# MARCO DE REFERENCIA PARA LA PROVISIÓN DEL SERVICIO DE IPTV MÓVIL EN COLOMBIA.

Anexo E
Aportes adicionales

**Gabriel Elías Chanchi Golondrino**
**Gustavo Adolfo Gallego Toledo**

Director
Mag. Oscar Mauricio Caicedo Rendón

Universidad del Cauca
**Facultad de Ingeniería Electrónica y Telecomunicaciones**
**Departamento de Telemática**
Línea de Investigación en Servicios Avanzados de Telecomunicaciones

Popayán, Octubre de 2008

## TABLA DE CONTENIDO

## LISTA DE FIGURAS

# ANEXO E
# APORTES ADICIONALES

## E1. DESCRIPCION GENERAL.

Adicionalmente a lo presentado en este trabajo de grado, se pretende exponer en este apartado, algunos aportes extras, relacionados con implementaciones de los protocolos y decodificación del formato de audio sobre el que se fundamenta IPTV Móvil. El ánimo de estas adiciones, es extender la implementación del piloto para la prestación del Servicio de IPTV a la carta, y en general cualquier servicio de IPTV Móvil que se desee desplegar sobre un cliente móvil que no soporte dichos protocolos, pero que soporte las características multimedia necesarias para reproducir dichos formatos.

Los aportes mencionados anteriormente son:

- Implementación del protocolo RTSP para el control de sesión mediante J2ME.
- Implementación del protocolo RTP para el envío de contenido multimedia usando J2ME.
- Decodificación y reproducción de contenido de audio AMR-NB sobre RTP usando J2ME

## E2. IMPLEMENTACION DEL PROTOCOLO RTSP.

El protocolo RTSP es usado para controlar la sesión de streaming de audio y video, de manera independiente al protocolo de trasporte usado para el llevar el contenido multimedia, debido a que este protocolo hace uso de identificadores de sesión para cada cliente; esta característica de independencia hace que el protocolo RTSP sea extensible para trabajar con cualquier protocolo de trasporte, sin embargo es mas difundido su uso en conjunto con el protocolo RTP [1].

El protocolo RTSP es un protocolo basado en texto de acuerdo a la norma UTF-8, trabaja sobre el protocolo TCP y es semejante al protocolo HTTP, lo cual lo hace fácil de leer e interpretar. Con el protocolo RTSP es posible establecer un canal

bidireccional de control independiente al canal de envío de contenido multimedia, a través del cual tanto cliente como servidor pueden intercambiar este tipo de mensajes. La estructura de un mensaje RTSP se muestra a continuación:

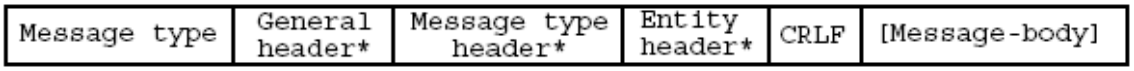


**Figura 1. Estructura del Mensaje RTSP.**

El tipo de mensaje puede ser de solicitud o respuesta, los campos con * son opcionales y el campo CRLF representa un espacio y salto de carro.

### E2.1 Mensajes de Solicitud RTSP.

Los mensajes de solicitud pueden ser usados por el cliente o por el servidor y su estructura se muestra a continuación:

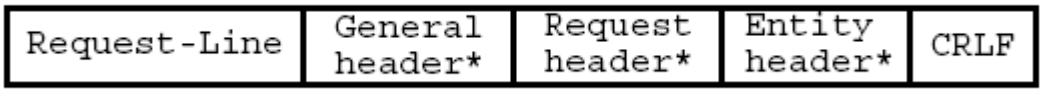


**Figura 2. Mensaje de Solicitud RTSP.**

El campo mas importante del mensaje es **Request – Line** y su estructura se presenta a continuación:


**Figura 3. Campo Request – Line.**

El campo Method representa el tipo de acción solicitada por el cliente o servidor, WS es un espacio en blanco, Request – URI corresponde a la dirección RTSP del recurso al que se desea acceder en el Servidor de Streaming, RTSP – Version es la versión del protocolo RTSP en cuestión.

Los posibles valores del campo Method son: DESCRIBE , GET_PARAMETER , OPTIONS, PAUSE, PLAY, PING, REDIRECT, SETUP, SET_PARAMETER, TEARDOWN, el siguiente es un ejemplo de un mensaje de solicitud, de acuerdo a lo anterior:

**OPTIONS rtsp://iptvmovil.net/ RTSP/1.0**

### E2.2 Mensajes de Respuesta RTSP.

La estructura del mensaje de respuesta se presenta a continuación:

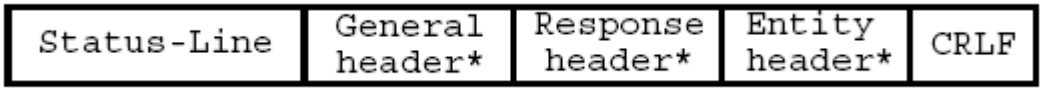


**Figura 4. Mensaje Response RTSP.**

El campo de mayor relevancia es **Status – Line** y su estructura es como se muestra a continuación:

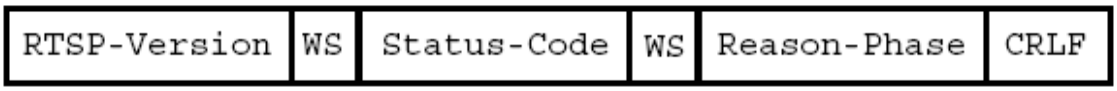


**Figura 5. Campo Status – Line.**

El campo **RTSP-Version** corresponde a la versión del protocolo RTSP, **WS** representa un espacio en blanco, **Status-Code** es un código de respuesta y Reason–Phase es una frase asociada con el código de respuesta.

Los valores del campo **Status–Code** y **Reason–Phase** son: 1XX – Informal, 2XX – Sucess, 3XX – Redirection, 4XX – Client Error, 5XX – Server Error. Un ejemplo de mensaje de respuesta es el siguiente:

**RTSP/1.0 551 Option not supported**

### E2.3 Comunicación RTSP.

En la comunicación RTSP entre el cliente móvil y el Servidor de contenidos se realiza el intercambio de mensajes, para describir, configurar y solicitar el contenido mediante los métodos del mensaje de solicitud de RTSP. Una vez el proceso de descripción, y configuración del contenido ha tenido una respuesta exitosa inicia el proceso de envío de paquetes mediante el protocolo RTP, tal como se muestra a continuación:

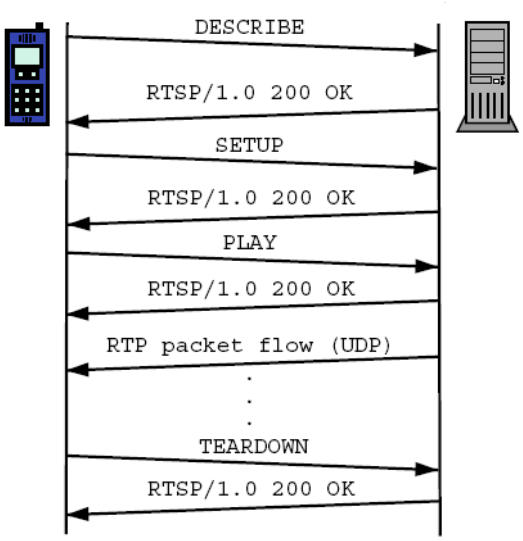


**Figura 6. Comunicación RTSP.**

## E2.4 Implementación del protocolo RTSP usando J2ME.

La implementación del protocolo RTSP mediante J2ME esta pensada para dispositivos móviles que soporte Java, con las funciones de conectividad habilitadas, al igual que la de reproducción del formato de Audio AMR. En otras palabras para dispositivos que soporten la configuración CLDC 1.1 y el perfil MIDP 2.0.

Teniendo en cuenta que el protocolo RTSP trabaja sobre TCP, la conexión del cliente móvil con el Servidor de Contenidos se implementa haciendo uso de sockets.

```
hilo = new Thread(this);
SocketConnection sc = (SocketConnection) Connector
        .open("socket://127.0.0.1:554");
```

**Figura 7. Conexión RTSP.**

Una vez establecida la conexión se abren los flujos de entrada y de salida por medio de los cuales se van a comunicar, el cliente con el servidor de Streaming.

```
is = sc.openInputStream();
os = sc.openOutputStream();
```

**Figura 8. Flujos RTSP.**

Para el envío de cada uno de los métodos del mensaje de solicitud RTSP, se elabora una cadena de caracteres de acuerdo al formato de los mensajes presentado anteriormente de tal manera que a través del flujo de salida se envíe dicho mensaje.

```
// debug
System.err.println(" ====== CLIENT REQUEST ====== ");
System.err.println(fullCommand + CRLF + CRLF);
System.err.println(" ============================ ");

// send a command
os.write((fullCommand + CRLF + CRLF).getBytes("UTF-8"));
os.flush();
```

**Figura 9. Envío del Mensaje RTSP.**

Las respuestas enviadas desde el Servidor de contenidos, son recibidas por medio del flujo de entrada y procesadas mediante análisis de cadenas de caracteres.

```
StringBuffer sb = new StringBuffer();
int input = is.read();

while(is.available()!=0)
{
            sb.append((char)input);
            input = is.read();


}
```

**Figura 10. Recepción del Mensaje RTSP.**

## E3. IMPLEMENTACION DEL PROTOCOLO RTP.

El protocolo RTP es el protocolo para el envío de contenido de streaming en tiempo real, al trabajar sobre UDP no garantiza la entrega en orden, ni la entrega a tiempo, ni la calidad de servicio de la información multimedia.

La siguiente es la estructura del encabezado RTP:

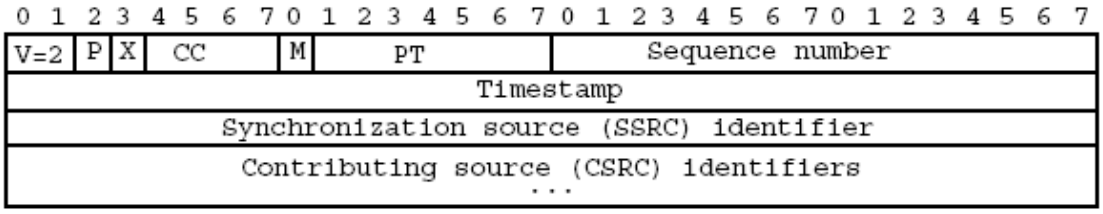


**Figura 11. Encabezado RTP.**

Algunos de los campos mas importantes del encabezado RTP son:

**PT (Payoad Type):** Contiene la información del tipo de información que el paquete RTP lleva en su carga útil, en este caso sirve para diferenciar el contenido de audio del contenido de video.

**Secuence Number (Número de Secuencia):** Es el numero aleatorio mediante el cual los paquetes son numerados, este valor se incrementa por cada paquete enviado por el Servidor.

**Timestamp (Estampa de Tiempo):** Es el numero que representa el orden en el que los paquetes RTP deben ser reproducidos.

Tras el encabezado del paquete RTP viaja la carga útil, que corresponde con el contenido multimedia de audio AMR o de video H263.

### E3.1 Implementación del protocolo usando J2ME.

Al igual que para el protocolo RTSP, el protocolo RTP requiere las mismas características técnicas a nivel del dispositivo móvil, mostradas en la parte superior.

Teniendo en cuenta que el protocolo RTP trabaja sobre el protocolo UDP, la conexión del cliente móvil con el Servidor de contenidos se realiza mediante datagramas.

```
// open a local socket on port 8080 to read data to
form.append("\n llegamos start");
socket = (DatagramConnection) Connector.open("datagram://:8080");
form.append("\n empezo play");
```

**Figura 12. Conexión RTP.**

Posteriormente los paquetes son recibidos haciendo uso de la conexión establecida:

```
try {
    packet = socket.newDatagram(socket.getMaximumLength());
    socket.receive(packet);
} catch (IOException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

**Figura 13. Recepción de paquetes RTP.**

La información del contenido multimedia es recibida en forma de arreglos de bytes, e interpretada de acuerdo al encabezado de la trama y a la longitud de cada uno de sus campos.

```
PT = (byte) ((buf[1] & 0xff) & 0x7f);

marker = (byte) ((buf[1] & 0xff) >> 7);
System.out.println("_____");
System.out.println("valor " + (buf[1] & 0xff));
System.out.println("El valor del marker es " + marker);
System.out.println("_____");

seqNo = (short) ((buf[2] << 8) | (buf[3] & 0xff));

timeStamp = (((buf[4] & 0xff) << 24) | ((buf[5] & 0xff) << 16)
        | ((buf[6] & 0xff) << 8) | (buf[7] & 0xff));

SSRC = (((buf[8] & 0xff) << 24) | ((buf[9] & 0xff) << 16)
        | ((buf[10] & 0xff) << 8) | (buf[11] & 0xff));
```

**Figura 14. Procesamiento de los campos RTP.**

## E4. DECODIFICACIÓN Y REPRODUCCIÓN DE CONTENIDO DE AUDIO AMR-NB SOBRE RTP USANDO J2ME.

La carga útil que viaja sobre el protocolo RTP corresponde a las tramas de audio AMR-NB (Narrow-Band). Cada una de las tramas AMR tienen un encabezado que se muestra en la siguiente figura:

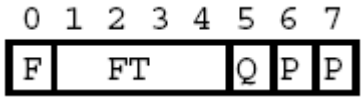


**Figura 15. Encabezado de la trama AMR.**

Los campos del encabezado AMR son:
F: Indica si esta es la última trama de información AMR-NB.
FT: Indica el tipo de formato AMR que se esta usando, AMR-NB o AMR-WB.
Q: Es el indicador de calidad de la trama.
P: Corresponde a información de relleno.

Cuando viajan varias tramas AMR-NB sobre el protocolo RTP, la configuración se muestra a continuación:

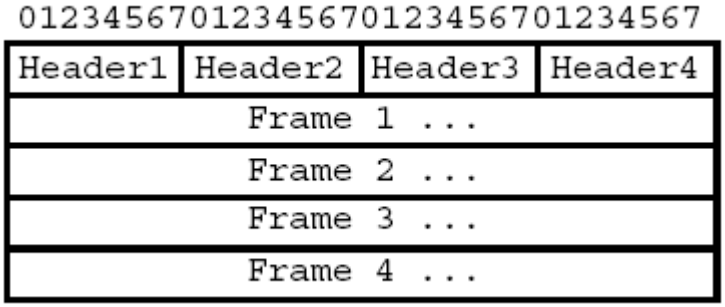


**Figura 16. Trama RTP con AMR.**

Donde Header(n) corresponde con los encabezados de cada una de las tramas AMR-NB y Frame(n) hace referencia a la información util de cada una de estas tramas.

Para pasar la información de la trama y convertirla en un archivo de audio AMR, es necesario adecuarlo a la siguiente estructura:

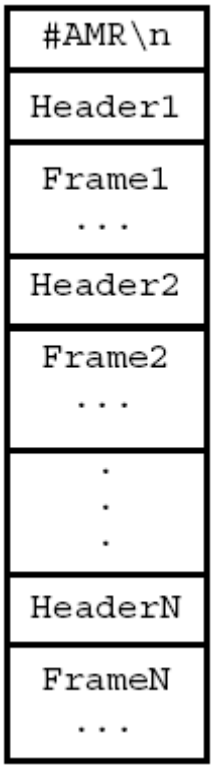


**Figura 17. Estructura del archivo de audio AMR.**

### E4.1 Reproducción del contenido AMR sobre RTP.

Lo primero que se hace es obtener la carga útil a partir del paquete RTP:

```
byte payload[] = new byte[packet.getLength() - 12];
```

**Figura 18. Carga útil del paquete RTP.**

Después se separa la información de encabezado de la información de carga útil del formato AMR:

```
for (int i = 1; i < data.length; i++) {

    if(i<=5){
        reproduccion1[(aux + ((i-1)*32))]=0x3c;
    }
    else{
        ntrama=((i-6)/31) + 1;
        reproduccion1[(ntrama + i + (160*npaquete))]=data[i];
    }
}
aux=aux + 160;
npaquete++;
```

**Figura 19. Separación de encabezado y carga útil**

Posteriormente se arman pequeños archivos de audio AMR, con el nuevo encabezado del archivo:

```
reproduccion2=new byte[4806];
reproduccion2[0]='#';
reproduccion2[1]='!';
reproduccion2[2]='A';
reproduccion2[3]='M';
reproduccion2[4]='R';
reproduccion2[5]='\n';
```

**Figura 20. Encabezado del archivo de audio**

Finalmente se reproduce el contenido de audio AMR-NB:

```
bis1 = new ByteArrayInputStream (RTPSourceStream.temporal);
player1 = Manager.createPlayer(bis1,"audio/amr");
player1.realize();
player1.prefetch();
fin1=true;
```

**Figura 21. Reproducción del archivo de audio**

Cabe resaltar que mientras se van reproduciendo cierta cantidad de tramas AMR-NB es necesario continuar leyendo en un hilo distinto la información restante.

## REFERENCIAS

[1] M. sillen y J. Nordlund. “Real-Time Audio Streaming in a Mobile Environment Using J2ME,”. Julio 2005. [Online].Disponible: http://www.cs.umu.se/education/examina/Rapporter/SillenNordlund.ps