An RTSP Proxy for Implementing the IPTV Media Function Using a Streaming Server

Zelalem S. Shibeshi, Alfredo Terzoli and Karen Bradshaw
Department of Computer Science
Rhodes University, P.O. Box 94, Grahamstown 6140
Tel: +27 46 6038247, Fax: +27 46 6361915
E-mail: zelalems@rucus.ru.ac.za; {A.Terzoli, K.Bradshaw}@ru.ac.za

Keywords: IPTV, RTSP Proxy, IPTV media function, IPTV services

Received: October 15, 2011

Multimedia content delivery in IMS, including IPTV, is handled by a separate unit, the Media Function (MF), made up of media control and media delivery units, which in the case of IPTV are the Media Control Function (MCF) and Media Delivery Function (MDF), respectively. According to the different specifications of an IMS based IPTV architecture, the User Equipment (UE) is expected to use the RTSP protocol as a media control protocol to interact with the MCF, and obtains delivery of media from the MDF using the RTP protocol. This also means that the streaming session needs to be initiated from the media controller on behalf of the user but the delivery of media is sent to the UE from the media deliverer (media server). Due both to the lack of free and open source Media Servers and the availability of free and open source Streaming Servers, the ideal choice for the delivery of multimedia services, including IPTV, by the research community is Streaming Servers. Nevertheless, because of denial of service attacks and other issues, most streaming servers do not allow a different location for the session setup request and the delivery of media in the streaming session. In other words, most streaming servers are not designed to be controlled by some other entity other than the RTSP client that consumes the media. This makes it difficult to have a separate media control unit for IPTV service in IMS if one wanted to use a streaming server as an MDF unit. So, while waiting for streaming servers to work in this manner, it is better to find a work around in order to use streaming servers to develop and test IPTV services in IMS environments. For this purpose we propose another component (an RTSP proxy and relay unit) as part of the IPTV MF and to mediate between the MCF and MDF. This unit correctly relays media control commands from the MCF to the MDF and RTP packets from the MDF to the UE. It also helps in the implementation of other streaming functionalities that are required for IPTV service delivery, but which are not implemented in the current open source streaming servers. Additional services can also be easily implemented with the help of this unit. This will facilitate the development of an IPTV service using readily available open source streaming servers and help researchers to evaluate their proposals on new services they would like to develop. In this paper we show how this RTSP proxy unit can be integrated into the Media Function of the IPTV architecture to ease the media delivery process of an IMS based IPTV service.

Povzetek: Članek predstavi posredniški strežnik za televizijo IP, ki uporablja standard RTSP.

1 Introduction

The popularity of YouTube and other Internet based video services shows the potential of these services in the telecom world[1]. Companies involved in video service delivery are reaping the benefits of this huge demand. According to [2], the growth of on-line video spending surpassed $2.12 billion in 2008, up 36% from 2007 and has been forecast to continue double-digit increases through the years to come. A recent report on "The Mobile TV Market" from the ABI Research Group also revealed that the mobile TV market has tremendous long-term promise as a next-generation infotainment experience and will grow to a value of more than $50 billion by 2013 [3]. On the other hand, users are moving beyond viewing short, low-quality clips of user-generated content on YouTube and increasingly seeking out TV shows, films, and other professionally created, high-quality video content. Nevertheless, because of the inherent characteristics of the Internet, quality of service (QoS) cannot be guaranteed with Internet based services, and here lies the advantage for Telecos to engage themselves in the delivery of video oriented services. As video is one service that requires large bandwidth, users will be even keener if they can obtain the service with the required quality of service. IMS (IP Multimedia System), as an implementation of NGN (Next Generation Networks), provides the required QoS for users and is the right environment in which to deliver the IPTV service.

[1] This work is being carried out in the Distributed Multimedia Centre of Excellence at Rhodes University, with financial support from Telkom, Stortech, Tellabs, Amatole Telecom Services, Bright Ideas 39, and THRIP.

Apart from granting users the ability to access their services using different devices and access technologies, the major goal of IMS is the delivery of multimedia services, like IPTV. There are different proposals of implementation standards for IMS by different standard bodies, each with particular emphasis on a specific service type. The major ones include: the 3rd Generation Partnership Project (3GPP) [4], European Telecommunications Standards Institute (ETSI) Telecommunications and Internet Converged Services and Protocols for Advanced Networking European Telecommunications Standards Institute (TISPAN) [5], and the Telecommunication Standardization Sector of International Telecommunications Union (*ITU*-T) [6]. Similarly, there are also different specifications for the delivery of IPTV, again from different bodies. However, the specification that has received the greatest interest from the research community for the development and testing of IPTV services is the one proposed by ETSI-TISPAN [7]. As such, we have adopted their standard in this paper.

Multimedia session delivery involves the use of a session control protocol to control the session and a media control protocol to control the media delivery. The media delivery in a standard IMS architecture, for example, is carried out by what is known as the Media Resource Function (MRF), consisting of two distinct parts, namely the MRFC (MRF Controller) and MRFP (MRF Processor). The IPTV specification also has a similar component for media delivery and control, which is called IPTV Media Function (MF). The control unit of the IPTV MF is the Media Control Function (MCF) and the delivery unit is the Media Delivery Function (MDF). The media delivery unit, MDF, is supposed to be implemented by a fully-fledged Media Sever. However, because of the lack of free and open source media servers, researchers mostly use open source streaming servers to develop and test media services.

In addition to the media delivery and control units, IMS services, like IPTV, are controlled by a service controller unit, which in the case of IPTV is the IPTV Service Control Function (SCF). Basically, this unit is a SIP application server (AS). So, if a user knows the service description of a given IPTV service, s/he contacts the SCF to obtain the desired service. The SCF, in turn, will contact the MCF to initiate the delivery of media. The MCF then initiates the media delivery by instructing the MDF to send the requested stream directly to the user. In general, the MCF initiates the media request on behalf of the user and the media server, MDF, delivers the stream to the user (not to the initiator of the session). As all media requests pass through the MCF, this means that if one were to follow the specification directly, all media requests including session initiation should pass through MCF.

As mentioned above, open source streaming servers are being used for the delivery of streaming media by IPTV researchers. However, most streaming servers do not allow the delivery of media to a destination that is not the client that initiated the streaming session. For this reason, researchers tend to combine the MCF and MDF units into one unit (the streaming server) and initiate the media delivery and control from the UE, instead of from the MCF. The most popular open source IPTV application server used by the research community is the one developed by researchers at the University of Cape Town (UCT) [8]. Because of the problems with the current open source streaming servers mentioned above, the application server has been developed in such a way that the UE sends media requests directly to the streaming server without any involvement by the MCF (which is contrary to the specification). The UCT IPTV AS has become the de facto standard for developing IPTV services by the research community. Nevertheless, because the UE directly contacts the streaming server, neither the AS nor the MCF has control over the session and thus, it would be difficult to develop services that involve media session control. The Convergence Research Group at Rhodes University also makes use of the UCT IMS client [8] to test IPTV services and has followed this approach all along. This, however, is not in accordance with the specification of the IPTV service. Actually, not only does it contradict the specification, but it also does not allow the control of media to be done by a controlling unit because the MCF is not involved in the media setup process. A recent article by researchers from UCT [25] referred to the development of a media server that can work with the MCF, but it is currently not available to the research community. As a result, a work around is required if streaming servers are to be used for media delivery.

In this paper, we describe in detail how including the new lightweight component introduced in [1], that is, the Streaming Server Proxy and Relay (SSPR) unit, in the IPTV MF can help a service developer to use streaming servers and also assist the development of advanced IPTV services. The paper also introduces new functionalities such as “media switching” and “bookmarking” that are included in the proxy. This new unit is integrated into the MF to overcome the problem mentioned above. The remainder of this paper is organized as follows. Section 2 gives background information. Section 3 describes related works, while Section 4 explains the new proposed architecture of an IPTV service. Section 5 presents implementation and discussion, and finally, Section 6 gives our conclusions and future work.

2 Background

As mentioned in the Introduction, the IPTV architecture proposed by ETSI-TISPAN is the one followed by many researchers. We also use this architecture to present and discuss our proposal. The major components of this architecture are: Service Discovery and Selection Functions (SDF and SSF, respectively), Service Controller Function, and Media Control and Delivery Functions. Figure 1 shows these functional units of the IMS-based IPTV architecture as proposed by the ETSI-TISPAN standards body.

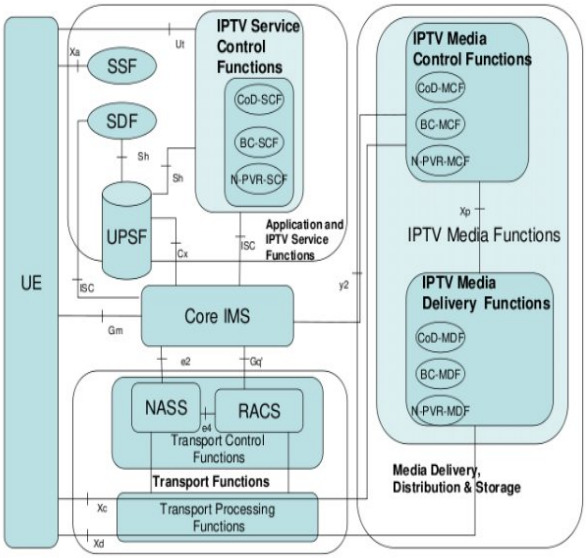

Figure 1: Functional Architecture of IPTV.

Table 1 gives the protocols used by the different reference points or interfaces.

The specification also makes it clear that the UE uses RTSP commands to communicate with the MCF for the purpose of media control. With regard to setting up media for an IPTV service, the specification specifies two methods for media initiation, referred to as Method 1 and Method 2. The distinction relates to where the RTSP session setup commands (specifically DESCRIBE and SETUP) are initiated from. In the Method 1 session setup technique, the session is initiated and the setup originates from the MCF. However, in Method 2, it is initiated by the UE, but the UE sends all RTSP commands (including session initiation and setup) to the MCF and not to the MDF. So even with the media initiation of Method 2, the media initiation request should be sent to the MCF.

Interface	Protocol used
ISC	SIP/SDP
X_d	RTP/RTCP
X_a	HTTP/DVBSTP or Flute
sh, C_x	Diameter
X_c	RTSP
U_t	HTTP
X_p	not defined

Table 1: Protocols used on reference points (adapted from ETSI TS 183063 V3.5.2).

In the following, we describe the steps taken by the UE to access an IPTV service using Method 2:

- The UE, like any IMS client, has to register with the IMS core before it attempts to request any service. After registration, the first step in accessing the IPTV service is the identification and selection of the service that the user desires. This is done by contacting the Service Discovery and Selection Function (SDF and SSF, respectively) units.
- Service discovery, also called service attachment, is accomplished by contacting the SDF, which provides information about the user's IPTV services and where the user can select the services. Basically, this is information about the address of the service server or portal that will provide the user with a description of the available service. In general the service attachment information consists of SSF addresses in the form of URIs and/or IP addresses.
- Once the UE obtains the service description, it contacts the SSF to retrieve relevant information about the IPTV service, like the URL of the media (content identifier), to initiate the IPTV session.
- After a service has been selected, the relevant content identifier is inserted in the SIP session initiation message sent to the IPTV Service Control Function (SCF) that provides access to this service. The UE does this by sending an INVITE request to the IMS core.
- The IMS core then forwards the request to the SCF that is responsible for controlling the service.
- The SCF then performs service authorization and credit control, selects the relevant IPTV media function, and forwards the request to the MCF that is responsible for controlling the media for this particular user.
- The MCF is responsible for initiating the media session by contacting the MDF that is supposed to serve the particular user. Once the media is set up correctly, the MCF notifies the UE of the status of the media session and the UE can send the RTSP PLAY media control command to start the media session.
- The delivery of media then starts from the MDF to the UE.

All media control commands from the UE are sent to the MCF, which then forwards the request to the MDF using the media control protocol. The media control protocol that the MCF uses to control the MDF is not specified in the specification and it is up to the implementers to choose an appropriate protocol.

As mentioned before, both media initiation (SETUP) and other media control commands are handled by the MCF, but the media is sent directly to the UE through the Xd reference point (see Fig. 1). For the MDF all media initiation requests come from the MCF. This also implies that the MDF should be able to handle media requests from a different location other than the UE and deliver the media to the UE.

The RTSP protocol [9] specifies a *"destination"* parameter that needs to be used in the transport section of a SETUP request to set up the destination of the media. The different versions of RTSP specification refer to this parameter by different names. Version 1.0, for example, specifies it as "destination", while version 2.0 of the RTSP protocol specification, which is an Internet draft, specifies it as *dest_addr* [10]. This parameter (field) needs to be included in the transport section of each SETUP request for which a different destination is needed. If the server supports this feature, it then sends the media to the specified destination when the media delivery begins, but continues to send the RTSP responses to the location that initiated the RTSP session. This could have been used by the MCF to initiate an

RTSP session from streaming servers on behalf of the user; however, based on our investigation of the available open source streaming servers, VLC [11], Darwin Streaming Server (DSS) [12], and the Mobicents Streaming Server [13] do not support the use of this parameter. Darwin returns an "Invalid Code" error code, while VLC and the Mobicents Streaming Server just ignore it and continue sending the media to the media session initiator. Live555 [14], on the other hand, allows the use of this parameter (using the name "destination") by modifying the RTSPServer.cpp file. Because of the possibility of denial of service attacks, this feature is disabled by default but can be enabled by inserting a "#define RTSP_ALLOW_CLIENT_DESTINATION_SETTING 1" statement at the beginning of the above mentioned file. Nevertheless, Live555 only plays video files that are encoded with the MPEG video codec, which is not supported by most UEs because it is not the default codec suggested by the IPTV specification. Consequently, we cannot make use of this media server either and the only option left to enable the use of a streaming server as the MDF is to include an RTSP proxy. The work presented in this paper aims to solve this problem.

The problem with open source streaming servers is not only related to the support for "destination" parameter, but also there are other features that IPTV services require, but the current open source streaming servers do not support. A bookmarking service, for example, requires that the current position of the media that is being played be recorded and kept together with detailed media information. As a result, the application server needs to request the media server to obtain this information in order to store bookmark information of the media that is being played. The RTSP protocol has a command that can be used for this purpose. The specification defines a *get_parameter* command for the purpose of querying a streaming server to obtain media related information including the current play time. Specifically one can use this command together with a *range* parameter to obtain the current media position. In fact, the Open IPTV Forum (OIPF) also suggests the use of this parameter for the purpose of bookmarking. However, both VLC and DSS do not support this. DSS responds with a 500 error code, while VLC again merely ignores it. Consequently, we have implemented this functionality in the proxy, with the details given in the Implementation section.

3 Related Work

Various researches have been carried out on the media processing aspects of IPTV, particularly with regard to the type of media control protocol to be used for IPTV. In this regard, an evaluation of SIP for the use of streaming control instead of the RTSP protocol has been presented in different IETF Internet drafts [15][16][17]. Taking this idea a bit further, various researchers have also reported their experiences with regard to implementing SIP as a media control protocol. The authors in [18] showed how a new SIP header (called SIP-MEX) and new SIP bodies (an XML document in the SIP INFO message) can be used to send media control commands to the MCF. On the other hand, other researchers have also suggested the integration of SIP and RTSP to create a comprehensive media control protocol [19]. However, as mentioned in [20], to avoid the IMS signaling procedures causing extra delays, it is always necessary to define a clear separation between service/session control performed at IMS level and media flow control handled end-to-end between user equipment and the content service. Actually, this could be one of the reasons that ETSI-TISPAN proposed a different media control protocol other than SIP in the standard specification. In general, those who have proposed SIP as a media control protocol have tried to justify their proposal from the point of view of media control requirements that cannot be handled by RTSP and also for the purpose of handling bandwidth reservation requests and responses. Nevertheless, as to the support of bandwidth negotiation, the IETF has developed extensions to SDP [21] and it should no longer be a problem to use RTSP. In general, both approaches have their own strengths and weaknesses, but these are not considered in detail here, as this is beyond the scope of this paper. However, the advantage of using RTSP as a media control protocol is that the MCF is not required to translate media control commands received from the UE when it forwards them to the MDF.

On the other hand, with regard to having a separate MCF and MDF, some researchers have also proposed the integration of the service selection function with the media function. In [22], for example, the authors proposed a comprehensive service function, called the Multimedia Service Control Function (MSCF). The MSCF combines the functionality of the SDF, SSF and MF functions of the IMS based IPTV units. The authors also proposed a Media Distribution Function consisting of three components, namely, Interconnection (similar to the IPTV MDF), Serving (IPTV MDF), and Primary (IPTV MDF), abbreviated as I-IMDF, S-IMDF, and P-IMDF, respectively. According to the authors, the function of the P-IMDF is to serve as the primary contact point, and also to handle the streaming function.

The concept of an RTSP gateway is also presented in [23], where the authors proposed a gateway that converts RTSP messages to SIP messages and vice versa. On the other hand, the use of an RTSP proxy for the delivery of streaming service for UEs without RTP support is presented in [24].

As mentioned before, researchers tend to use streaming servers for media delivery in IPTV services. However, with regard to the implementation of a proper IPTV media function, Ref. [25] discusses an initiative for the development of the UCT IPTV testbed and mentions the current work on the MCF and MDF. The authors have not, however, clearly specified what media control protocol they used, nor explained how the MDF is implemented. The UCT IPTV client and AS are very popular open source IMS components in the research community, but this particular project was new and not

available at the time of conducting the research presented in this paper. As a result, until the issue of the media control protocol settles down, and an open source IPTV MF is commonly available to the research community, we hope that our proposal will be helpful to researchers wishing to develop an IPTV service using streaming servers particularly as it conforms to the specification. In fact, the proxy also implements new functionalities such as easy media switching and bookmarking services for use by service developers.

4 Proposed Architecture

The aim of this paper is to describe how streaming servers can be used as an MDF unit by incorporating the proxy explained below.

The proposed architecture is basically the same as the TISPAN architecture presented in Fig. 1, except that the SSPR unit is added within the MF. Thus, the focus of this section is on the MF unit.

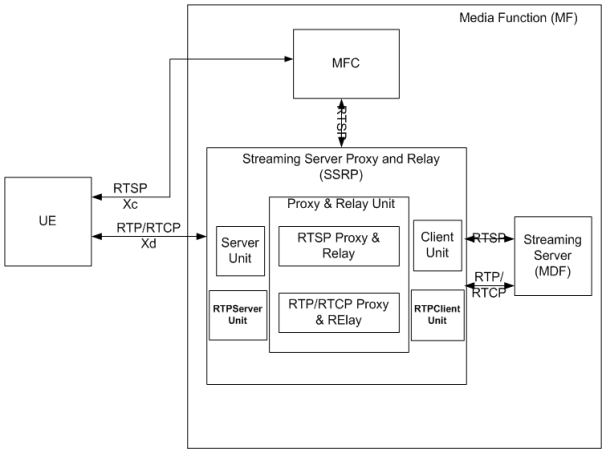

Figure 2: Block diagram of the modified Media Function.

As can be seen from Figure 2, which shows a block diagram of the modified Media Function, the SSPR has five main components or units: the proxy and relay, server, and client units. The server and client units are responsible for handling RTSP traffic. The proxy and relay unit also has two distinct components: the RTSP proxy and relay, and the RTP/RTCP proxy and relay units. We have also RTPServer and RTPClient units which are responsible for relaying RTP/RTCP packets from the server to the client and also back to the server. The following paragraphs briefly describe the function of each of these units.

- The server unit handles all RTSP requests coming from the MCF. Upon receipt of an RTSP request, it forwards the request to the proxy and relay unit, which is responsible for forwarding the request to the client unit.
- The client unit acts like an RTSP client to the streaming server and sends the request that it receives from the proxy and relay unit to the streaming server, and also forwards the response it receives from the server back to the proxy and relay unit.
- The RTPServer unit sends RTP/RTCP packets to the client and also sends RTCP packets back from the client to the server unit. It communicates with the proxy and relay unit to do this.
- The RTPClient unit relays RTP/RTCP packets that come from the streaming server to the client through the proxy and relay unit. It also forwards RTCP packets that come from the client to the server unit.
- The proxy and relay unit is responsible for forwarding requests from the server units to the client unit and also forwards responses from the server unit to the client unit. It also relays RTP/RTCP packets from the streaming server to the client and vice versa. The proxy and relay unit must change the request that comes from the client (MCF) so that the streaming server can return the responses and media delivery to it. For this purpose, it records the address information of the client and generates or uses its own address before forwarding the request to the client unit. It does the same thing when forwarding responses that come from the streaming server back to the client (MCF).

Basically, for the streaming server (MDF), the request comes from the SSPR and the response is also sent back to the SSPR. As a result, the problems mentioned in the previous section do not arise in this scenario. The SSPR

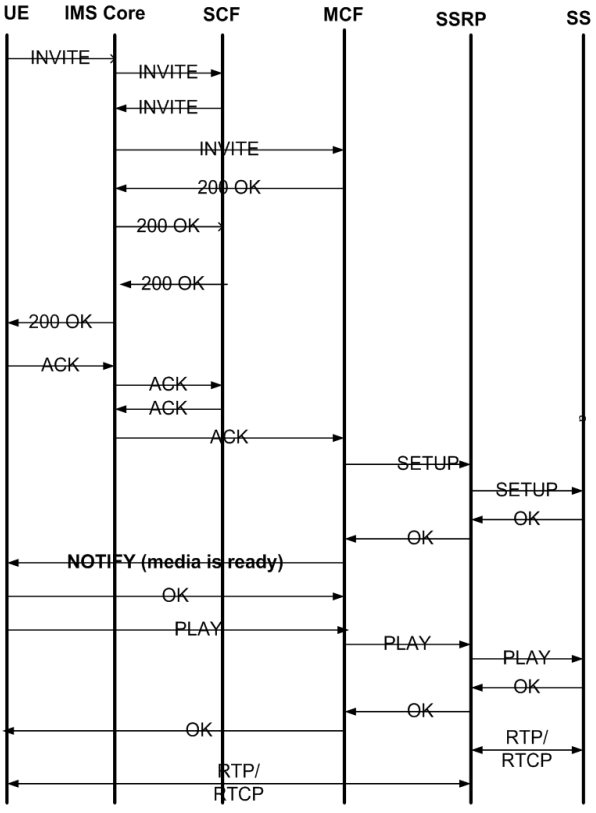

Figure 3: IPTV service access in the proposed architecture (using media access according to Method 1).

is designed to manage streaming sessions and can also handle the proper proxy and relay functions to process stream control commands and deliver the stream. The flow diagram in Fig. 3 shows the IPTV service initiation and access using this architecture.

5 Implementation and Discussion

5.1 Implementation

The Open IMS Core testbed from the *Fraunhofer Institute* FOKUS [26] is the most popular IMS testbed in the research community. The Convergence Research Group at Rhodes University has been using this testbed to develop and test IPTV services. We also used this testbed to test the functionality of the proxy. Regarding an IMS user agent, we used the UCT IMS client, described earlier. The client is designed to work with Method 2 of the IPTV media access methods. As a result, to avoid excessive work on the client side we used the client with this setting. The client has the capability of sending all RTSP commands.

Since there is no open source Service Discovery and Selection component, we learn from others' experiences and used the technique provided in [27] to deliver the URL of the media to the UE. As mentioned in the paper, the AS upon receipt of an INVITE from UE, includes the URL of the media in the SIP OK message that is sent to the UE. Although we adopted this technique, in our case we transferred this functionality to the MCF, instead of the AS. As a result, as can be seen in Fig. 4, upon receipt of an INVITE from the UE, the AS forwards the INVITE to the MCF and the MCF then matches the requested channel to a URL and sends it to the UE including the URL in the SIP OK message. If there is no MCF, the approach taken is that after establishing the SIP session with the AS, the UE then sends the DESCRIBE and SETUP commands to the streaming server to initiate and set up an IPTV media session. However, this time around, the client sends these commands to the MCF instead of the streaming server. When the MCF gets the RTSP command from the UE, it forwards the request to the RTSP proxy. The MCF uses the *destination* parameter of the RTSP protocol, discussed in an earlier section, to pass on the destination of the media, i.e., the address of the UE. This parameter is included in the SETUP command of the request. The MCF obtains client address information from the SDP payload of the SIP INVITE command. Accordingly, the proxy forwards the request to the streaming server and upon receipt of the media, delivers it to the UE. The proxy also uses a configuration file to obtain information about the streaming server, such as its address and port.

The proxy has different handlers on the client and server sides for both the RTSP and RTP/RTCP sessions. On the client side, the RTSP session is created with the MCF while the RTP/RTCP session is created with the client (UE). On the streaming server side, both sessions (RTSP and RTP/RTCP) are created with the streaming server.

Session handling is one important aspect of media servers. An RTSP session initiation request (for example, DESCRIBE) may not necessarily end up in an RTSP session. The client may not be able to play the media (video) if it does not support the codec that the media is encoded in. As a result, even though there is an I/O (network) session between client and server, an RTSP session is basically created when the client sends a SETUP request to the server. This tells the server that the client can play the media and the server generates and sends a unique session ID within the response. Both the client and server use this id to refer to the session in subsequent communication. The proxy is also designed in a similar manner. As is the case with any proxy system, a session that is supposed to be established between a client and a server is divided into two sessions: one on the client side and another on the server side. Similarly, if we consider the RTSP session, we have two separate RTSP sessions (one on the client side – MCF to proxy's client side and another from the server side – proxy's server side to streaming server). Similar to the RTSP principle of establishing an RTSP session mentioned above, a proxy session object is created when the client sends a SETUP request to the server. The proxy then creates a unique random session ID for the client side RTSP session and records it in the proxy session object that it created for this particular session. When a response comes from the server with the server's session ID, that session ID is also recorded in the proxy

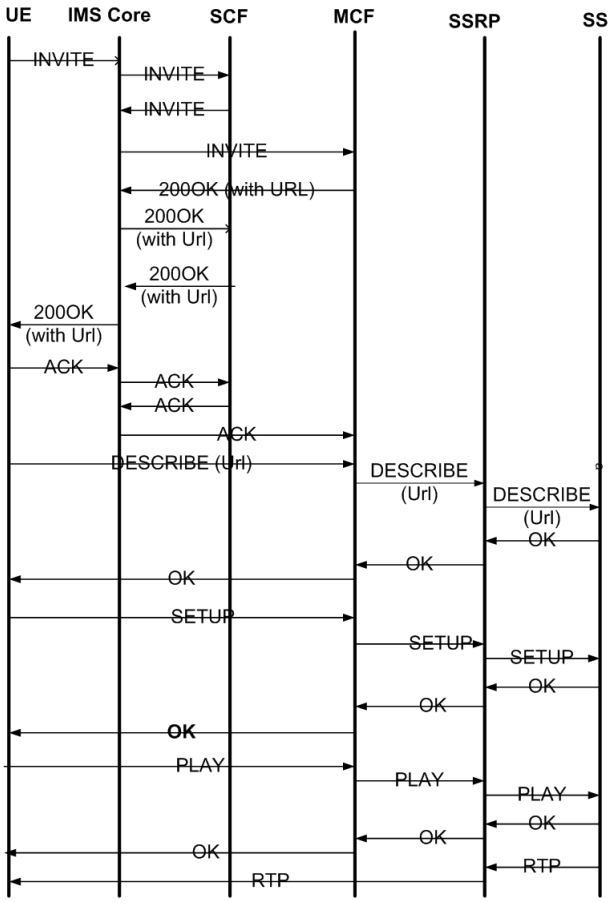

Figure 4: IPTV service access in the proposed architecture (using media access according to Method 2).

session object. In other words, the proxy session contains the client and server RTSP sessions. The proxy session object also contains a track list.

Multimedia sessions may contain more than one media (track). For this purpose we also defined an object called a Track object that contains detailed information about a track, including client address, server address, client RTP/RTCP ports, server RTP/RTCP ports, and client and server RTP/RTCP sessions, which are useful for forwarding requests and responses from the client to the server and vice versa. The track object has methods to forward requests to the server and responses to the client. The track class contains a hash map of the different track objects related to different sessions. As a result, a particular track object is identified using the session id of a request or response.

In a similar way to RTSP sessions, RTP sessions are also identified by unique ids called the SSRC (Synchronization Source) identifier. As a result, the proxy also creates a "proxy SSRC" id to identify the RTP/RTCP session between the proxy and the client (MCF). The server sends its own SSRC id when it starts sending RTP packets. The RTPClient unit is responsible for handling the RTP/RTCP packets that come from the server and forwards them to the client (MCF) through the proxy and relay unit. Similarly the RTCPServer unit handles the relay of RTP and RTCP packets to the MCF. The proxy modifies the SSRC id before forwarding the RTP/RTCP packets to the MCF. For the client (MCF), the RTP/RTCP packets come from the proxy.

The ProxySession class also has a hash table to match client and server RTP/RTCP sessions together with the proxy object mentioned before. In general, RTSP sessions are identified using the "Session ID" and RTP/RTCP sessions are identified using the "SSRC id".

Advantages of media sessions being handled by the proxy can be seen in the creation of session related services. For example, if a media switch is requested, an efficient way of doing this would be to use the existing connection on the client side and create a new connection on the server side. One advantage could be to continue feeding media to the user until the new media setup is ready on the server side. Another advantage is the reduction of processing time because there is no session setup on the client side. In addition to this, interesting services like "Switch with Pause" can be developed with this type of approach. Switch with Pause involves pausing the current media and switching to the new media. Once the new media finishes, the proxy can resume playing the previous media. In general, using this technique the proxy creates different RTSP and RTP/RTCP sessions for the new media on the server side by sending and receiving the RTSP requests/responses itself automatically until the media setup is ready. When the new media setup has been completed, it uses the same session on the client side to deliver the media. In other words, a new connection is only created on the server side (from the proxy to the streaming server.)

Figure 5 shows a proxy session containing both active and paused sessions. The "media switch" command is defined and sent to the proxy using the RTSP OPTION command. This command is extended by defining a field named "switch" that can take parameters like "immediately" or "number of seconds" after which the switch is sought. The URL of the media to be switched is also included in the RTSP OPTION command.

We have also implemented a bookmarking feature in the proxy. The RTP protocol includes a feature whereby each packet contains a timestamp of the packet being delivered. The timestamp is the sampling frequency of the packet being delivered relative to a running clock. As a result, the timeframe of the first packet does not start from zero and it is always good to record the first timestamp as a reference point to obtain the relative position of future packets. So, having recorded the start time, we obtain the final time (when bookmarking is requested) and subtract the two to get the difference. We divide this by the clock rate of the media, which is included in the SDP of the media. This information is recorded in the Track class discussed in a previous section.

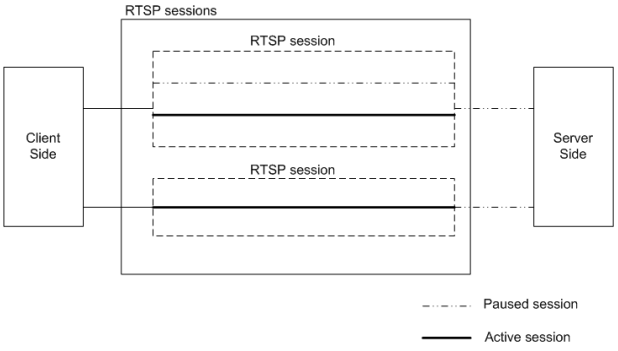

Figure 5: Proxy session handling.

As mentioned above, the RTSP proxy and relay component can be easily extended to include a variety of functions required by IPTV developers.

The RTSP proxy and relay is implemented using the Java programming language, and particularly the Apache MINA framework [28]. The MCF unit is implemented as a SIP AS and the RTSP proxy and relay is included as a separate class within the MCF and initiated from within the MCF.

5.2 Discussion

We tested the proxy on a dual Core Intel 2.66 GHz PC with 2 GB RAM. The machine also ran the streaming server. We used the Java System API (the nanoTime() method) to record the delay introduced by the proxy. The time was recorded when the proxy received a request and also when the same request left the proxy to the streaming server. The difference was calculated to determine the delay through the proxy. The same approach was used for the responses. The average delay introduced by the proxy was found to be negligible (close to 40 nano seconds). As a result, we believe that the proxy can be considered a good solution for those who wish to use a streaming server for media delivery in IMS based IPTV services, as it does not introduce much delay into the whole system.

As mentioned previously, among the many advantages of having a separate RTSP proxy in place is the ability to change media within an existing session (e.g., inserting an advert), which can be done easily without the involvement of the UE. This is important functionality because the UE does not require a different connection to obtain the new stream as the proxy can handle that itself. The AS can initiate the modification of media within an existing session based on different sets of rules and the proxy can deliver the modified stream without involving the UE. We believe this will help researchers to implement new and innovative services that can be implemented by any standard UE. Because of the nature of the RTSP protocol, in the event that the media source disappears for whatever reason, there is no way that the MCF can know about the situation. On the other hand, if we use a proxy, because the proxy also handles RTP packets, it knows if the session is alive or dead and can take the appropriate action immediately when a problem arises.

The proxy can also be easily extended to include other features. For example, a transcoder unit can be integrated into the proxy and it can used to carry out transcoding based on the capabilities of the UE, if such functionality is required.

6 Conclusion and Future Work

The IPTV research community mainly uses streaming servers for the delivery of media for IPTV services. On the other hand, the IMS-based IPTV specification specifies that a streaming session is initiated by the MCF on behalf of the user. Nevertheless, most open source streaming servers do not allow the initiation of a streaming session by a different client to the RTSP client intending to consume the stream. To overcome this limitation, a proxy as described in this paper, can be used as a work around thereby enabling researchers to use the available streaming servers while adhering to the standard. The RTSP proxy can be integrated into the MCF or be deployed as a separate entity. According to timing experiments conducted, the proxy does not introduce a significant delay to the service delivery process and as such the authors believe it to be a good solution. In addition to allowing the use of streaming servers as MDF units for the delivery of IPTV services, the paper also presented some of the additional benefits arising from use of the proxy.

As a future work we plan to include a transcoding capability in the proxy so that the stream can be transcoded or transrated on the fly based on devices' capabilities.

We also intend extending this work to include load balancing functionality in the proxy, so that the proxy can choose different streaming servers based on their status. The proxy will also be packaged as an API by abstracting the streaming servers and providing interfaces that service developers can use.

References

[1] Z. S. Shibeshi, A Terzoli, K Bradshaw (2010). Using an RTSP Proxy to implement the IPTV Media Function via a streaming server. In ICUMT'10: Proceedings of the International Congress on Ultra Modern Telecommunications and Control Systems. Moscow, Russia, 18 to 20 October 2010

[2] Accustream media research. Online Video Spend Tops $2.12B in 2008. http://www.marketingcharts.com/interactive/online-video-spend-tops-212b-in-2008-225-growth-forecast-in-2009-7955/. Accessed: July 1, 2010.

[3] ABI Research – Technology Research Market. http://www.abiresearch.com/home.jsp. Accessed on July 1, 2010.

[4] The 3rd Generation Partnership Project. http://www.3gpp.org/

[5] ETSI Telecommunications and Internet converged Services and Protocols for Advanced Networking (ETSI- TISPAN). http://www.etsi.org/tispan/

[6] The Telecommunication Standardization Sector of ITU (ITU-T). http://www.itu.int/ITU-T/

[7] ETSI TS 182 027: IPTV Architecture;IPTV functions supported by the IMS subsystem, March 2011.

[8] The UCT IMS Client. http://uctimsclient.berlios.de/. Accessed November 15, 2011.

[9] H. Schulzrinne, A. Rao, and R. Lanphier (1998). Real Time Streaming Protocol (RTSP). http://tools.ietf.org/html/rfc2326

[10] Internet draft Real Time Streaming Protocol 2.0 (RTSP). .http://tools.ietf.org/html/draft-ietf-mmusic-rfcrfc2326bis-28. . Expires: April 30, 2012.

[11] VLC. VideoLAN, Free streaming and multimedia solutions for all OS. http://www.videolan.org/. Accessed: November 15, 2011.

[12] Darwin, "Open Source Streaming Server," http://developer.apple.com/opensource/

[13] Mobicents-Public. http://groups.google.com/group/mobicents-public/web

[14] Live555. Internet Streaming Media, Wireless, and Multicast technology, services, & standards. http://www.live555.com/

[15] Internet draft. Framework of media control for IPTV services draft-siva-iptv-media-01.txt. http://tools.ietf.org/html/draft-siva-iptv-media-01. . Expires: September 2010.

[16] Internet draft. An Evaluation of Session Initiation Protocol (SIP) for use in Streaming Media Applications draft-whitehead-sip-for-streaming-media-00.txt. http:// tools.ietf.org/html/draft-whitehead-sip-for-streaming-media-00. Expires: April, 2006.

[17] Internet draft. Media Playback Control Protocol Requirements draft-whitehead-mmusic-sip-for-streaming-media-03. http://tools.ietf.org/html/draft-whitehead-mmusic-sip-for-streaming-media-03. Expires: August 2008.

[18] S. Sivasothy, G. M. Myoung, and N. Crespi (2009). A unified session control protocol for IPTV services. In *Proceedings of the 11th International Conference on Advanced Communication Technology*. pp. 961-965, February 15-18, 2009, Gangwon-Do, South Korea.

[19] R. G. Shiroor (2007). IPTV and VoD services in the context of IMS. In *International Conference on IP Multimedia Subsystem Architecture and Applications*, pp. 1-5, December 6-8, 2007.

[20] B. Chatras, M. Saïd. Delivering Quadruple Play with IPTV over IMS. www.icin.biz/files/programmes/Session8A-1.pdf. *Accessed: July 1, 2010.*

[21] Internet draft. SDP media capabilities Negotiation draft-ietf-mmusic-sdp-media-capabilities-09. http://tools.ietf.org/html/draft-ietf-mmusic-sdp-media-capabilities-09. Expires: August 2010.

[22] E. Mikoczy. IMS based IPTV services: architecture and implementation. In *Proceedings of the 3rd International Conference on Mobile Multimedia Communications*. pp. 1-7. 2007. Brussels, Belgium.

[23] C. Riede, A. Al-Hezmi and T. Magedanz (2008). Session and media signaling for IPTV via IMS. In *Proceedings of the 1st International Conference on Mobile Wireless Middleware, Operating Systems, and Applications*. February 13 - 15, 2008. Brussels, Belgium.

[24] Z. Shibeshi, A. Terzoli, and K. Bradshaw (2010). Streaming Session Transfer between Registered User Agents. In *SATNAC'10: Proceedings of the 13th Southern African Telecommunications Networks and Applications Conference*. Spier Estate, Stellenbosch, South Africa, 5 to 8 September 2010.

[25] R. Spiers, R. Marston, R Good and N. Ventura (2009). The UCT IMS IPTV Initiative. In *Proceedings of the 2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies*. pp. 503-508. 2009.

[26] The Open Source IMS Project. http://www.openimscore.org/

[27] P. R. Wilson and N. Ventura (2009). A Direct Marketing Platform for IMS-Based IPTV. In *SATNAC'09: Proceedings of the 12th Southern African Telecommunications Networks and Applications Conference*. Ezulwini, Swaziland, 30 August to 2 September 2009.

[28] The Apache Mina Software Foundation. http://mina.apache.org/

Mr. Zelalem S. Shibeshi holds an MSc in Information Science, Diploma in Computer Science, and BSc in Physics, all from Addis Ababa University, Ethiopia, and is currently working towards his PhD in the Computer Science Department at Rhodes University.

Alfredo Terzoli is a Professor of Computer Science at Rhodes University, where he heads the Telkom Centre of Excellence in Distributed Multimedia. He is also Research Director of the Telkom Centre of Excellence in ICT for Development at the University of Fort Hare. His main areas of academic interest are converged telecommunication networks and ICT for development.

Dr. Karen Bradshaw is a Senior Lecturer in the Computer Science Department at Rhodes University. Her research interests lie in Distributed Systems and Parallel Programming.