Journal of Mobile, Embedded and Distributed Systems, vol. III, no. 2, 2011
ISSN 2067 – 4074

Security Issues in Streaming Server for Mobile Devices Development

Dan Barbu
Cybernetics, Statistics and Economic Informatics Faculty
Academy of Economic Studies
Romana Square 6, Bucharest
ROMANIA
danciprian.barbu@gmail.com

Abstract: The paper presents a solution for streaming audio and video content in IP networks using RTP and SIP protocols. Second Section presents multimedia format and compression for the audio content that is streamed by SS4MD. Streaming protocols are shown in third section. In the forth section there is an example of an application which does uses all above. Conclusions are contoured in the final chapter.

Key-Words: mobile multimedia streaming, RTP/RTSP/SDP, SecureRTP, SS4MD – Streaming Server for Mobile Devices

1. Introduction

The streaming technology allows one to send media content such as music, video files or just live coverage over the internet. Instead of downloading the entire multimedia file the player is just 'taking' as much content as could possibly be shown over a period of time. While the already downloaded content gets played, the player 'grabs' much more content form the streaming server which would get stored in its internal buffer for future usage.
Devices like video phones, notebooks or even desktop are all able to use this technology. The streaming process is almost invisible to the viewer except for the short period of initial buffering when many parameters gets 'agreed' with the server.
Some of the most common the advantages of using any streaming architecture over the conventional media files usage could be known as following:

- real time coverage possibility;
- protection against media piracy;
- individual user identification;
- limitless monetize possibilities.

Probably the basis in whether or not streaming is to be something useful for our development represents the fact that it can deal great with network bandwidth limitations.
The streaming technology is acting undoubtedly towards people and can be easily named a service. Therefore as any other service it must take good care of people needs and aspirations. Having this in mind just imagine the impact of such a service which do offers instant access to any kind of media. People would definitely want to access as well as taking it through as various use cases as possible.
Therefore I do consider that the universal goal in using the streaming technology is to offer as much media content as possible to a myriad of devices in order to guarantee users' satisfaction.
Turning media content available to end users using the streaming technology could be done in one of the following 3 ways:

- unicast streaming - a one-to-one
- connection between the server and a client using the request/receive paradigm;
- broadcast steaming- a one to many relationship between a server and more clients;

This is a post conference paper. Parts of this paper have been published in the Proceedings of the 3rd International Conference on Security for Information Technology and Communications, SECITC 2010 Conference (printed version).

- multicast streaming - is a many-to-many relationship between servers and the clients receiving the stream;

The difference between broadcast and multicast is that in multicast streaming the audio content gets transmitted to a so called multicast group whereas for the former doesnt. Receivers that want to receive data join the multicast group and the network routing protocol is the one responsible for making the traffic available to everyone.

By offering either audio or video content doesn't imply it gets free of charge or it could be used for any specific purpose. Whether or not it's about IPTV or just the case when a user downloads on his IPod the latest release of his favorite singer, media should be protected against any illegal usage. This kind of copyright infringement has been representing an issue ever since media storage devices such as Betamax or VHS tapes were first released to general public back in 1980s. Therefore to frame usage scenarios, a control mechanism named Digital Rights Management (DRM) was released. It has been targeted towards hardware manufacturers, publishers, copyright holders and individuals in order to impose limitations on the digital content usage as well as on various devices.

One DRM related implementation other than using SRTP is the 'FairPlay' technology from Apple. It is built into the QuickTime multimedia software and used by the iPhone, iPod, iPad, Apple TV, iTunes, iTunes Store and the App Store. Any protected song or other form of media purchased from the iTunes Store with iTunes is encoded with FairPlay. It digitally encrypts the AAC audio files and prevents users from playing these files on unauthorized computers.

DRM restrictive policies are present on the gaming industry as well not only on streaming media. Dedicated gaming server such as Blizzard's online Starcraft 2 or EA's Need for Speed one are just a small number of which are using DRM techniques. Games have always been subject to piracy behavioral acts and therefore the producers have to protect somehow their intellectual property.

SS4MD – Streaming Server for Mobile Devices is the application which I will refer later as a support of the theoretical aspects covered in this article.

2. Multimedia file formats and compression used in SS4MD

Media content has dealt a great development during the past decade. Modification consisted in both structural and functional points of view. The path from audio cassettes, later mp3, nowadays torrents represent just a glimpse of what people have been facing with. What for many appeared a closed 'battle' has transformed in a more agile war in terms of media distribution. However media producing companies have been trying to impose their view upon the open market and the overall success of YouTube like sites couldn't diminish their actions.

2.1. WAVE

Every common thing within this universe is perceived by everyone through its vibration. This kind of action could be represented just like as waves are in an ocean. The oscillation length, meaning the distance between one wave's peek and the very next one, is a characteristic by which we could easily group different happenings. Waves vibrating at different frequencies manifest themselves different. Because of this feature the regular sound and lightening events could be observed by all human beings. Beside these we could group events in other groups such as infrared and ultraviolet spectra as well as sub and ultrasonic categories.

Frequencies are typically described in units called Hertz (Hz) which mean "cycles per second". Despite human range of frequencies being between 20 Hz and 20 kHz most adults can only hear sounds below 16 Hz although most of the women tend to preserve the ability to hear higher frequencies than men do.

Another important piece of information is the fact that the human brain is filtering the information it gets. This way it allows only what is really interesting to get to us, tunneling important info to the fore and ignoring any irrelevant data.

ISSN 2067 – 4074

These above represents the principles of every know codec. There is little point in storing information that cannot be perceived by the human ear. Some of the best recordings store a lot more information than we could be hearing just because of the recording equipments use. All these are sensitive to a broader range of sounds and audio resolutions than normal human year.
The transition from analog signal to digital is done by measuring the wave many, many times a second and converting that measurement into a binary number, accordingly to [1]. This process is called sampling and every of measurement is named sampling period.
Another important factor, which is just as vital as sampling period is the bit depth. The latter could just be translated as the number of bits used by every sampling period.
Therefore higher bit depth would allow the analog signal to be converted using more binary numbers.
As such 8-bit sampling uses an 8 digit binary number to record the level, having 2^8, or 256 potential values and 16-bit sampling is using 2^16 (65,536). As such one could easily conclude the bigger the depth is, the better the digital recording will sound like.
One of the main disadvantages of using this media type could be represented by the great storage cost it has.

For instance:

- an uncompressed audio data, CD quality 44.1 KHz in sample rate, 16 bits (having 2 bytes per channel), 2 channels (stereo), lasting 30 minutes would result 44100 x 2 x 2 x 30 x60 ~ 302 MB;
- 96 KHz, 24 bits (having 3 bytes per channel), 48 channels and lasting 30 minutes would occupy 96000 x 3 x 48 x 30 x 60 ~ 23.17 GB.

These examples are the main reason why uncompressed audio data isn't used at large scale. It cannot be high quality without very high storage costs. Therefore it would be more appropriate if there would be used a compression factor

2.2. MPEG

MP3 file format is one of the most used audio data format. It is both widespread and controversial the same time. It has been dropping from the lips of politicians and advocates for all sides of the controversial intellectual property debate almost ever since it was released.
How to define MP3? Usually, as most other things do, MP3 file format has at several possible definitions available as well. One of them, the long one, available in technical papers which filled with math doesn't make the purpose of this article. As such I will not point it out and instead I'll be using the definition available in newspapers, gazettes and so forth. Accordingly to these sources MP3 file format is a result of a transformation process applied to a digital raw signal. Thought this process it gets eliminated any unheard sound by the human year. The size reduction is a great factor which kept this file format in the spotlight for so long.
Human brain plays a great role in filtering and analyzing the signals received when noise is encountered.
One of the most important is the Haas effect. This effect states that two identical sounds arriving within 30-40 ms of each other from different directions will be perceived as a single sound coming from the direction of the first one. It gets used in public address systems, to consolidate the sound from the speaker desk even if the loudspeakers are located far on the side! MP3 techniques use this effect in a process called frequency masking when multiple channels are involved.
MP3 file format stands for "MPEG-x Audio Layer 3". MPEG name means "Motion Pictures Expert Group" which stands for the developers who created this standard.
The most important encoders released by the MPEG group are MPEG-1 and MPEG2. The first one MPEG-1 was was widely used to convert the raw audio material on the size of a CD and was easily adopted by digital satellite/ cable TV companies.
MPEG-2 encoder was a real blast over the audio data market. It was not meant to replace the MPEG-1 but to offer it additional encoding parameters. It has replaced MPEG-1 and set by Advanced Television Systems Committee and

implemented in Digital Video Broadcasting as a de facto standard for over the air digital transmissions such as digital satellite TV services like Dish Network, digital cable television signals. Every one of us may thank to this encoder when watching any setup box.
As Layer 3 of the MPEG-1 or MPEG-2 specification, there are obviously two previous audio layers before MP3, which did not catch on in the consumer market (few of us listen to MP2s at home).
The same is valid for MP1 which was mainly used for digital compact cassettes.
Accordingly to [3] the process of converting the raw material to MP3 consists of 4 different stages:

1. group the raw samples into "frames" each with 1152. For Layer 3 there are two granules of 576 samples each;
2. all these samples are ran through a filter bank that divides the sound into 32 frequency ranges. The purpose of this filter is to let only a specific band of frequencies to pass through. Layer 3 divides each of these frequencies by a factor of 18 creating 576 smaller adaptive bands. Each of these contains 1/576 of the frequency range from the original
3. two parallel processes takes place. A Fast Fourier Transform (FFT) on one side and Modified Discrete Cosine Transform (MDCT). The former process consists of turning each frequency band into information that can be fed into the encoder's psychoacoustic model. The latter is responsible for sorting the samples into different "windows" whether or not these contain constant noise;
4. cold compression which is composed of Quantatization and Huffman coding. At this very point each granule contains two parts mandatory for reconstructing the audio data: the scale factor for each band and the long chunk of Huffman bits.

At play time (after decompression takes place) the decoder combines the two granules into a single frame. As such MP3 file format is composed of frames of audio data following the ID3 metadata.
Some of the elements needed to describe an MP3 file format are:

- **Bit Rate** - number of bits used per unit of playback time after data compression: 8, 16, 24, 32, 40, 48, 56, 64, 80, 96, 112, 128, 144, 160 kbit/s. Considering that audio data stored on a CD has the Bit Rate equals to 1411.2 kbit/s the bitrates like 64, 128 or 160

represents compression ratios of 22:1, 11:1 and 9:1 respectively.

- **Sample Rate** - numbers of samples per second (or per unit) taken from a continuous signal in order to make a discrete signal. Sample rates available for MPEG-2 Layer III (MP3) are 16, 22.05 and 24 kHz. For most MPEG-1 Layer III a sample rate of 44.1 kHz is almost always used because it's the same as for CD- quality.

The total size in bytes for a single frame is:
pointed by the first 4 bytes in Figure 2.2 is detailed in Tab 2.1 below.

- MPEG-1 = 144 * BitRate/ (SampleRate +Padding)
- MPEG-2 = 144/2 *BitRate/ (SampleRate+ Padding)

The file used by SS4MD is MPEG-2 Layer III, 22050 Hz, 64 kbit/s and therefore the frame's size in bytes is: 72 *64000 / (22050+1) ~ 209 bytes.

Figure 2.1 presents the structure of a frame as defined by [3].

```
frame()
{
    header() //4 bytes long
    error_check() – if protection_bit (header) ==0 (2 bytes)
    side_info_structure() – if mode == stereo (32 bytes)
    audio_data() (173 bytes)
}
```

Figure 2.1. Internal Mp3 Frame Structure

ISSN 2067 – 4074

The header part of the frame as pointed by the four first bytes in Figure 2.2 is detailed in Tab 2.1 below.

Table 2.1. Mp3 Frame Header

0xFF	*0xF3*	*0x82*	*0x00*
1111	1111	1000	0000
1111	0011	0010	0000

The Audio_Data within this frame has in this particular case (Padding_bit==1) 173 bytes. It starts at the end of the frame's Side _Info structure and it ends at the very end of this frame.
As it can easily be observed from within the Figure 2.2 the next frame would starts just at the end of the audio data.

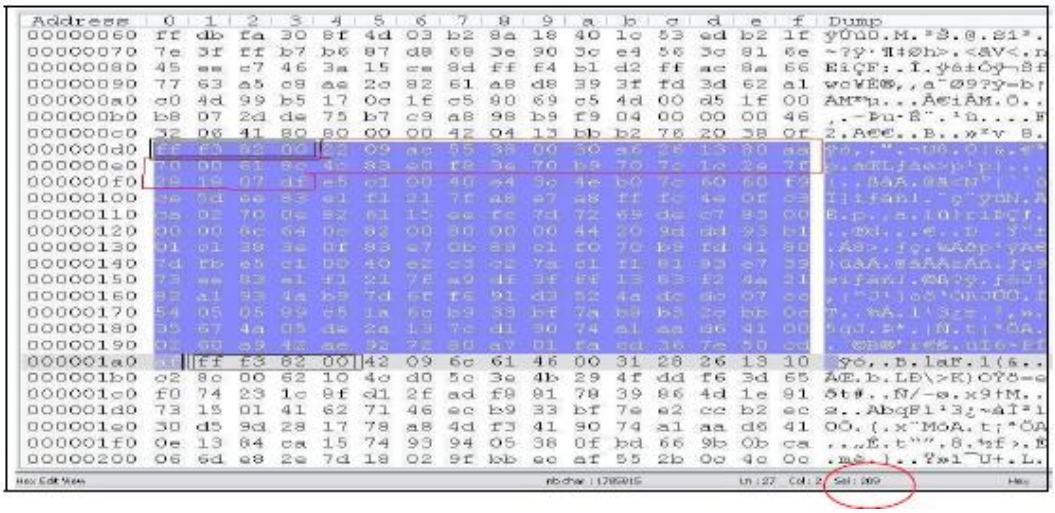

Figure 2.2. Single MP3 Complete Frame

3. RTP/RTSP/SIP concepts used in SS4MD

The media segment development across the years, accompanied by higher and higher demands from its clients in terms of quality has turned the existing infrastructure into an incredibly congested one. The Internet technical connectivity solutions dynamics has a lot slower growing pace than media's enlargement.
For achieving goals as maintaining a continuous data flow or setting up the environment, one could be using Real Time Streaming Protocol (RTSP) [8]. Even though the first version has been released in 1998 [8], the new version waiting currently for approval [5], it has represented a milestone in Internet streaming technology ever since.
Generically speaking RTSP protocol is an application-level protocol over the IP/TCP stack. It provides an extensible framework to enable controlled delivery of real-time data, such as audio and video over either TCP or UDP.

3.1. SDP

In order for one player device to get the media description right, it will be used Session Description Protocol (SDP) together with RTSP. This SDP was first published in 1998 in RFC2327 [9], the same year as RTSP. The protocol was updated in 2006 with RFC4566 [10].
It is widely used in describing multimedia communication sessions. Its purpose is not to deliver real media, but to offer a negotiation standard for media types, formats and other associated attributes.
SDP descriptions are entirely textual using the ISO 10646 character set in UTF-8 encoding. SDP field names and attributes names use only the US-ASCII subset of UTF-8, but textual fields and attribute values may use the full ISO 10646 character set.
As [9] states the SDP syntax consists of a number of lines of text using the following

form: <type>=<value> <type> A serviceable representation of the SDP in presented in Figure 3.1 below:

```
v=0
o=- 0 0 IN IP4 /127.0.0.1
s=Audio
i=<Dan Barbu>
t=0 0
a=tool:Disertatie
a=type:broadcast
a=control:*
a=range:npt=0-223.000000
m=audio 0 RTP/AVP 96
c=IN IP4 0.0.0.0
a=rtpmap:96 MPA/90000
a=ptime:26
a=control:track1
```

Figure 3.1. SDP Attributes / Values

SDP protocol as stated by [9] SDP consists of two kinds of description fields in terms of mandatory and optional requirements.
The most important mandatory SDP Fields are in Table 3.1.
Beside these there are some other significant optional attributes within the SDP; for instance the session attribute field ('a=').
From the purposeful point of view each field within the SDP message could falls into one of the following categories:

- session name;
- time the session is active;
- media info contained by the in the session;
- information about the available bandwidth and contact info.

An important optional attribute is "rtpmap" which defines the mapping of RTP payload codes (which are used in the <format list> in the "m=" field) to a codec name, clock rate, and other encoding parameters. In our example message, one of the "rtpmap" attributes is " a=rtpmap:96 MPA/90000"
Other key parameter (a=ptime:<packet time>) describes the length (ms) carried by each RTP packet.

Table 3.1. Mandatory SDP Fields

Field	Meaning	Format
v=	protocol version	**v=0**
o=	session owner and identifier	**o=<username> <session Id> <version> <network type> <address type> <address>** In our example message above (Figure 3.1), the field was: o=- 0 0 IN IP4 /127.0.0.1 ("IN" stands for Internet, "IP4" means IP version 4 address)
s=	session name	**s=<session name>**
t=	time the session is active	**t=<start time> <stop time>** The times are decimal Network Time Protocol values (in seconds since the year 1900). In applications other than SIP, the SDP message can contain several "t= " lines, specifying additional periods of time when the session will be active.
m=	media type, format, and transport address	**m=<media> <port> <transport> <format list>** In our example message (Figure 3.1),, this is: m= audio 0 RTP/AVP 96 The **<media>** is either "audio" or "video" (if the file is composed of both audio and video sections there would be two "m=" lines). The <port> should always be even (the even port is used by RTP and the next odd port by RTCP). **<transport>** is usually "RTP/AVP", denoting the RTP protocol with the profile for "Audio and Video Conferences with Minimal Control" .

3.2. RTSP

In order for the SDP to be fully functional, it has to be acquired by the player during the first phase of the Real Time Streaming Protocol RTSP communication.

The RTSP is intentionally similar in syntax and operation to HTTP/1.1. However, it differs in a number of important aspects from HTTP: [8]

- RTSP introduces a number of new methods and has a different protocol identifier.
- a RTSP server needs to maintain state by default in almost all cases, as opposed to the stateless nature of HTTP.

ISSN 2067 – 4074

- both an RTSP server and client can issue requests.
- data is carried out-of-band by a different protocol (RTP);
- RTSP is defined to use ISO 10646 (UTF-8) rather than ISO 8859-1, consistent with current HTML internationalization efforts.
- the Request-URI always contains the absolute URI.

If it isn't supplied any port number, the default port for RTSP is assumed, which is 554.
Furthermore a TCP connection is made to the obtained server address/port. The client transmits the DESCRIBE request specifying Sequence Number (Cseq) and Protocol version.
The "CSeq" header field is used in sequence numbering and It is necessary when operating over an UDP connection to ensure initial package ordering. The server will blindly echo the sequence number, including a "CSeq" header in the response messages
The server may respond to the DESCRIBE request in any number of ways but it is safe to assume that any other response than "RTSP/1.0 200 OK" in the first line might indicate an error situation.
After DESCRIBE command, the client issue a SETUP request. The streaming server should respond accordingly to [8]. Anything other than "RTSP/1.0 200 OK" might indicate error, and should result in ending the connection. In the Setup response, high importance has the source attribute within the transport header field, and the *client_port* attribute. It will be needed to start the media tools later on.
To start getting the media packets, the client has to send a "PLAY" request to the server. The same "RTSP/1.0 200 OK" must be sent back along with all the additional headers such as CSeq, or SessionID

3.3. RTP

Considering the RTSP protocol is only used for media control, the actual data flow will take place using a transport mechanism such as UTP or TCP stream.
Responsible for the media content flow is the RTP protocol (Real-time Transport Protocol) [11]. It acts as a "messenger" and only wraps the media content in a way it could be send through the network. This extra info being added by the RTP protocol would be used by the client's player in order to play the media stream.
The RTP message follows right after the UDP header. It is composed of a header of (12 bytes long) and the media data.
The most important parts of the header are:

- *PT, Payload Type 7 bits (10-16)* - which identifies the format of the RTP payload and determines its interpretation, by the application.
- *Sequence Number 16 bits (17 - 32)* - the sequence number increments by one for each RTP data packet sent
- *Timestamp 32 bits (33 - 64*) represent the sampling instant

4. Streaming Server 4 Mobile Devices (SS4MD) solution

The wide range of multimedia devices used today is a result of the great technological explosion started several years ago. The end-users' preferences and aspirations have pushed rapidly the technological blast into a spin which big hardware vendors are struggling to sustain.
If these companies would not come up with something new in a relatively short period of time they would definitely be "eaten" by market acquisition rush.
The fact that these days' devices have so many futures represents results of enormous surveys conducted by the "big players" in terms of hardware development. This continuous motion does concern for sure the software development teams who are trying to add in as many features as possible to their products. As such the software applications have to be portable in order to follow closely the technical boom.
The Internet zone offers today a wide variety of services. The multitude of devices connected to this zone represents the real challenge for developers.

4.1 Architecture

SS4MD application architecture consists of two distinct modules which both are dependent of one another.
Therefore there are:

- A module for managing content requests named "Data Control";
- A module for sending media content to the user named "Data Flow".

The Data Control (Figure 4.1) module receives requests on a TCP port (assigned by the server application) and responds accordingly to RTSP protocol specifications. The commands implemented by the SS4MD application are Describe, Setup, Play or Pause.

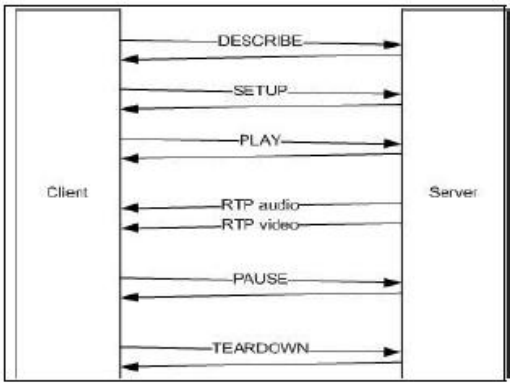

Figure 4.1. Data Control Module [12]

The Data Flow module is represented by the server module which handles the data flow between the client and media storage area.

During the *SETUP* phase of Data Control module, the client application sends its request to receive the media content on a particular UDP port.

The request has the following data: C(Client) -> S(Server): **Transport**: RTP/AVP;unicast;**client_port**=4268-4269

Immediate after the server receives this request it issues a RTSP 200OK message followed by a Transport field just as the one:
S -> C: **Transport**: RTP/AVP;unicast; **client_port**=4268 - 4269; **server_port** = 4270-4271

From this point forward SS4MD application, initiates RTP packets over the above specified UDP ports. The two ports option is used for separate audio / video transmission in case of video files. Considering that currently the SS4MD application only supports audio files it will only take knowledge of the first one described in the associated transport field.
The architecture of the Data Flow module is accordingly to figure 4.2.

The audio data transmission being based on RTP protocol over UDP, does not guarantee the fact that the destination receives all the packets. Unlike TCP connection which handles this issue with the SYN/ACK methods, the UDP protocol is not taking care of any packet loss. If instead SS4MD application had used a TCP connection for transmitting media content, it would have heavily overloaded the network resources. This could mean that that for multiple connections would have been impossible for any source application to guarantee a Quality of Service (QoS) of an acceptable manner. As such, network congestion is today's most important setback factor.

4.2. Solution Data Flow & Traffic Analysis

Stream caching is based on producer-consumer paradigm. The client application is requesting a resource whereas the server has to provide the content after the client reaches a waiting state.

The data flow pertaining to SS4MD application could be classified in one of the two categories. As you will see further on, these two data flows are not independent but are relying one another on some essential aspects. The two data flow modules are:

1. TCP stream data flow – in terms of media control sequence;
2. UDP stream media content flow.

ISSN 2067 – 4074

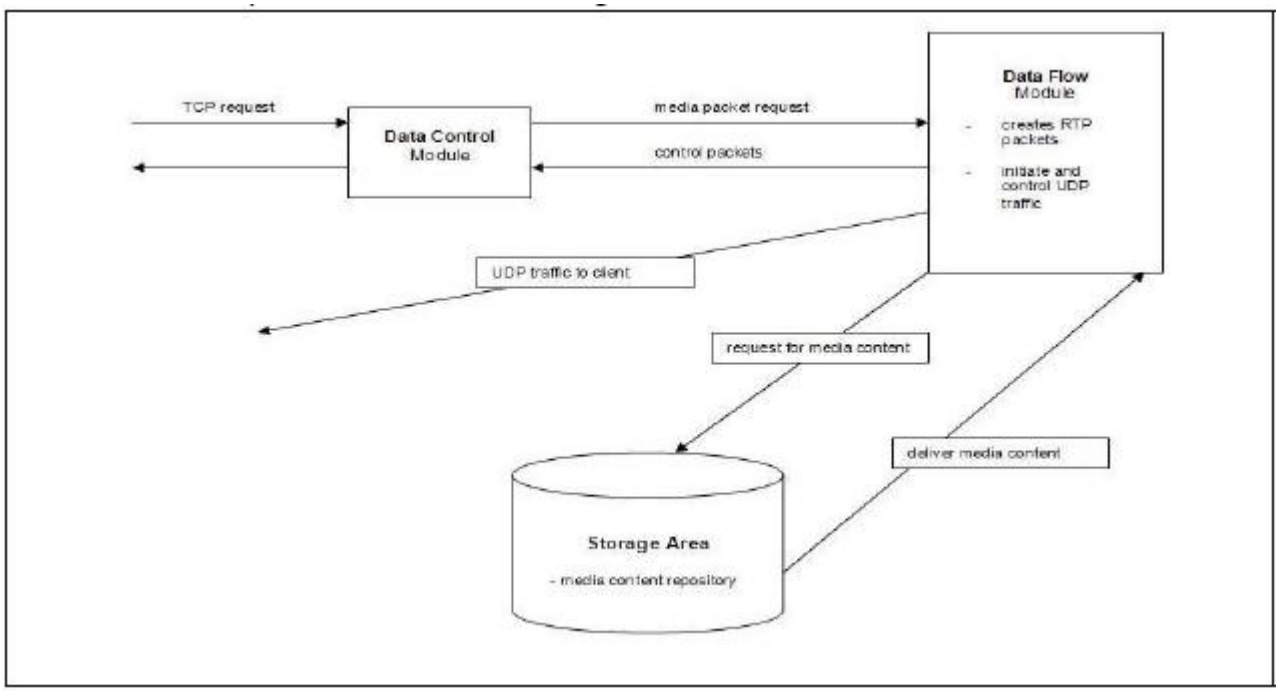

Figure 4.2. Data Flow Module

1) TCP stream data flow – in terms of media control sequence.
The SS4MD application accepts connection on the TCP port 9000. Therefore it sets the TCP port to the Listening state and waits for input any connections.

A particular example of the communication flow, pertaining to streaming, through port 9000 is show in figure 4.3.

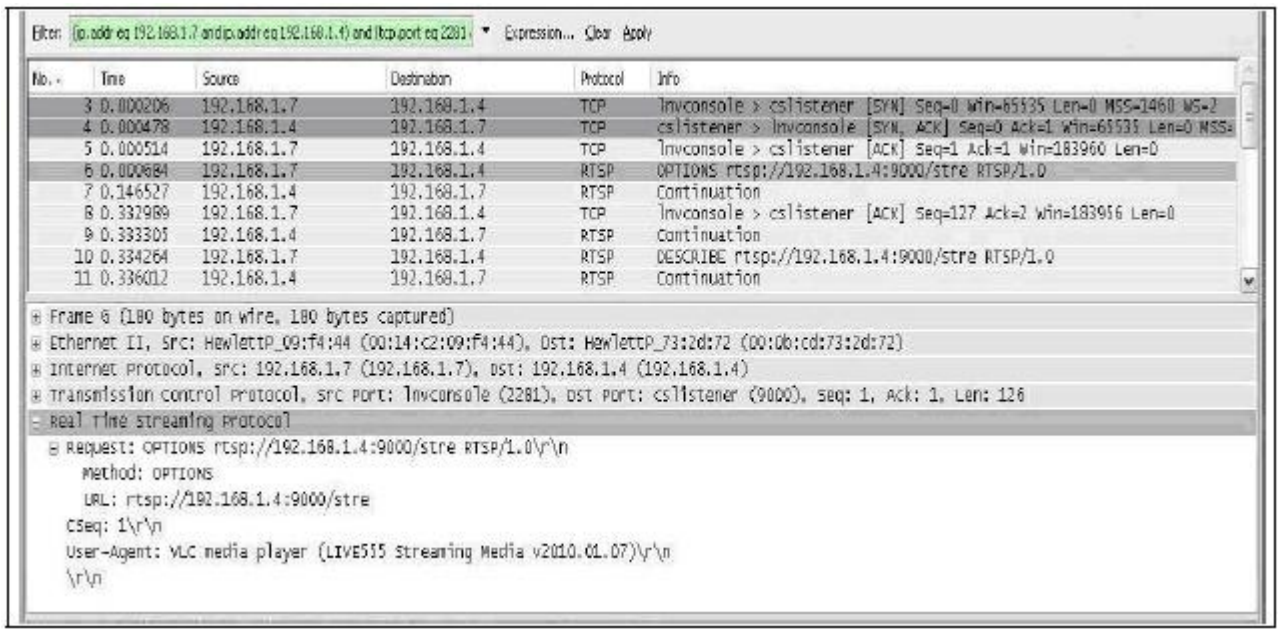

Figure 4.3 RTSP Traffic Capture

The source host (**192.168.1.7**) in Fig 4.3, represents the computer which had requested the media content (the client computer running a VLC like client). The server has the IP address **192.16.1.4**

The client sends the server the Options request in a packet which contains the **Ethernet** Header, followed by the **IP** header, **TCP** Header and RTSP message. The highlighted part of the packet content (figure 4.3) represents the **Options** RTSP message

2) UDP stream – media content flow

UDP stream consists of a series of steps which are in direct linkage to the RTSP stream flow. If the steps are not executed in a proper manner and most importantly in the order I will describe, the entire SS4MD application would be nothing more than useless code.
As such, setup sequence steps, in order to appropriately use the UDP stream are:

- client would open two different UPD ports (audio -video) on the host OS;
- client would sent the numbers of the above to the server;
- the server takes knowledge of the two ports;
- the server sends media content wrapped a UDP datagram structure The correlation with the RTSP data flow is very important as it represents the kick off of the media content flow.

This linkage with RTSP is marked on the **Setup** command sent by the client towards the server. The communication is presented in figure 4.4.

As shown in figure 4.4 the client initiate the following steps:

- opening UDP ports before initiating the Setup Command;
- send the UDP port numbers to the server is done through the Transport attribute of the RTSP command Setup;

As exposed the required UDP ports that would receive media content data are 2282-2283.

The server application step takes knowledge of the two ports and opens two different UDP ports on its side for correspondence transmission.

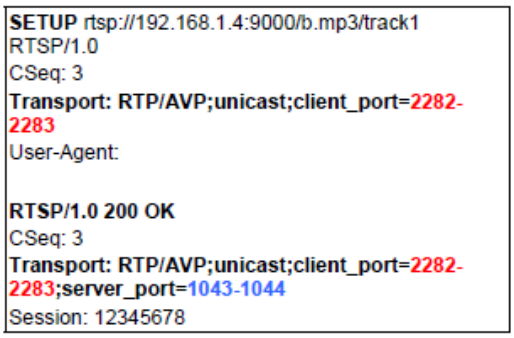

Figure 4.4. UDP Stream – RTSP Correlation

After parsing the Transport attribute, received on the TCP port as part of the RTSP Setup command, the server would be able to further process media content request to completion.

After taking knowledge of the client ports the server application is preparing the UDP packets to be sent to the client.
The audio file used by the SS4MD application has the following characteristics. These will be necessary as calculating the RTP & SDP related fields as presented in section 3 of this paper.

There will get all attributes calculated based on the flowing file specifications

- MPEG -2 Layer III (.mp3);
- bitRate = 64000 bits/s;
- sampleRtate = 22050 Hz;
- channel stereo.

Being coded using MPEG-2 Layer III, the frames are containing **576 samples** each.

Therefore accordingly to ***sampleRate*** definition (the number of samples per second) yields:
No of Frames / second = 22050 / 576 = 38.28 frames / second

As such any player in order to play the sounds in a continuous manner, would require to receive *1 frame each - -* ***26.12 ms*** *(1000 / 38.28)*. Therefore the server would have to be able to pack the frame content and send it to the receiver in little less time than 26.12 ms.

In order to be able to compute the RTP packet, the server has to determine the reading offset of the mp3 file. As such it

ISSN 2067 – 4074

has to be able to 'read' exactly as many bytes as 576 samples would take.

For MPEG-2 Layer III, the following formula must be applied in order to precisely determine the frame length: **frameLength** = 144 / 2 * BitRate / (SampleRate+ Padding) **=** 144 / 2 * 64000 / (22050 + 0) **≈ 209** bytes.

One such frame could be just the one presented in Fig 2.2. One can easily distinguish the following structures:

- 4 byte **Frame Header** (***0xFF 0xF3 0x82 0x00***);
- 32 bytes **Side_Info Structure;**
- 173 bytes **Audio Data Content;**

The streaming flow in order to be played in a continuous manner has to be composed of such frames sent at a rate of at least one every **26 ms**.
After the server application reads the media content it constructs a UDP packet containing the already read information together with the RTP packet specific header.

The RTP header fields are used as follows:
payload type: Distinct payload types should be assigned for video elementary streams and audio elementary streams. In the above example I have used a dynamic payload type:

- **96 sequence number:** the packet's number, by which the client player application would reorder of all packets when arrived;
- **timestamp**: ***32-bit 90K Hz*** timestamp representing presentation time of MPEG picture or audio frame. It is the same for all packets that make up a picture or audio frame. It may not be monotonically increasing in audio stream.

Therefore the media file being sampled at 22050 Hz (samples per second) and having each frame containing 576 samples then the timestamp would have to increase with one for each sampling period therefore 576 per frame.

As such 576 samples per frame for 22050 Hz would represent in 90000 HZ sampling period:
TimeStamp / frame = *90000 x 576 / 22050* ≈ **2351** – added for each frame

The Payload would now be exactly the audio frame. In order for the player application to be able to decode the media data, beside the RTP header and in front of the payload there is required another kind of structure.

This structure [13] is defined in [4] and it is composed of 4 bytes (green mark) Fig 4.5. It has a 2 byte field which represents the byte offset into the audio frame for the media content data in this packet

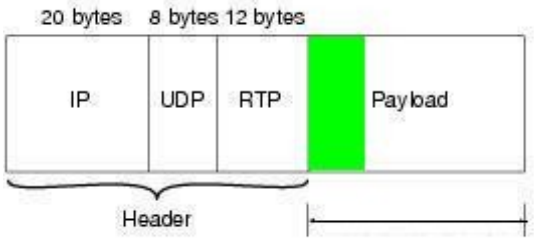

Figure 4.5. UDP+RTP+Audio

In order to compute the RTP packet in a proper way, the SS4MD packetize module has to test first whether the MPEG frame header has the ProtectionBit or the PaddingBit set.
If the ProtectionBit would be set, the audio frame would contain right after the header, a **2 bytes** CRCCheck structure.
Checking whether or not the audio frame is padded is an extremely important task. It could be translated as the server application module responsible for reading the audio frame would have to be able to adjust to certain situations. As such, the module has to set its reading pointer back and forth depending on the padded bit.
The UDP flow could be implemented within the Data Flow module using any regular while loop. Therefore the application is sending the packets at a rate specified by the *.timeout()* method of its the current thread.

4.3. Software Quality Analysis

During the past years, the overall performance was measured only by

examining simple metrics such as network utilization, CPU, disk transfer ratio, etc. These days such basic methods are useless considering the distributed architecture of the streaming systems.
In order to capture the complex interdependence between different modules one could use the *Reliability* indicator.
Considering the case where SS4MD application would be able to adapt the media content broadcast under the heterogeneous network within the Internet zone, a proper solution would be to extend the streaming architecture. It could have a great impact using storage areas as spread around the Globe as possible. One such solution might come up form Amazon which offers *Cloud Front Web Service* in addition to Amazon *Simple Storage Service* (S3). The Cloud Front option offers the possibility to dynamically *divert the user's request to the nearest storage area* in order to maintain the Reliability factor at high values.
Another quality characteristic could be represented by Scalability [6]. It is meant to express how cost-efficient a solution composed of several mutually supporting modules is, against a specified usage scenario. One such usage scenario might become a precise requirement in term of capacity range when acquiring the solution.
Scalability is most common characterized by low equipments cost. But when analyzing cost-effectiveness, it must be included additional costs as well. These ones could be named as:

- the installation cost of a certain system;
- the operating expenses;
- the maintenance cost of the entire infrastructure;
- the training cost of the service team and so forth.

Other QoS characteristic is represented by the Portability indicator, as stated by [7]

5. Conclusions

The major argument in whether or not streaming is something useful for our development is the fact that it can deal great with network bandwidth limitations. I have just imagined the following perspective: 'Wouldn't someone want to view a live sport event on its Smartphone while on a picnic 50 km away from any TV station?' I bet he would. Even more than that, any person interested in the media content offered would pay good money on it as well.
Therefore the only solution would be to send media in a way it could be played by the device disregarding the media content type. It could be roughly compared to a TV set functionality. The latter doesn't care either HD content gets played or either the movie was DivX encoded or so forth. It's just set up to play any kind of media which description was first implemented in its standards. The same mechanism applies even if the above user would access an iTunes like store. He would want to play a sound track or a movie trailer without being forced to wait while the entire media content gets downloaded.
As such the ultimate objective in using streaming technology is to offer as much media content as possible to a myriad of devices in order to guarantee users' satisfaction. Don't you consider it's all about having fun! Imagine what would the young generation feel like when being offered on the geography class full live coverage on their classroom desktop from an ongoing a natural event? Further away streaming could be used in learning activities, live meeting within corporations with strong respect to the costs being involved.
One important setback in using the steaming technology over the internet is the fact that most ISPs have turned off the multicasting within their networks. This means that users are bounded being dependent of the streaming options offered by ISPs. Such providers can be easily identified as Orange, Voadafone, Cosmote etc which do offer streaming to their clients, but only on their choosing. To overpass any restriction imposed by the multicast being turned off, there can be a solution using tunneling techniques. (Fig 5.1)
A tunnel is represented two endpoints between which traffic is tunneled. For that reason there must always be two instances

ISSN 2067 – 4074

of a tunneling application running, one at each end of the tunnel. The basic principle of a tunneling capable application is simple enough: the application listens to a number of multicast sessions, encapsulates all traffic received on these sessions and sends it to the other tunneling endpoint. At this end the network packets gets unpacked and are sent by the tunneling application within the local network using multicast.

The kind of applications that are implementing such tunneling options usually runs as a user process at application level. This way these allow indeed easy and lightweight deployment.

The data sent using tunneling techniques could be just as light as unicast UDP "connection" packets between the two endpoints. By grouping media content packets together in order to send them using unicast towards the tunnel application endpoint, network traffic congestion issues could be easily over passed. Imagine what a network traffic would look like having multicast sessions for a hundred receivers! How about for a thousand receivers? The network traffic would be jammed.

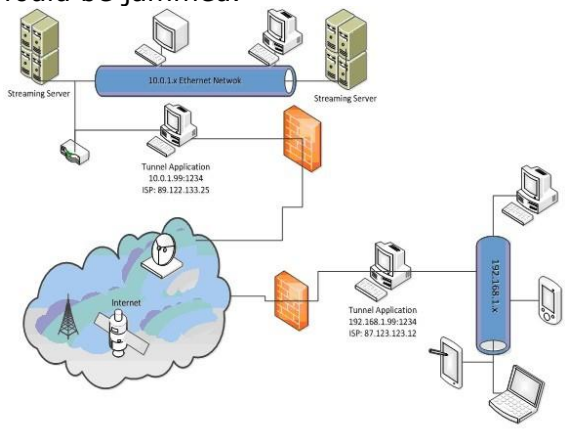

Figure 5.1. Streaming Tunnel Architecture

Compressing packets together is definitely a great mechanism to bypass the above restrictions. Grouping packets would allow for only one IP/UDP header to be used instead of one for each packet. The cost of placing two packets together is a 28 bytes (IP + UDP headers) gain in traffic usage. By grouping n packets together the save is (n-1)*28 bytes.

References

[1] Thomas Wilburn - The AudioFile: basics of uncompressed digital audio – September 2007, http://arstechnica.com/old/content/2007/09/ The-AudioFile-basicsof-uncompressed-digital-audio.ars

[3] International Organization for Standardization (ISO) - Information technology -- Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – July 2001

[4] D. Hoffman, G. Fernando – "RTP Payload Format for MPEG1/MPEG2 Video", Internet Engineering Task Force (IETF), Network Working Group, January 1998

[5] MMUSIC Working Group - Real Time Streaming Protocol 2.0 (RTSP) draft-ietf-mmusic-rfc2326bis-24, July 2010, http://tools.ietf.org/html/draft-ietf-mmusic- rfc2326bis-24

[6] Jerry Zeyu Gao, Jacob Tsao, Ye Wu - Testing and Quality Assurance for Component-Based Software, Artech House Publishers, 1580534805, 2003

[7] Sam Barnes – OS & Browser Usage, http://www.thesambarnes.com/web-project-management/the-whole-world-uses-windows-and-internet-explorer/, 2009

[8] RFC 2326 – RTSP Real Time Streaming Protocol - http://www.ietf.org/rfc/rfc2326.txt

[9] RFC2327 - SDP Session Description Protocol - http://www.ietf.org/rfc/rfc2327.txt

[10] RFC 4566 - SD: Session Description

Protocol http://www.ietf.org/rfc/rfc4566.txt

[11] RFC 3550 – RTP - A Transport Protocol for Real-Time Applications - http://www.ietf.org/rfc/rfc3550.txt

[12] W3C - UA Server RTSP Communication - http://www.w3.org/2008/WebVideo/Fragments/wiki/UA_Server_RTSP_Communication

[13] RFC 2250 - RTP Payload Format for MPEG1/MPEG2 Video - http://www.ietf.org/rfc/rfc2250.txt