Práctica: Streaming Video con RTP y RSTP
Protocolos y Tecnologías para Servicios Móviles y Multimedia

En esta práctica vamos a implementar un cliente y un servidor de streaming de video que se comunican utilizando el Protocolo de Streaming en Tiempo Real (RTSP) y envían datos utilizando el Protocolo de Transferencia en Tiempo Real (RTP). El objetivo final es implementar el protocolo RTSP en el cliente, así como la paquetización RTP en el servidor.

Tienes a tu disposición el código que implementa el protocolo RTSP en el servidor, el código para desencapsular los datos RTP en el cliente, y para mostrar el vídeo transmitido. Observa el código pero no hagas cambios en él.

Clases

Se proporcionan cuatro ficheros de clases:

Client
: Esta clase implementa el cliente y la interfaz de usuario que se utiliza para mostrar el video y controlar la reproducción mediante comandos RTSP. Más abajo se muestra el aspecto de la interfaz. *Tendrás que implementar las acciones que se realizan cuando se pulsan los botones*.

Server
: Esta clase implementa el servidor, que responde a las peticiones RTSP y transmite el vídeo al cliente. La interacción RTSP está ya implementada y el servidor invoca a las rutinas de la clase RTPpacket para empaquetar los datos de vídeo. *No es necesario modificar esta clase*.

RTPpacket
: Esta clase se utiliza para manejar los paquetes RTP. Tiene rutinas para el manejo de los paquetes recibidos en el lado del cliente, que no tendrás que modificar. *Tendrás que completar el primer constructor de esta clase para implementar el encapsulado de los datos de vídeo en RTP.* El segundo constructor es utilizado por el cliente para desencapsular los datos y no es necesario que lo modifiques.

VideoStream
: Esta clase se utiliza para leer datos de video de un fichero almacenado en disco. No tienes que modificar esta clase.

Ejecución del Código

Una vez completado el código, puede ejecutarlo de la siguiente manera.

Primero, lanza el servidor en una shell o terminal con el siguiente comando:

```
java Server server_port
```

donde `server_port` es el puerto en el que tu servidor escucha las conexiones RTSP entrantes. El puerto RTSP estándar es el 554, pero tendrás que elegir un número de puerto mayor que el 1024, a menos que lo ejecutes con privilegios de administrador.

Después, ejecuta el cliente con el comando:

```
java Client server_name server_port video_file
```

donde `server_host` es el nombre de la máquia donde se encuentra el servidor, `server_port` es el puerto donde el servidor está escuchando, y `video_file` es el nombre del fichero que vas a solicitarle a servidor (i.e., el fichero `movie.Mjpeg` proporcionado). El formato de este fichero se describe en el Apéndice.

El cliente se conecta al servidor y muestra una ventana como la que se muestra a continuación:

Puedes enviar comandos RTSP al servidor pulsando los botones. Una interacción RTSP normal es la siguiente.

1. El cliente envía un mensaje SETUP para configurar los parámetros de la sesión y de transporte
2. El cliente envía el comando PLAY para iniciar la reproducción.
3. El cliente puede enviar PAUSE si quiere parar la reproducción del video.
4. El cliente envía el comando TEARDOWN termina la sesión y cierra la conexión.

El servidor siempre responde a todos los mensajes que envía el cliente. Los códigos de respuesta son aproximadamente los mismos que en HTTP. El código 200 significa que la solicitud fue exitosa. En esta práctica no es necesario implementar ningún otro código de respuesta.

Para más información sobre RTSP, puedes consultar el RFC 2326.

Cliente

Lo primero que tienes que hacer es implementar RTSP en el lado del cliente. Para ello, tienes que completar las funciones invocadas cuando el usuario hace clic en los botones de la interfaz de usuario. Tendrás que implementar las siguientes acciones en la correspondiente función manejadora.

Cuando el cliente se inicia, abre el socket RTSP al servidor. Utiliza este socket para enviar todas las peticiones RTSP, que se detallan a continuación:

SETUP

- Crea el socket para recibir datos RTP y establece un timeout en el socket a 5 milisegundos.
- Envía el comando SETUP al servidor. Tendrás que incluir la cabecera de transporte en la que indicarás el puerto utilizado por el socket de datos RTP creado en el paso anterior.
- Lee la respuesta del servidor y obtén el identificador de la sesión incluido en ella.

PLAY

- Envía el comando PLAY. Tendrás que incluir una cabecera de sesión para indicar el identificador de sesión devuelto por el servidor como respuesta al comando SETUP. No debes poner la cabecera de transporte en esta solicitud.
- Lee la respuesta del servidor.

PAUSE

- Envía el comando PAUSE. Debes incluir la cabecera de sesión con el identificador de sesión. Tampoco debes incluir la cabecera de transporte en este tipo de solicitud.
- Lee la respuestas del servidor.

TEARDOWN

- Envía el comando TEARDOWN. Al igual que en las dos peticiones anteriores debes incluir la cabecera de sesión pero no la de transporte.
- Lee la respuesta del servidor.

Nota: *Debes incluir la cabecera CSeq en cada mensaje que envíes al servidor.* El valor de la cabecera es un número que deberás incrementar en uno por cada nueva petición enviada.

Ejemplo

A continuación se muestra un ejemplo de interacción entre el cliente y el servidor. Las peticiones del cliente están marcadas con "C:" y las respuestas del servidor con "S:". En esta práctica, tanto el cliente como el servidor son muy simples y **esperan que los campos de la cabecera estén en el orden que se ve a continuación**. Es decir, en una petición, la primera cabecera es CSeq, y la segunda es Transport (para SETUP) o Session (para todas las demás peticiones). En la respuesta, CSeq es de nuevo la primera cabecera y Session es la segunda.

```
C: SETUP movie.Mjpeg RTSP/1.0
C: CSeq: 1
C: Transport: RTP/UDP; client_port= 25000

S: RTSP/1.0 200 OK
S: CSeq: 1
S: Session: 123456
```

```
C: PLAY movie.Mjpeg RTSP/1.0
C: CSeq: 2
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 2
S: Session: 123456
```

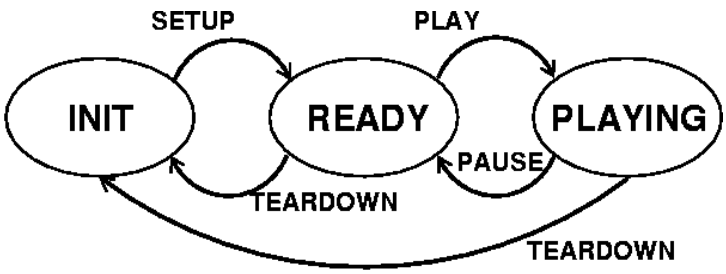
```
C: PAUSE movie.Mjpeg RTSP/1.0
C: CSeq: 3
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 3
S: Session: 123456
```

```
C: PLAY movie.Mjpeg RTSP/1.0
C: CSeq: 4
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 4
S: Session: 123456
```

```
C: TEARDOWN movie.Mjpeg RTSP/1.0
C: CSeq: 5
C: Session: 123456

S: RTSP/1.0 200 OK
S: CSeq: 5
S: Session: 123456
```

Estado del cliente

Una de las principales diferencias entre HTTP y RTSP es que, en RTSP, cada sesión tiene un estado. En esta práctica tendrás que mantener el estado del cliente actualizado. El cliente cambia de estado al recibir respuestas del servidor, según el siguiente diagrama de estados:



Servidor

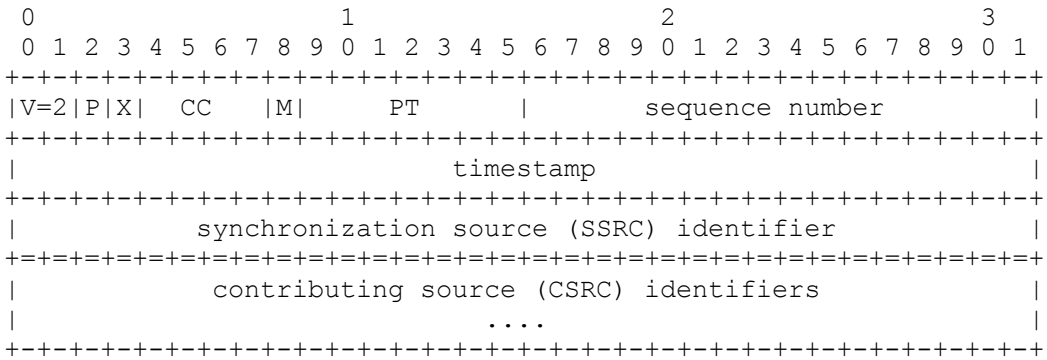
En el servidor tendrás que implementar el encapsulado de los datos de vídeo en paquetes RTP. Para ello tendrás que crear el paquete, establecer los campos en la cabecera del mismo y copiar la carga útil (es decir, un fotograma de vídeo) en el paquete.

Cuando el servidor recibe la solicitud de PLAY del cliente, inicia un temporizador que se activa cada 20 ms. Cuando salte el temporizador, el servidor leerá un fotograma de vídeo del archivo y lo enviará al cliente. El servidor crea un objeto RTPpacket, que representa el frame de video encapsulado en RTP.

El servidor llama al primer constructor de la clase RTPpacket para realizar el encapsulado. *Tu tarea es escribir esta función*. Tendrás que hacer lo siguiente[1]:

1. Rellenar el campo version (V) con el valor decimal 2.
2. Rellenar los campos padding (P), extension (X), número de fuentes (CC), y marker (M). *Todos deberán estar a 0 en esta práctica*.
3. Rellenar el campo Payload Type (PT) para indicar que enviamos un flujo MJPEG según lo establecido en el RFC 3551.
4. Rellenar el campo número de secuencia. El servidor da este valor como argumento `Framenb` al constructor.
5. Rellenar el valor de timestamp. El servidor da este valor como argumento `Time` al constructor.
6. Rellenar el identificador de fuente (SSRC). Este valor identifica al servidor y por tanto puedes elegir cualquier valor que te parezca.

Dado que sólo tenemos una fuente de datos (i.e., el campo CC es 0), el campo CSRC no se incluirá en la cabecera y la longitud total de esta será 12 bytes. Es decir, solo se incluirán las tres primeras filas del diagrama mostrado a continuación.



Debes rellenar la cabecera en el array `header` de la clase RTPpacket. También tendrás que copiar la carga útil (dada como el argumento `data`) a la variable `payload`. La longitud de la carga útil viene dada en el argumento `data_length`.

[1] Las letras entre paréntesis se refieren a los campos de la cabecera RTP.

Trabajando con bits

Debido a que el campo de cabecera de la clase RTPpacket es un array de tipo byte, tendrás que configurar la cabecera byte a byte. El primer byte tiene los bits 0-7, el segundo byte tiene los bits 8-15, y así sucesivamente. En Java un entero (int) tiene 32 bits o 4 bytes.

A continuación mostramos algunos ejemplos de cómo alterara el valor de determinados bits. Por ejemplo, para poner a 1 el bit *n* en la variable `mybyte` de tipo byte:

```
mybyte = mybyte | 1 << (7 - n);
```

Para rellanar los bits *n* y *n + 1* de la variable `mybyte` con el valor `foo`:

```
mybyte = mybyte | foo << (7 - n);
```

Nótese que `foo` debe tener un valor que se puede expresar con 2 bits, es decir, 0, 1, 2, or 3.

Para copiar un entero `foo` de 16 bits en las variables `b1` y `b2` de tipo byte:

```
b1 = foo >> 8;
b2 = foo & 0xFF;
```

La variable `b1` tendrá los 8 bits más significativos de la variable `foo` y `b2` tendrá los 8 bits menos significativos de la variable `foo`. También puedes copiar un entero de 32 bits en 4 bytes haciendo algo similar.

Si no estás acostumbrado a trabajar a nivel de bit, puedes encontrar más ejemplos aquí.

Ejemplo

Supón que quieres rellenar el primer byte de la cabecera RTP con los siguientes valores:

- V = 2
- P = 0
- X = 0
- CC = 3

En binario esto se representaría de la siguiente forma:

```
 1 0 | 0 | 0 | 0 0 1 1
 V=2   P   X   CC = 3

2^7 . . . . . . . . 2^0
```

Puedes calcular el valor entero de ese número y aplicarlo al primer byte de la cabecera.

Ejercicios Opcionales

- Calcula las estadísticas de la sesión. Tendrás que calcular la tasa de pérdida de paquetes RTP, la tasa de datos de vídeo (en bits o bytes por segundo) y cualquier otra estadística interesante que se te ocurra.
- La interfaz de usuario en el cliente tiene 4 botones para las 4 acciones. Si comparas esto con un reproductor multimedia estándar, puedes ver que sólo tienen 3 botones para las mismas acciones, PLAY, PAUSE y STOP (que corresponden aproximadamente a TEARDOWN). No hay ningún botón de SETUP. Dado que el SETUP es obligatorio en una interacción RTSP, ¿cómo se implementaría en un reproductor multimedia? ¿Cuándo envía el cliente el SETUP? Piensa en una solución e impleméntala.
- En la versión actual, el cliente y el servidor sólo implementan las interacciones RTSP mínimas necesarias y PAUSE. Implementar el método DESCRIBE que se utiliza para pasar información sobre el flujo multimedia. Cuando el servidor recibe una petición DESCRIBE, devuelve un archivo de descripción de la sesión que indica al cliente qué tipos de flujos hay en la sesión y qué codificaciones se utilizan. Muestra esta información en la consola.

Apéndice

En esta práctica, el servidor transmite un vídeo que ha sido codificado en un formato de archivo MJPEG propio. Este formato almacena el vídeo como imágenes JPEG concatenadas, con cada imagen precedida por una cabecera de 5 bytes que indica el tamaño de bits de la imagen. El servidor analiza el flujo de bits del archivo MJPEG para extraer las imágenes JPEG sobre la marcha. El servidor envía las imágenes al cliente a intervalos periódicos. El cliente muestra las imágenes JPEG individuales a medida que llegan del servidor.