido del archivo comprimido. El reproductor multimedia procede a descomprimirlo y envía el flujo de bytes resultante a la tarjeta de sonido. Como se puede apreciar, la desventaja de este método es que el navegador debe recibir el contenido del archivo completo y esto puede producir un retardo inaceptable si el tamaño del archivo es considerablemente grande.

5.2. Streaming Alternativo

Designamos **Streaming Alternativo** para distinguirlo del Steaming Tradicional, es decir, del que es proporcionado por un servidor web. Más explícitamente, es el caso donde el servidor de streaming no se encuentra separado de servidor web. En este tipo de streaming todos los pedidos se gestionan por medio de HTTP; y a diferencia del mencionado Streaming Tradicional, en este no se emplea un determinado protocolo para controlar el streaming y otro distinto para el envío de datos.

Dentro del streaming alternativo podemos encontrar distintas opciones, pero nosotros por cuestioens de forma, los reduciremos a tres tipos: **Descarga progresiva**, **HTTP Pseudostreaming** y **DASH**.

5.2.1. Descarga progresiva

Este tipo de descarga se presenta, cuando el contenido del archivo es enviado directamente al reproductor en lugar de hacerlo a través del navegador. De esta forma el reproductor procede a obtener su contenido de manera normal mediante HTTP/TCP. Al recibir el contenido del fichero, el reproductor simplemente dirige el flujo comprimido hacía la memoria de reproducción. Tras un retardo predefinido para permitir que la memoria se llene parcialmente (diez segundos en el caso del audio) comienza a leer el flujo desde la memoria y, tras descomprimirlo, envía el flujo resultante hacía la tarjeta de audio o de video. Se realiza una descarga progresiva de la información, de manera que cuando se empiece a disponer de ella se pueda comenzar con la reproducción.

La descarga se realiza utilizando el máximo ancho de banda que disponen Cliente y Servidor, y no hay ningún control para evitar cortes en la reproducción: el medio se va almacenando en disco conforme se descarga, pero si el ancho de banda es más reducido que el necesario para la reproducción, el flujo se reproduce ”a saltos” ya que se reproduce el contenido conforme se obtiene la información.

- **Ventajas**: Fácil de configurar.

- **Desventajas**: El video entero es descargado a menos que el usuario cierre el navegador. El video se descarga completamente por lo que el contenido no está protegido. El usuario no puede reproducir el video desde cualquier momento. Solo podrá visualizar o reproducir aquella parte que se encuentre ya descargada.

5.2.2. HTTP Pseudo-Streaming

Este tipo de streaming está basado en el anterior. Lo que pretende es simular el comportamiento del streaming bajo demanda, agregando la posibilidad de adelantar o retroceder la reproducción. Es decir que las partes que se "saltean" no se descargan permitiendo así, reducir el ancho de banda que en determinas situaciones se pierde. Pseudo-streaming requiere adaptaciones tanto para el lado del cliente como para el lado del servidor.

Para el lado del servidor, existen plugins disponibles para Apache, como ser *litghttpd*. Mientras que del lado del cliente, es necesario reproductores adaptados o modificados, que permitan resincronizar el video, leer metainformación, etc.

- **Ventajas**: Posibilidad de interactuar con el stream. Mejor utilización del ancho de banda.
- **Desventajas**: Necesita implementaciones tanto por parte del cliente como del servidor. El video no está protegido, es posible encontrarlo en la cache del browser.

5.2.3. DASH

Dynamic Adaptive Streaming over HTTP (DASH) tiene como idea central el de dividir el video en pequeñas partes y enviarlo por HTTP. Luego esas partes son combinadas en el lado del cliente y reproducidas. Este método soporta video bajo demanda y lo que se denomina como *live streaming*, permitiendo entregar el flujo teniendo en cuenta, cuán saturada esté la red. Existen numerosas implementaciones de este método con diferentes nombres, todas basadas en la idea antes mencionada: HTTP Live Streaming (Apple), Smooth Streaming (Microsoft) HTTP Dynamic Streaming (Adobe), Adaptive Bitrate (Octoshape).

- **Ventajas**: Sobre HTTP ofrece una solución muy robusta. Mejoras en cuanto a la protección del contenido dado que el video es dividido en pequeñas partes.
- **Desventajas**: Las diferentes implementaciones no son del todo compatible. Requiere un desarrollo adicional del lado del cliente.

6. Herramientas

6.1. Materiales utilizados para el presente trabajo

Para la resolución del presente Laboratorio se utilizaron las siguientes herramientas:

1. Software de virtualización VirtualBox.
2. Software para el análisis de protocolos WireShark.
3. Reproductor de audio y video VLC.
4. Servidor de Medios Liver555 Media.
5. Arch Linux V5.1.11
6. Para la composición del presente Informe se utilizó el paquete *texlive-latexextra 2018.50031-1*

Bibliografía

[1] Arjan Durresi y Jain Raj. "RTP, RTCP and RTSP - Internet Protocols for Real Time Multimedia Communication". En: *The Industrial Information Technology Handbook* 5 (2005), pág. 28.

[2] James F. Kurose y Keith W. Ross. *Computer Networking: a Top Down Approach*. Pearson, 2017.

[3] Colin Perkins. *RTP: Audio and Video for the Internet*. Addison Wesley, 2003.

[4] Andrew S. Tanenbaum. *Computers Networks*. Pearson, 2003.