Práctica 6 - Sesión SIP *peer-to-peer*

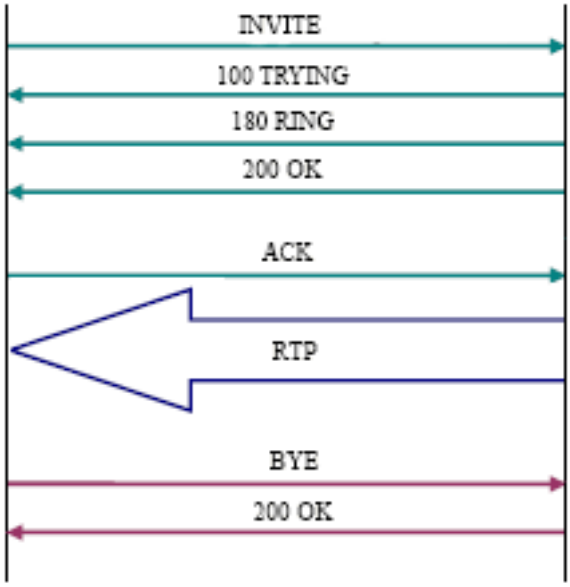
Protocolos para la Transmisión de Audio y Vídeo en Internet

Versión 11.0.1 – 9.12.2020

Nota: Esta práctica se puede entregar para su evaluación como parte de la nota de prácticas, pudiendo obtener el estudiante hasta 0,7 puntos. Para las instrucciones de entrega, mira al final del documento.

Introducción

Esta práctica tiene como objetivo implementar dos programas Python, de manera que se pueda realizar una sesión SIP como la que se muestra en la siguiente figura[1]:



En esta sesión:

- No habrá ni servidor de registro ni *proxy*. La comunicación será *peer-to-peer* (directa entre dos UAs[2]).
- El UA de la izquierda constará exclusivamente de la parte cliente. Esta parte será la que inicie la sesión con un `INVITE` y la cierre con un `BYE`.
- El UA de la derecha constará exclusivamente de la parte servidora. Esta parte será la que envíe el audio vía RTP al cliente.

[1] En la figura las respuestas SIP están en mayúsculas para mejorar su lectura; en tu implementación deberían seguir el estándar, en el que sólo la primera letra es mayúscula

[2] UA: User Agent. Un UA se compone de un cliente y un servidor, aunque en esta práctica -por simplificación- cada UA tendrá sólo una de las dos partes.

Objetivos de la práctica

- Implementación de una conversación de audio iniciada con SIP.

Conocimientos previos necesarios

- Nociones de SIP (las de clase de teoría)
- Funcionamiento de `wireshark`.

Tiempo estimado: 10 horas

Creación de repositorio para la práctica

Con el navegador, dirígete al repositorio `ptavi-p6` en la cuenta de *ptavi* en GitLab[3] y realiza un `fork`, de manera que consigas tener una copia del repositorio en tu cuenta de GitLab. Clona el repositorio que acabas de crear a local para poder editar los archivos. Trabaja a partir de ahora en ese repositorio, sincronizando los cambios que vayas realizando.

Como tarde al final de la práctica, deberás realizar un `push` para subir tus cambios a tu repositorio en GitLab. En esta práctica, al contrario que con las demás, se recomienda hacer frecuentes `commits`, pero el `push` al final.

Parte cliente

El cliente ha de ejecutarse de la siguiente manera:

```
$ python3 client.py metodo receptor@IPreceptor:puertoSIP
```

donde *metodo* será un método SIP, `receptor@IPreceptor:puertoSIP` será el login, la IP y el puerto[4] al que se dirige el mensaje.

Por ejemplo:

```
$ python3 client.py INVITE batman@193.147.73.20:5555
$ python3 client.py BYE batman@193.147.73.20:5555
```

En caso de no introducir el número de parámetros correctos o de error en los mismos, el programa debería imprimir siempre por pantalla:

```
Usage: python3 client.py method receiver@IP:SIPport
```

Peticiones SIP

El cliente debería poder enviar las siguientes peticiones SIP:

- INVITE sip:receptor@IP SIP/2.0

  Mediante este método, el UA indica que quiere iniciar una conversación con el receptor con dirección *receptor* en la máquina dada por la *IP*[5].

[3] `http://gitlab.etsit.urjc.es/ptavi/ptavi-p6`

[4] En el mundo "real", el puerto sería el puerto de SIP por defecto, el 5060, ya que sólo puede haber un UA activo por máquina - si arrancas otro UA SIP cuando hay uno ya lanzado, no te dejará. Como en la práctica final tendremos dos UAs en la misma máquina, hemos de especificar un puerto para que no se "pisen".

[5] La IP puede ser 127.0.0.1.

- ACK sip:receptor@IP SIP/2.0

  Método de asentimiento. No se pasará al programa `client.py` como parámetro desde la `shell`, ya que se enviará de manera automática una vez se hayan recibido las respuestas `100 Trying`, `180 Ring` y `200 OK` de la parte servidora.

- BYE sip:receptor@IP SIP/2.0

  Mediante este método se indica que queremos terminar una conversación con el receptor con dirección *receptor* en la máquina dada por la IP. Se deberá enviar una vez haya acabado de recibir *streaming* de audio (vía RTP) enviado desde el servidor.

Parte servidora

El servidor ha de ejecutarse de la siguiente manera:

```
$ python3 server.py IP puerto fichero_audio
```

Por ejemplo:

```
$ python3 server.py 127.0.0.1 5555 cancion.mp3
```

En caso de no introducir el número de parámetros correctos o de error en los mismos (se ha de comprobar si existe el fichero de audio), el programa debería imprimir:

```
Usage: python3 server.py IP port audio_file
```

En caso de no haber error al arrancar, el servidor imprimirá por pantalla:

```
Listening...
```

Códigos de respuesta

- `SIP/2.0 100 Trying`: al recibir un INVITE.
- `SIP/2.0 180 Ringing`: al recibir un INVITE.
- `SIP/2.0 200 OK`: en caso de éxito.
- `SIP/2.0 400 Bad Request`: si la petición está mal formada.
- `SIP/2.0 405 Method Not Allowed`: si se manda en la petición cualquier otro método diferente de INVITE, BYE o ACK.

Para simplificar la práctica, se enviará **en un único mensaje** (i.e., un único paquete) la respuesta `SIP/2.0 100 Trying`, `SIP/2.0 180 Ringing` y `SIP/2.0 200 OK`.

Ejercicios alternativos

Dependiendo de tu último número de tu DNI / NIE, deberás realizar los siguientes ejercicios que se describen en la Tabla 1.

Último número DNI/NIE	Ej. A	Ej. B	Ej. C	Ej. D	Ej. E	Ej. F
0	X			X		X
1	X			X	X	
2	X		X			X
3	X		X		X	
4	X			X		X
5		X	X			X
6		X	X		X	
7		X		X	X	
8		X		X		X
9		X		X	X	

Table 1: Tabla de ejercicios alternativos a realizar según el último número de DNI/NIE del estudiante.

Ejercicio A

El cuerpo del INVITE debe incluir la descripción de sesión en formato SDP (*Session Description Protocol*). Contará con los siguientes parámetros:

- v (versión, por defecto la "0"),
- o (origen e identificador de sesión: la dirección del origen y su IP),
- s (nombre de la sesión, que puede ser el que se desee),
- t (tiempo que la sesión lleva activa, en nuestro caso, siempre 0), y
- m (tipo de elemento multimedia y puerto de escucha y protocolo de transporte utilizados, en esta práctica "audio", el número de puerto pasado al programa principal como parámetro y "RTP").

Así, un ejemplo de descripción de sesión dentro de un INVITE podría ser:

```
INVITE sip:INVITE batman@193.147.73.20 SIP/2.0

v=0
o=robin@gotham.com 127.0.0.1
s=misesion
t=0
m=audio 34543 RTP
```

Nótese que en el ejemplo anterior 34543 es el puerto donde esperamos que el otro participante en la conversación nos envíe los paquetes RTP con audio. También se ha de tener en cuenta que entre las cabeceras (en este caso la línea de INVITE) y el cuerpo (la descripción de la sesión) ha de haber obligatoriamente una línea en blanco.

Ejercicio B

El cuerpo de la respuesta 200 OK debe incluir la descripción de sesión en formato SDP (*Session Description Protocol*). Contará con los siguientes parámetros:

- v (versión, por defecto la "0"),

- o (origen e identificador de sesión: la dirección del origen y su IP),
- s (nombre de la sesión, que puede ser el que se desee),
- t (tiempo que la sesión lleva activa, en nuestro caso, siempre 0), y
- m (tipo de elemento multimedia y puerto de escucha y protocolo de transporte utilizados, en esta práctica "audio", el número de puerto pasado al programa principal como parámetro y "RTP").

Así, un ejemplo de descripción de sesión dentro de un INVITE podría ser:

```
SIP/2.0 200 OK

v=0
o=batman@gotham.com 127.0.0.1
s=misesion
t=0
m=audio 67876 RTP
```

Se ha de tener en cuenta que entre las cabeceras (en este caso la línea de INVITE) y el cuerpo (la descripción de la sesión) ha de haber obligatoriamente una línea en blanco. Nótese que como el UA que inicia la conversación no envía paquetes RTP, la información en el SDP no se utilizará.

Ejercicio C

El envío RTP se realizará mediante la biblioteca `simplertp`[6]. Un ejemplo del código Python de envío de paquetes RTP sería el siguiente:

```
import simplertp

RTP_header = simplertp.RtpHeader()
RTP_header.set_header(pad_flag=0, ext_flag=0, cc=0, marker=0, ssrc=ALEAT)
audio = simplertp.RtpPayloadMp3(audio_file)
simplertp.send_rtp_packet(RTP_header, audio, ip, port)
```

Donde ALEAT será un número entero aleatorio obtenido a partir de la biblioteca `random` de Python3[7].

La IP y puerto a la que se ha de enviar el RTP vendrá dada en el SDP (Ejercicio A). En el caso de que el estudiante no tenga que realizar ese ejercicio, utilizará la IP 127.0.0.1 y el puerto 23032 como destino del envío.

Nótese que `audio_file` es uno de los parámetros que se le pasa al servidor al ejecutarse desde la línea de shell como parámetro. También ha de notarse que si sale el aviso **Warning: Connection refused. Probably there is nothing listening on the other end.** es porque no hay ningún servicio *escuchando* al otro lado; no es un error, indica que los paquetes han llegado al destino final, pero que no han sido gestionados.

[6] En el repositorio se encuentra el módulo Python `simplertp.py`

[7] https://docs.python.org/3.8/library/random.html

Ejercicio D

El envío RTP se realizará mediante la biblioteca `simplertp`[8]. Un ejemplo del código Python de envío de paquetes RTP sería el siguiente:

```
import simplertp

RTP_header = simplertp.RtpHeader()
RTP_header.set_header(version=2, marker=BIT, payload_type=14, ssrc=200002)
audio = simplertp.RtpPayloadMp3(audio_file)
simplertp.send_rtp_packet(RTP_header, audio, ip, port)
```

Donde BIT será un bit aleatorio obtenido a partir de la biblioteca `secrets` de Python3[9].

La IP y puerto a la que se ha de enviar el RTP vendrá dada en el SDP (Ejercicio A). En el caso de que el estudiante no tenga que realizar ese ejercicio, utilizará la IP 127.0.0.1 y el puerto 23032 como destino del envío.

Nótese que `audio_file` es uno de los parámetros que se le pasa al servidor al ejecutarse desde la línea de shell como parámetro. También ha de notarse que si sale el aviso **Warning: Connection refused. Probably there is nothing listening on the other end.** es porque no hay ningún servicio *escuchando* al otro lado; no es un error, indica que los paquetes han llegado al destino final, pero que no han sido gestionados.

Ejercicio E

Se ha de añadir la cabecera `Content-Type: application/sdp` a **todos** los paquetes SIP que contengan SDP.

Así, un ejemplo de descripción de sesión dentro de un INVITE podría ser:

```
INVITE sip:INVITE batman@193.147.73.20 SIP/2.0
Content-Type: application/sdp

v=0
o=robin@gotham.com 127.0.0.1
s=misesion
t=0
m=audio 34543 RTP
```

Ejercicio F

Se ha de añadir la cabecera `Content-Length`, cuyo valor será la longitud (en bytes) del cuerpo del paquete, a **todos** los paquetes SIP que contengan SDP.

Así, un ejemplo de descripción de sesión dentro de un INVITE podría ser:

```
INVITE sip:batman@193.147.73.20 SIP/2.0
Content-Length: 66

v=0
o=robin@gotham.com 127.0.0.1
```

[8] En el repositorio se encuentra el módulo Python `simplertp.py`

[9] https://docs.python.org/3/library/secrets.html

```
s=misesion
t=0
m=audio 34543 RTP
```

Captura

Una vez terminada la práctica, se pide que se realice una captura de un establecimiento de llamada, el envío RTP de audio y la finalización de la llamada.

La captura original se filtrará para que sólo incluya los paquetes SIP y los tres primeros y tres últimos paquetes RTP con audio[10].

La captura filtrada resultante se guardará en el fichero `invite.libpcap` y se subirá al repositorio.

Fecha y modo de entrega

La entrega de práctica se deberá hacer antes del miércoles 16 de diciembre de 2020 a las 23:59. Para entonces, se debe tener un repositorio `git` en GitLab con:

- 2 módulos Python y la captura realizada con `wireshark`:
  - `server.py`
  - `client.py`
  - `invite.libpcap`

Se han de tener en cuenta las siguientes consideraciones:

- Se valorará que al menos haya diez *commits* realizados en al menos dos días diferentes.
- Se valorará que la captura esté bien filtrada y sólo contenga los paquetes que se indican en el guión.
- Se valorará que el código entregado siga la guía de estilo de Python (véanse PEP8 y PEP257).
- Se valorará que los programas se invoquen correctamente y que muestren los errores correctamente, según se indica en el enunciado de la práctica.

Se puede comprobar la correcta entrega de la práctica utilizando el programa `check-p6.py`. Este programa se ejecuta desde la línea de comandos de la siguiente manera:

```
$ python3 check-p6.py login_gitlab
```

donde `login_gitlab` es tu nombre de usuario en GitLab.

[10] Los heurísticos de Wireshark puede que no identifiquen los paquetes RTP como tales, sino como paquetes UDP. La solución es la siguiente: en el menú de "Analyze", selecciona "Enabled protocols" y ahí busca por RTP para activar "rtp_udp".