İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

SIP – RTSP CONVERGENCE: RTSP-C

M.Sc. Thesis by
İbrahim BİLGİN, B.Sc.
(504041517)

Date of submission : **5 May 2008**

Date of defence examination: **9 June 2008**

Supervisor (Chairman): **Assist. Prof. Dr. Feza BUZLUCA**

Members of the Examining Committee **Prof.Dr. Muhittin GÖKMEN (İ.T.Ü)**

Assoc. Prof.Dr. Selçuk PAKER (İ.T.Ü)

JUNE 2008

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

SIP-RTSP YAKLAŞTIRMA MODELİ: RTSP-C

YÜKSEK LİSANS TEZİ

Müh. İbrahim BİLGİN
(504041517)

Tezin Enstitüye Verildiği Tarih : **5 Mayıs 2008**

Tezin Savunulduğu Tarih : **9 Haziran 2008**

Tez Danışmanı : **Yard.Doç.Dr. Feza BUZLUCA**

Diğer Jüri Üyeleri : **Prof.Dr. Muhittin GÖKMEN (İ.T.Ü)**

Doç.Dr. Selçuk PAKER (İ.T.Ü)

JUNE 2008

ACKNOWLEDGEMENT

I would like to express my deep appreciation and thanks for my advisor. He guided me on both scientific perspective and pratical approach through my study.

June 2008 İbrahim Bilgin

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

B2BUA **:** Back to Back User Agent
IDE : Integrated development Environment
IMS **:** IP Multimedia Subsystem
IPTV **:** Internet Protocol Television
PSTN : Public Switched telephony network
QoE : Quality Of Experiences
QoS : Quality Of Service
RTCP **:** Real Time Control Protocol
RTP **:** Real Time Transport Protocol
RTSP **:** Real Time Streaming Protocol
SCIM : Service Capability Interaction Manager
SIP **:** Session Initiation Protocol
UA : User Agent
VoD : Video On Demand

SIP-RTSP CONVERGENCE: RTSP-C

SUMMARY

In this study, using Session Initiation Protocol (SIP) as transport and placing Real Time Streaming Protocol (RTSP) capabilities in the SIP message body, a media control model has been introduced for Voice Over IP (VOIP) networks. With the recent developments on both VOIP and IPTV technologies, convergence of these two technologies become one of the most important steps in the evolution of telecommunication. This new convergence model, RTSP-C, targets the interoperability of the two leading protocols of each technology: SIP and RTSP. The new convergnece model also resolves some open points on media control request authentication and session presentation (SDP) exchange. This new model is also valid for NAT Traversal methods applicable to SIP while it lifts the necessity of NAT Traversal methods for RTSP. The major advancement this model provides is: it makes the media control method/state information available to SIP. By doing that, this model enables the development of new streaming based SIP services. In this project content a Video on Demand (VoD) system is developped to instantiate the new convergence model. The implementation validated the operability of RTSP-C convergence model. The comparison of the results with other models on literature showed that the model provided adequate solutions on the pre-determined problems.

SIP-RTSP YAKINLASTIRMASI: RTSP-C

ÖZET

Bu çalışma ile SIP protokolü iletimde kullanılarak ve RTSP protokolü yetenekleri SIP mesaj gövdesine yerleştirilerek VOIP ağlarında yeni bir medya kontrol modeli öne sürülmüştür. VOIP ve IPTV teknolojilerindeki en son gelismeşler bu iki teknolojinin buluşmasını itetişim istemlerinin evriminin en önemli adımlarından biri haline getirmiştir. RTSP-C olarak isimlendirilmiş olan bu yeni yakınlaşma modeli her iki teknolojinin lider iki protokolü olan SIP ve RTSP'nin bir arada çalışmasını sağlamayı hedefliyor. Bu yeni yakınlaşma modeli aynı zamanda medya kontrol isteklerinin asıllanması ve oturum sunum bilgisi (SDP) alış verişindeki bazı açık noktalara çözüm getirmektedir. Medya kontrol mesajlarının iletiminde taşıyıcı olarak SIP protokolünün kullanılması medya kontrol mesajlarının asıllanmasında SIP protokolüne ait altyapının kullanılmasını sağlamakta ve bu sayede asıllama işlemi IPTV içerik sağlayıcısına ulaşmadan hizmet sağlayıcının ağında gerçekleşmektedir. Bu yeni model RTSP protokolünün NAT geçirimine ait yöntemlere gereksinimini ortadan kaldırmakla beraber, SIP protokolünün NAT geçirim yöntemleri geçerliliğini korumaktadır. Bu modelin sağladığı asıl gelişme medya kontrolü bilgisi ve durum bilgisini SIP protokolüne açık hale getirmesidir. Bu sayede bu model medya yayınına dayalı yeni SIP servislerinin geliştirilmesine olanak sağlamaktadır. Bu proje kapsamında ortaya koyulan yeni modeli örneklendirmek amacıyla bir İsteğe Bağlı Görüntü Yayını (VoD) sistemi geliştirilmiştir. Bu uygulama ile RTSP-C yakınlaştırma modelinin çalışabilirliği doğrolanmıştır. Sonuçlar literatürdeki diğer örnekler ile karşılaştırıldığında modelin daha onceden belirlenen sorunlara uygun çözümleri sağladığı görülmüştür.

1. INTRODUCTION

1.1 Purpose of the Thesis

The ongoing adoption of IP technologies on both the applications and networking level has evolved the Internet for delivering multimedia streaming services ranging from classical video telephony up to interactive IPTV services as well as other media rich services. The "Triple Play" is the new buzzword describing the convergence of the three terms: "voice (telephony), internet and TV as commercial notation for driving market rather than a new technology [1].

Several approaches have already been introduced struggling with the need for an integrated solution with the core IMS network or the development of a separated subsystem for next generation IPTV services [1]. The main focus for the integrated solution lies on the difficulties in using the Session Initiation Protocol (SIP) or SIP in combination with the Real Time Streaming Protocol (RTSP) for both the signaling and media control including so called Trick Functions that allow to manipulate media delivery by e.g. pausing and fast forwarding the content [1].

The main objective of this thesis is to suggest a media control model for SIP based IPTV applications. Main area of interest in this project is Video on Demand (VoD) which defines providing video based services through internet. However, with further work, the control model can be adapted to other IPTV applications. The protocol suggested, RTSP-C as "RTSP Compact Mode" enables a convergence between SIP and RTSP; provides a stronger framework to SIP based IPTV on the areas below:

- Transmission of media-control
- Authentication of media-control
- Streaming based network services.

1.2 Literature Work

There has already been work on SIP – RTSP interworking. [1] introduces RTSP streams to be described in SDP and to be established using SDP Offer/Answer model. [2] focuses on usage of SIP in multimedia streaming. Initial version (Version 00) of [2] introduces 2 main solution models to achieve it:

All SIP Solution: Relies on existing functionalities of Sip protocol to achieve media streaming control.

Dual Stack Solution: SIP is used at the establisment of the stream and further media streaming control is done via RTSP. Therefore clients and media servers should have both SIP and RTSP stacks.

Version 02 of [2] still hightlights the dual stack solution. Both drafts are expired on April 2007. Latest version, version 03, of the draft [3] is released on 2008 and highlights the requirements of the desred media control protocol.

Reference [3] focuses on IMS network interworking and how the SIP network can be aware of media control commands. The proposed solution is triggering SIP NOTIFY messages each time a media control media control message is issued.

[4] Focuses on blending IPTV services with system and introduces the concept of Service Blending, which means different applications controlling each other: like web or phone controlling a TV application.

The solution suggestions to interwork SIP and RTSP can be refined to 4 categories which are covered deeper in the following sections:

- All SIP solution
- Dual stack solution
- RTSP event notification to SIP
- Service Blending

1.2.1 All SIP Solution

1.2.1.1. Extensions to SIP Protocol

[5] defines the rules and suggestions to extend SIP protocol. The suggested method is the usage of "P-" headers which stands for "preliminary", "private" or "proprietary". The standard also defines the steps required for the usage of any new header.

Some example usages of "P-" headers are:

"P-Asserted-Identity" and "P-Preferred -Party" headers defined in [6] to carry private identity,

"P-Answer-State" header defined in [7] for Push-to-Talk implementations,

"P-User-Database" header used in 3GPP to identify the database address of the user profile and defined in informative RFC [8].

As a seperate protocol RTSP has quite many headers to introduce to be introduced to SIP. Also, RTSP includes media level controls. Both SIP and RTSP protocols use message body for media level control as they operate on session/presentation level.

Using "P-" headers in SIP would partially satisfy RSTP requirements. They can especially be used to transmit RTSP URL on setup time. However, mid-call signallling to introduce trick-play operations is another work item. Work load in introducing each "P-" header that carry information is to be considered. One other open item is the transmit of media level controls.

1.2.1.2. Extensions to SIP/TEL URI

SIP Protocol makes use of SIP and TEL URI schemes to address the endpoints. TEL URI scheme is defined in [9] to describe resources identified by telephone numbers. As one popular usage of SIP protocol is to be an edge protocol to interwork Legacy TDM systems, extensions to SIP is required to support TDM/ISDN based features over SIP. One method is to extend tel URI: add new header field to conform the requirements. This new fields can also be carried to SIP URL: the related conversion between SIP and TEL URL schemes is defined in [10].

Some example usages of SIP/TEL URL extensions:

"isub" field which is used to address agents behind PBX and defined again in [9],

“phone-context” parameter which is used to address validity domain of local telefon numbers (again defined in [9]),

“tgrp” and “trunk-context” fields defined to adress transit trunk and carrier in a telephony network and defined in [11].

Using SIP or TEL URI to transmit information provides limited space as the information must be transmitted in URL fields of Request URI, to, from or contact headers. As is “isub” (ISDN Subaddress) implementation defined in [9] SIP or TEL URI can be used to addess non-SIP resources. Therefore, it can act as a mechanism to address media resources like RTSP URL.

1.2.1.3. Mime Body Approach

SIP supports transmission of different mime body types [10], other than the most common one SDP. “Content-type” header specifies the type of the mime body transmitted while “Content-Encoding” header informs if there is any special encoding applied to the content.

[10] also states SIP support for multipart bodies, which is defined in [12]. Using multipart-mime mechanism, SIP protocol messages can be used to transmit mode than one content at a time, for example: SDP information and text at the same time.

With capability to transmit different types of content, SIP has usage to tunnel non-SIP protocols. This usage is especially favoured to carry TDM and ISDN protocol messages. Examples of this usage are:

SIP-T (for SIP Telephony) which which encapsulates ISUP protocol messages in a mime body and transmits using multipart-mime body support of SIP. The standard is introduced via [13] .

[14] registers application/isup and application/qsig mime types to IANA, in order to enable tunneling of ISUP and QSIG protocols over SIP.

On other work on this subject is the introduction of SIP INFO method via [15] . The intent of the INFO method is to allow for the carrying of session related control information that is generated during a session. The INFO method is used for the carrying of mid-call signaling information along the session signaling path.

Some potential uses of INFO method listed in [15]:

- Carrying mid-call PSTN signaling messages between PSTN gateways.
- Carrying DTMF digits generated during a SIP session.
- Carrying wireless signal strength information in support of wireless mobility applications.
- Carrying account balance information.
- Carrying images or other non streaming information between the participants of a session.

As soon as the mime-type is registered as a standard, rtsp related information can be carried as a SIP message body without any modification to the existing SIP protocol. Any information on the existing RTSP protocol can be carried directly while re-formatting the information to reduce the message size is also another alternative. As SIP and RTSP has some common attributes, it is possible to remove some of the headers from RTSP message content. While stream setup can be implemented in paralll to call setup negotiation over SIP (by using multipart-mime bodies); mid-call RTSP signalling like trick-plays can be implemented using SIP INFO message.

1.2.2 Dual Stack Solution

Dual Stack approach is the rather straightforward one which requires the usage of SIP for session setup and RTSP for streaming control. This way, the two protocols are kept in two different layers of control, seamingly independent from each other (there is still need for SIP to initiate the media session for RTSP to control the session.).

The solution is originally highlighted by [2]. Though the approach is straight forward, since the two protocols share some similar functionalities, further work is required to ensure synchronization of these two protocols.

1.2.3 RTSP Event Notification to SIP

This approach is suggested by [3] as the way of SIP and RTSP interworking on IMS. The method can be presented as tracking or “snooping” video control messages (or a subset thereof) and use SIP Notify method to inform the IMS framework of the c urrent state of the useris video session (what they are watching, whether the session is paused, etc.).

Such functionality is commonplace for IGMP, and it is therefore relatively straightforward to generate a companion SIP message that describes the changing user state (e.g. User X watching Channel Y). RTSP snooping is more complex, due to the comparative richness of this protocol, and as a result greater packet processing capability is required. However, a similar methodology can be used in RTSP.

One other aspect of the method is the creation of RTSP event package. The requirements of an event package is defined in [16]. Finally this event package should be registered with IANA (Internet Assigned Numbers Authority). [16] also presents a guideline to register such event packages.

The subscription method can be explicit (using SIP SUBSCRIBE) or implicit (without SUBSCRIBE) depending on the network implementation. [16] defines the requirements to both approaches.

An example to event notification to SIP is Message Waiting Indication event package and associated message-summary content defined in [17].

Though introduced as a solution to IMS networks, the RTSP Event Notification is highly applicable to basic SIP networks, where an intermediate RTSP node, an RTSP proxy can be introduced for "snooping" function. The requirement of this RTSP proxy is to have an integrated SIP stack to be able to send out NOTIFY messages corresponding to RTSP events.

1.2.4 Service Blending

[4] introduces two concepts: Service Bundling and Service Blending. Service Bundling offers unified ordering and billing, limited interaction with otherwise separate services. Service Blending, on the other hand, enables different services to control each other.

[4] suggests a new IMS node: Service Broker, which serves as a SCIM [13], but actually goes beyond SCIM functionality; SCIM is designed for blending SIP-based services whereas the Service Broker not only blends Sip-based services, but services with a variety of interfaces and delivery mechanisms. Service Broker communicates with the CSCF to retrieve and send call and session events. It also communicates with one or more application servers to coordinate their behavior according to the requirements generated by blended service specifications [4].

[18] mentions the lack of standardization on RTSP and HTTP proxy based wireline/wireless applications. The document shows an RTSP and HTTP/WAP based high level architecture (Figure 1.1).

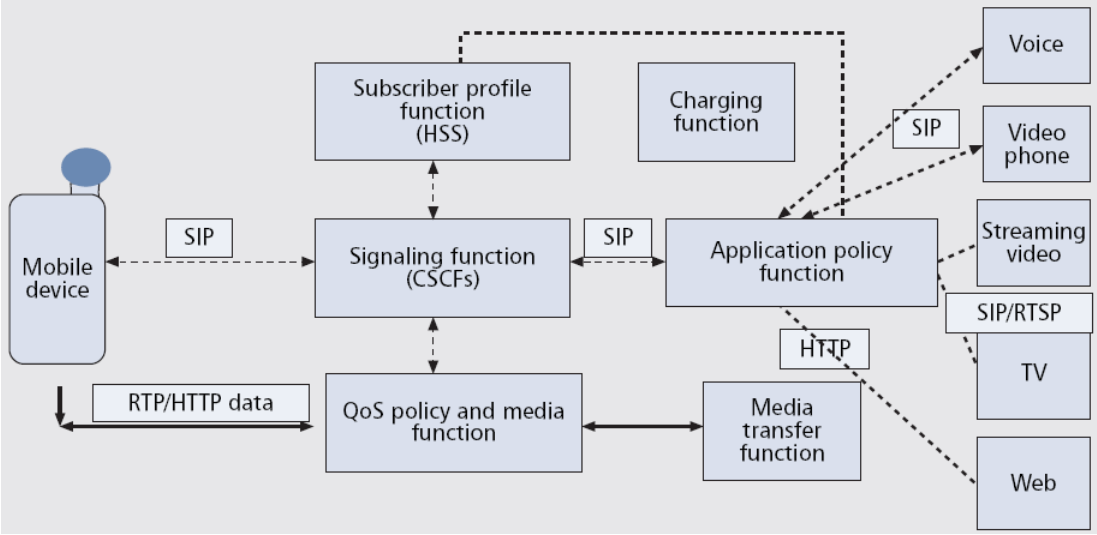

Figure 1.1 : Extended IMS Architecture

introduces two prototype applications: SSV and MMP: the two defines different types of IMS-IPTV interactions.

The prototype SSV application uses SIP/IMS for call setup and session management, RTSP for streaming video, and HTTP to create a rich user interface for conference control sharing the streaming video.

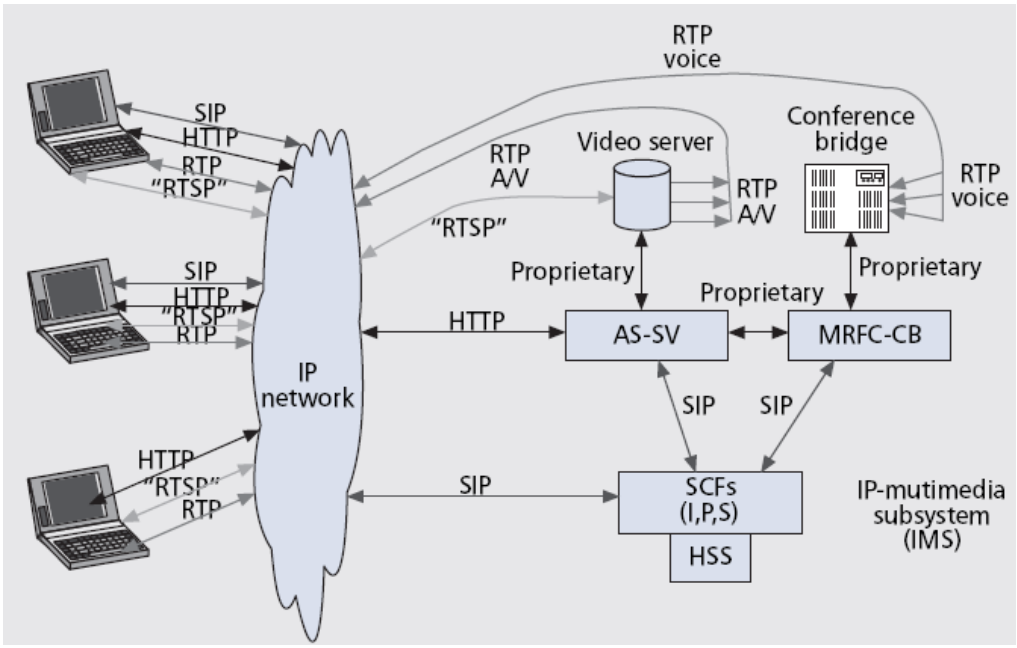

Figure 1.2 : SSV Solution

The Prototype MMP is introduced as a way of enhancing the IMS architecture to harmonize HTTP and RTSP services in both the existing and IMS domains, by combining proxy functions that currently exist in different networks. It contains integrated SIP, RTSP and HTTP proxies to be on the messaging path and provide necessry interaction between these protocols.

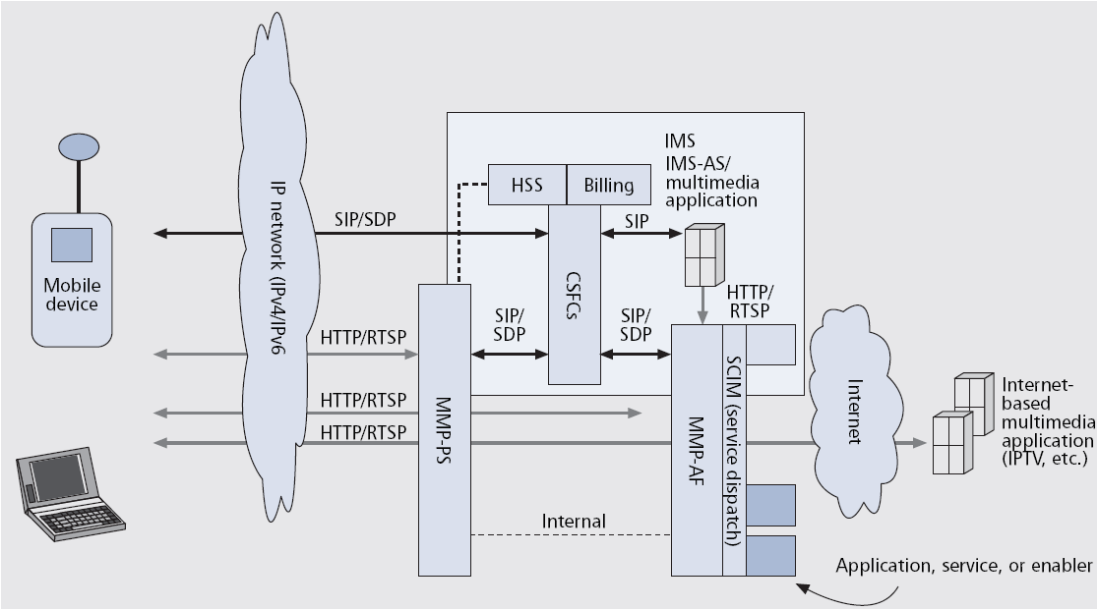

Figure 1.3 : MMP Solution

Service Brokering, which encapsulates the Service Blending concept, is an open discussion area on IMS network. Related work is published through 3GPP specification [19] . One of the requirements on the specification is to allow service integration between SIP and non-SIP applications available via the IMS service architecture. This requirement would sure include SIP and RTSP interaction.

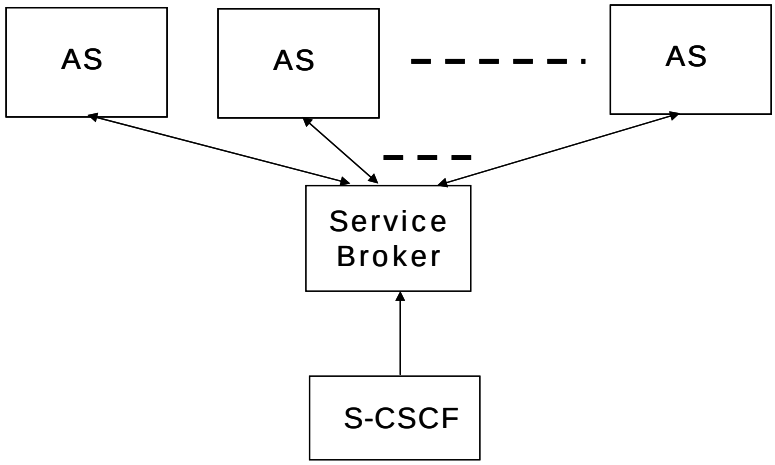

Figure 1.4 : Service Broker Architecture

1.3 Problem Definition

SIP protocol and RTSP share a lot of concepts in common. These similarites can be used to combine these protocols to overcome some problems encountered in the coexistance of two.

Both protocols:

- Are built on HTTP and reuses HTTP concepts.
- Support HTTP Digest Authentication scheme for security.
- Are easily extendible and parsable.
- Are transport independent
- Are multiple-server capable
- Support SDP for presentation description
- Are proxy and firewall friendly

Dual Stack approach on SIP-RTSP interworking require these concepts to coexist independently. However, independent coexistence of some of the concepts can introduce problems mentioned in the following section

1.3.1 Security Consideration

One of the security requirements of IPTV defined by ITU-T is authorized access: "The IPTV architecture shall provide one or more mechanisms to establish authorization before service delivery. The IPTV architecture shall provide a mechanism to allow the service provider to force users of the service to participate in an authentication and authorization procedure with the network, before granting access to the service. The IPTV Architecture shall provide mechanisms to control unauthorized access by unsubscribed end-users to the service. It may redirect unsubscribed end-users to a mechanism where they may subscribe." [20]

HTTP Digest Authentication Scheme [21] is the authentication that is already supported by SIP [10] and RTSP [22]. The medhod requires the subscriber password to be stored on the server . By doing that, the server is able to authenticate any incoming request against the subscriber; it is validated whether the digest response in the Authorization header is actually generated using the subscriber password [21].

On RTSP case, request authentication is an important security measure; since, a DoS attack based on “TEARDOWN” message flooding can prevent the whole system not to function. On the other hand, not all media servers may be part of the service provider network: there is possibility that the service provider pays to a third party Content Provider for the IPTV service. In that case, the service provider is obliged not to share the subscriber passwords with the third party IPTV provider. This situation raises the necessity of an RTSP proxy, which knows the subrciber and is capable to apply HTTP Digest authentication to subscriber requests, inside the service provider network.

Instead of using an RTSP proxy, tunelling RTSP embedded in SIP can be another alternative to solve this problem. This approach is a simpler solution which eliminates the work on how to insert a RTSP proxy into the service providers network.

1.3.2 Session Presentation Layer

Dual-Stack approach introduces the presentation information to be negotiated independently via SIP and RTSP. However, some media NAT-traversal solutions would require the presentation information to be always transparent on SIP layer. These type of solutions introduce a media proxy to be inserted on the media-path. In order to insert the media proxy a SIP proxy or a SIP BBUA (like SBC) needs to acquire the presentation information via SIP messages in each event requiring presentation exchange (hold/retrieve or codec change…). There are two alternatives to overcome this conflict:

To implement an RTSP proxy which is also capable of inserting media proxy.

To completely abondon presentation exchange on RTSP layer and leave that resposibility to SIP.

First approach suggests rather much work. Second approach is easiest to implement and is part of the RTSP-C solution

1.3.3 NAT Traversal

NAT traversal is a common requirement TCP/IP networking, of establishing connections between hosts in private TCP/IP networks that use NAT devices. There are suggested mechanisms (STUN, TURN, ICE) to overcome this issue some of which are also introduced to SIP [23] and RTSP [24]. These methods are able to solve the NAT traversal problem at the expense of some messaging overhead.

If SIP and RTSP are to be used together, combining the two protocols will benefit on a unified/light-weight NAT traversal procedure.

1.3.4 Streaming Based Network Services

Based on the developments on VOIP and IPTV, there would be a requirement for the SIP core to be aware of the streaming state. Once the SIP proxy is aware of the streamin state, it can provide services based on this information. For instance, a subscriber can define a call processing rule for himself using CPL [25] as:

“When a call is received from anyone on my Personal Address Book, if I am watching video, and the state is not paused, push the video URL to the caller”

This hypotetical feature requires to have the rtsp url (media url) and streaming state as inputs. If the client is getting a VoD session and the session is not paused, the url defining the network resource can be sent to the caller (The implementation on this area can vary). If VoD is at paused state, the next set of CPL rules are applicable. This would mean another pre-defined rule or the default rule which will ring the client. In that case, the subscriber is able to put the streaming session on hold while accepting the incoming call.

[3] proposes a solution for IMS network based on a network element snooping media control messages (RTSP or IGMP) and triggering SIP notify messages each time a media control message is issued. However, this option requires a network node that can understand RTSP messages and notify the SIP Proxy (proxies) on the media event. When a simple SIP Proxy interworking is considered, inserting this component to the RTSP path is another area that should be researched in detail.

1.4 Background

1.4.1 IPTV Technology

IPTV specifies the medium of communication of pictures and sound that operates over an IP network [26]. ITU-T defines IPTV as "IPTV is defined as multimedia services such as television/video/audio/text/graphics/data delivered over IP based networks managed to provide the required level of QoS/QoE, security, interactivity and reliability" [20].

IPTV offers [26]:

Content: IPTV technology promises to make more content available, make it easier to access and make it portable (while maintaining security).

Convergence: The utilization of an IP network will allow applications to be run over multiple end-user devices, all over a single service delivery network.

Interactivity: The two-way nature of the IP network will enable unprecedented interaction among subscribers, content providers and service providers.

The following figure shows the main roles. Four roles are described as follows:

Content Provider: The entity that owns or is licensed to sell content or content assets.

Service Provider: The entity that providing the IPTV Service to the Customer.

Typically, the Service Provider acquires or licenses content from Content Providers and packages this into a service that is sold to the Customer.

Network Provider: The entity providing that connects the Customers and the Service Providers.

Customer: The entity that consumes and pays for the IPTV Service.

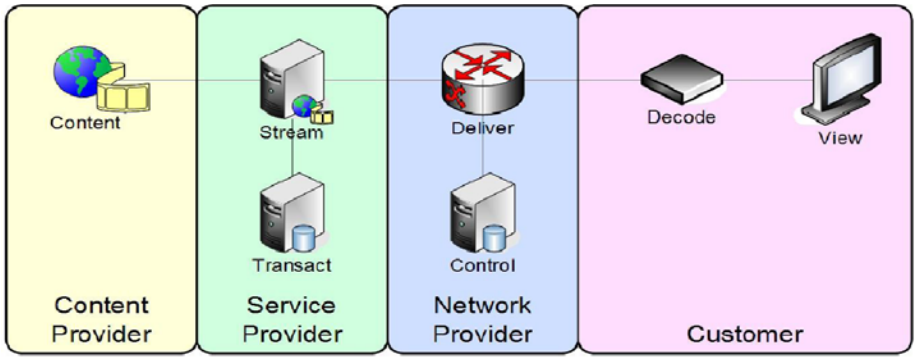

Figure 1.5 : IPTV Reference Model

1.4.1.1. IPTV Video On Demand

Video on demand (VoD) is a special form of IPTV which operates in a different manner than linear (broadcast) television service as the IPTV system provides the subscriber with a unicast stream of programming with VCR-like controls including pause, fast forward and rewind [26].

1.4.2 Session Initiation Protocol (SIP)

SIP describes how to set up Internet telephone calls, video conferences, and other multimedia connections. Unlike H.323, which is a complete protocol suite, SIP is a single module, but it has been designed to interwork well with existing Internet applications. For example, it defines telephone numbers as URLs, so that Web pages can contain them, allowing a click on a link to initiate a telephone call (the same way the mailto scheme allows a click on a link to bring up a program to send an e-mail message)[27].

SIP can establish two-party sessions (ordinary telephone calls), multiparty sessions (where everyone can hear and speak), and multicast sessions (one sender, many receivers). The sessions may contain audio, video, or data, the latter being useful for multiplayer real-time games, for example. SIP just handles setup, management, and termination of sessions. Other protocols, such as RTP/RTCP, are used for data transport. SIP is an application-layer protocol and can run over UDP or TCP [27].

SIP supports a variety of services, including locating the callee (who may not be at his home machine) and determining the callee's capabilities, as well as handling the mechanics of call setup and termination. In the simplest case, SIP sets up a session from the caller's computer to the callee's computer, so we will examine that case first [27].

Telephone numbers in SIP are represented as URLs using the sip scheme, for example, sip:ilse@cs.university.edu for a user named Ilse at the host specified by the DNS name cs.university.edu. SIP URLs may also contain IPv4 addresses, IPv6 address, or actual telephone numbers[27].

The SIP protocol is a text-based protocol modelled on HTTP. One party sends a message in ASCII text consisting of a method name on the first line, followed by additional lines containing headers for passing parameters. Many of the headers are taken from MIME to allow SIP to interwork with existing Internet applications[27].

To establish a session, the caller either creates a TCP connection with the callee and sends an INVITE message over it or sends the INVITE message in a UDP packet. In both cases, the headers on the second and subsequent lines describe the structure of the message body, which contains the caller's capabilities, media types, and formats. If the callee accepts the call, it responds with an HTTP-type reply code. Following the reply-code line, the callee also may supply information about its capabilities, media types, and formats[27].

Connection is done using a three-way handshake, so the caller responds with an ACK message to finish the protocol and confirm receipt of the 200 message[27].

Either party may request termination of a session by sending a message containing the BYE method. When the other side acknowledges it, the session is terminated[27].

The OPTIONS method is used to query a machine about its own capabilities. It is typically used before a session is initiated to find out if that machine is even capable of voice over IP or whatever type of session is being contemplated[27].

SIP has a variety of other features including call waiting, call screening, encryption, and authentication. It also has the ability to place calls from a computer to an ordinary telephone, if a suitable gateway between the Internet and telephone system is available[27].

1.4.3 Real Time Streaming Protocol (RTSP)

The protocol supports the following operations:

1.4.3.1. Retrieval of media from media server:

The client can request a presentation description via HTTP or some other method. If the presentation is being multicast, the presentation description contains the multicast addresses and ports to be used for the continuous media. If the presentation is to be sent only to the client via unicast, the client provides the destination for security reasons.

1.4.3.2. Invitation of a media server to a conference:

A media server can be "invited" to join an existing conference, either to play back media into the presentation or to record all or a subset of the media in a presentation. This mode is useful for distributed teaching applications. Several parties in the conference may take turns "pushing the remote control buttons."

1.4.3.3. Addition of media to an existing presentation:

Particularly for live presentations, it is useful if the server can tell the client about additional media becoming available.

1.5 Hypothesis

SIP based IPTV applications enable SIP/VOIP features to IPTV systems and enable IPTV to be reachable in VOIP world. Media stream control remains to be an important aspect of IPTV systems and needs to be integrated with SIP. As SIP and RTSP evolved in two separate paths, they posses many similar features. Therefore, the best way to integrate the two protocols is to merge them together: treating SIP as the carrier and RTSP as the message body. Therefore dublicate features can be trimmed out of RTSP leaving a compact form; while the protocol capabilities are enriched with SIP features.

2. SIP-RTSP CONVERGENCE: RTSP-C

2.1 Objectives

The main focus of this chapter is to introduce a convergence model for SIP and RTSP. RTSP-C which is the compact mode of RTSP is defined as a multipart mime body element of a SIP message body and is the suggested protocol to implement Video on Demand type IPTV applications on SIP network.

As referenced in section 1.2 there are two options to achieve media control and SIP interoperability: all-SIP approach and Dual Stack approach. Dual Stack approach fulfils the basic requirements of interoperability; enabling both protocols to setup and operate in their own layer. However, it does not define a way for SIP service layer to be aware of media control commands.

On the other hand, all-SIP approach requires SIP to be used to transport the media control. Therefore, SIP Core (the SIP Proxy) is aware of the media control and is able to provide services based on streaming/media control.

2.2 Protocol Details

RTSP-C is built on existing concepts RTSP but is made more compact for the use of SIP protocol. As some of the funtionalities of RTSP is already covered by SIP (like authentication, NAT traversal, session presentation exchange) unnecessary message overhead is excluded to form a more compact messaging.

RTSPC relies on two mechanisms for transport:

- Multipart mime body extension on SDP offer/answer model in SIP
- SIP INFO messages

Call setup and mid/call negotiations would also require RTSP-C negotiation. For that purpose, related RTSP-C messages are introduced as multipart mime extensions [28]. Therefore the content-type of the SIP request/response would be “multipart/mixed” while the actual content-type of the multipart mime element is “application/rtspc”.

By definition, the INFO method is used for communicating mid-session signaling information along the signaling path for the call [15]. The INFO method provides additional optional information which can further enhance the application using SIP. This property makes it the appropriate vessel to carry mid-call RTSPC signalling (Trick-plays). Since the body content does not need to be multipart here, SIP Content-type header will contain “application/rtspc”.

As SDP exchange is handled through either INVITE or UPDATE messages, there is no need for ANNOUNCE or DESCRIBE messages. These messages are removed in this version of RTSP-C.

As GET_PARAMETER method requires an answer with a message body, it is removed off this version of RTSP-C.

REDIRECT method is also removed as the redirectioning of the media will be handled on SIP level.

In order to advertise the support of RTSP-C, “mediacontrol” field is suggested as a new supported header field. If this field is present, a SIP UA can look for RTSP-C content in the message body or send RTSP-C content.

The sytax of the protocol is a bit different than that of the original. In order to simplify the parsing operation, request URL is also divided to headers:

“M” header identifies the method while “v” identifies the protocol version. “url” is represented in “u” header, which is explained in more detail in the following section.

2.2.1 Session Identification

RTSP uses RTSP-URL (defined at [22] p.15) to identify the network resource. There are two protocol extensions defined: “rtsp” for reliable transport (TCP) and “rtspu” for unreliable transport (UDP). An example RTSP URL:

rtsp://media.example.com:554/twister/

where:

“rtsp” defines the protocol and that it uses reliable transport (TCP), “media.example.com” defines a valid host name for the media server,“554” is the default rtsp port, “twister” is the media resource

Since it is transported via SIP protocol, RTSPC does not require to identify a transport method or a port number. Thus, regarding RTSPC URL can be formed as:

rtspc://media.example.com /twister/

RTSPC url is carried via url header “u”:

u= rtspc://media.example.com /twister/

As SIP Call-ID already identifies the dialogs, and therefore the session, a session-id is not required. On the other hand, in order to reserve the capability to transfer the session to another client, a unique session-id must be negotiated between the client and the media-server.

i=some-random-sequence

Just like session identification, message sequencing is also covered by SIP cseq header. Therefore this functionality is also not needed on RTSPC.

RTSPC treats media just like SDP. It is probable that the media is composed of more than one track (video, audio, subtitle…). For this case, instead of explicitly referencing each track with an rtspc url, a media description element, “m”, is used as the one in SDP. An “m” element can also have attributes included in a “a” header.

2.2.2 Session Initiation

The SETUP message is used to address the media of interest; the network resource to be requested from the media server.

Success response for SETUP is 200 OK, which can also be used to advertise the media control capabilities of the server and some attributes of the media of interest. The options header, “o”, is used to list the valid methods executable by the client. This header can be used to construct the user interface on the client. The media server can also advertise language options of each track (audio, subtitle) via the optional media attribute “a=lang:”.

When the referenced media is not found on the media server, this also means end of the call. Therefore, the approptiate SIP response, “404 Not Found” is returned by the media server.

When the media server is not capable of RTSP-C processing, it should return with “415 Unsupported Media Type”.

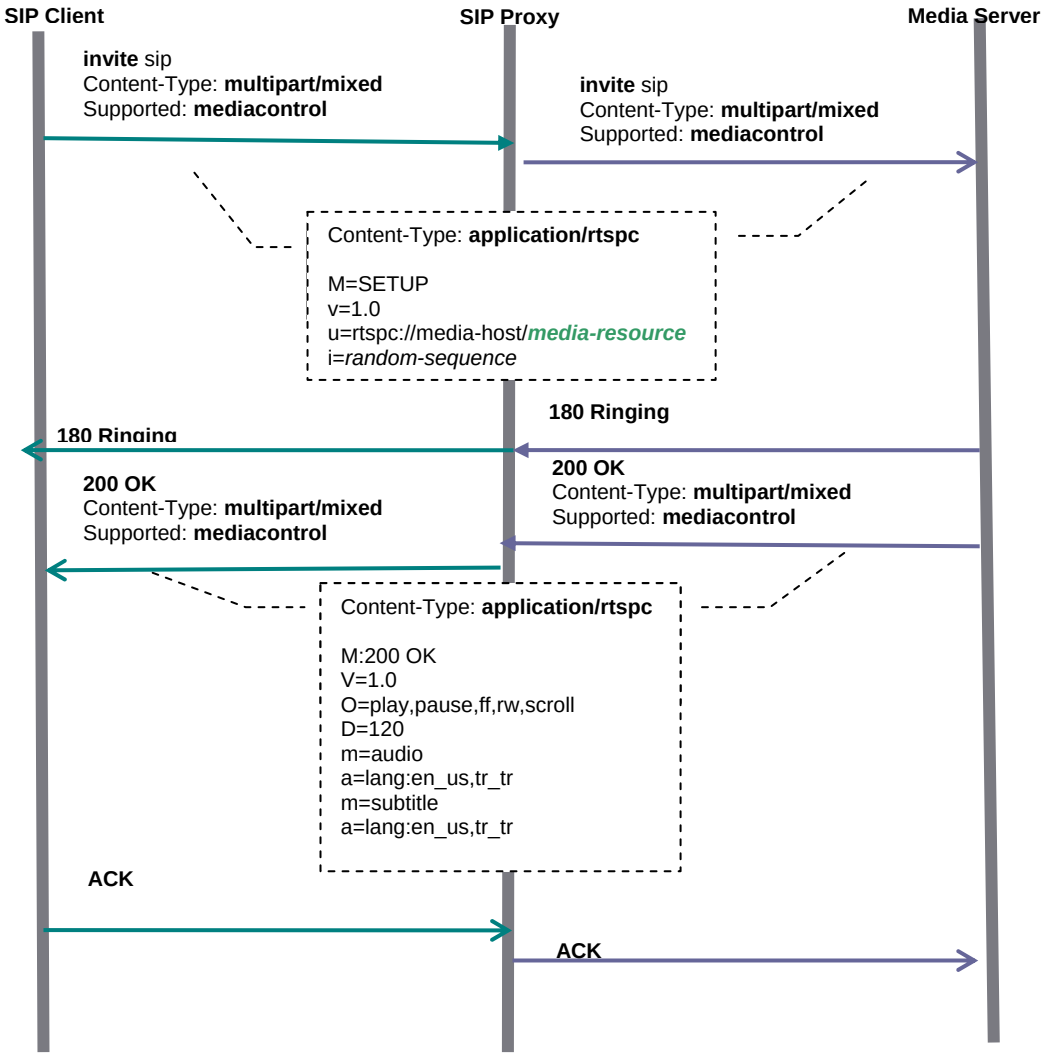

Figure 2.1 : RTSP-C Session Initiation

Figure 2 shows a simple call flow including the RTSP-C sub-messaging.

2.2.3 Mid-Call Streaming Control (Trick Play Implementation)

For mid-call streaming control, SIP INFO message is used as the container. It is mandatory that each RTSP-C body has an RTSP-C url referencing the media in question. To reduce messaging, more than one track setting can be posted in a message. Figure 3 shows an example PLAY message, where language settings of audio and subtitle are presented independent of each other.

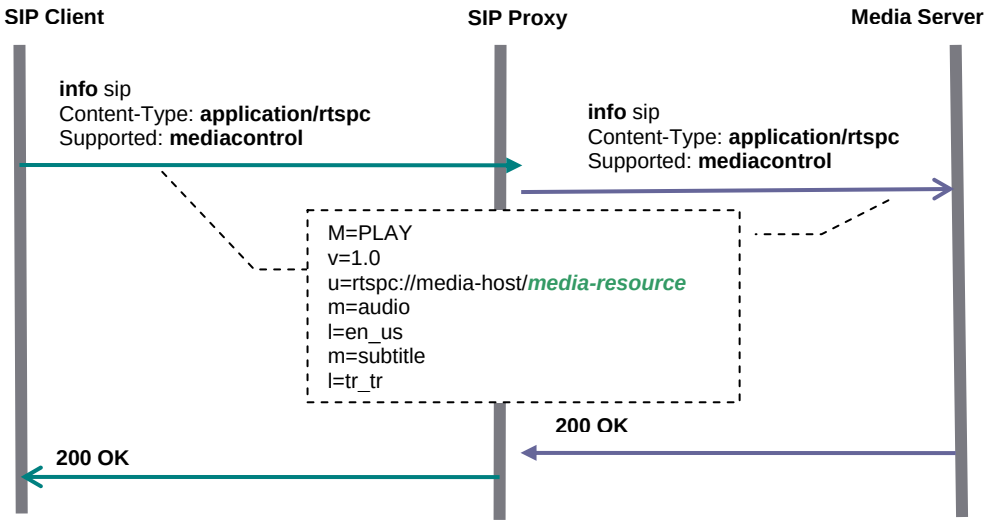

Figure 2.2 : Sample PLAY Message Flow

As of protocol version 1.0, supported trick-play methods are PLAY, PAUSE and RECORD. Section 3.1.2.5 explains these methods in more detail. Rewind, fast-forward and time scrolling are implemented as variations of PLAY method.

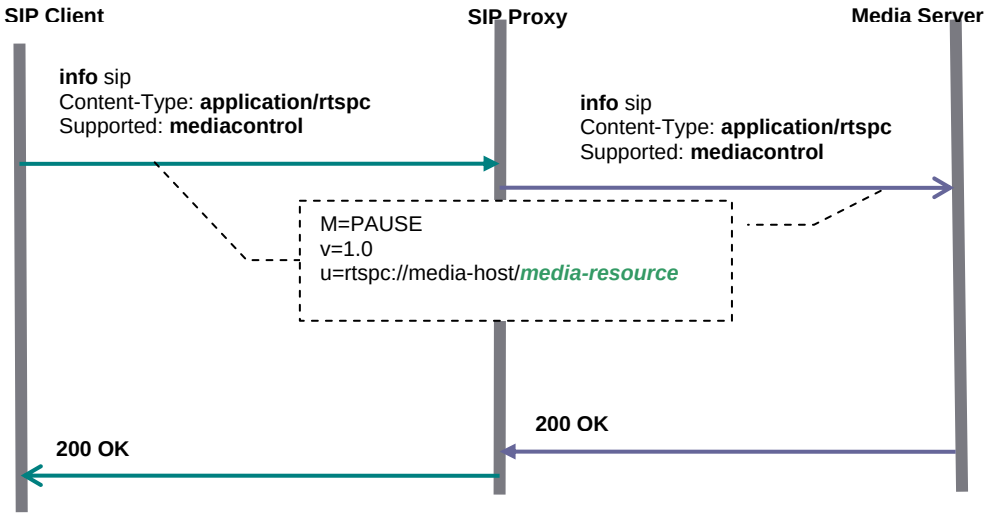

Figure 2.3 : Sample PAUSE Messaging

As defined in [15], it is not permitted 200 OK response to INFO message to have a body. Therefore, the meaning of 200 OK response is that the media server auccessfully acquired the RTSP method to be processed.

2.2.4 Session Tear-down

As session tear-down would also mean the end of the call, SIP BYE message can be used to end the streaming.

2.2.5 Authentication

SIP provides a stateless, challenge-based mechanism for authentication that is based on authentication in HTTP. Any time that a proxy server or UA receives a request it MAY challenge the initiator of the request to provide assurance of its identity. Once the originator has been identified, the recipient of the request SHOULD ascertain whether or not this user is authorized to make the request in question [10].

Therefore, Trick-plays, which are transmitted through INFO messages can also be authenticated using HTTP Digest scheme.

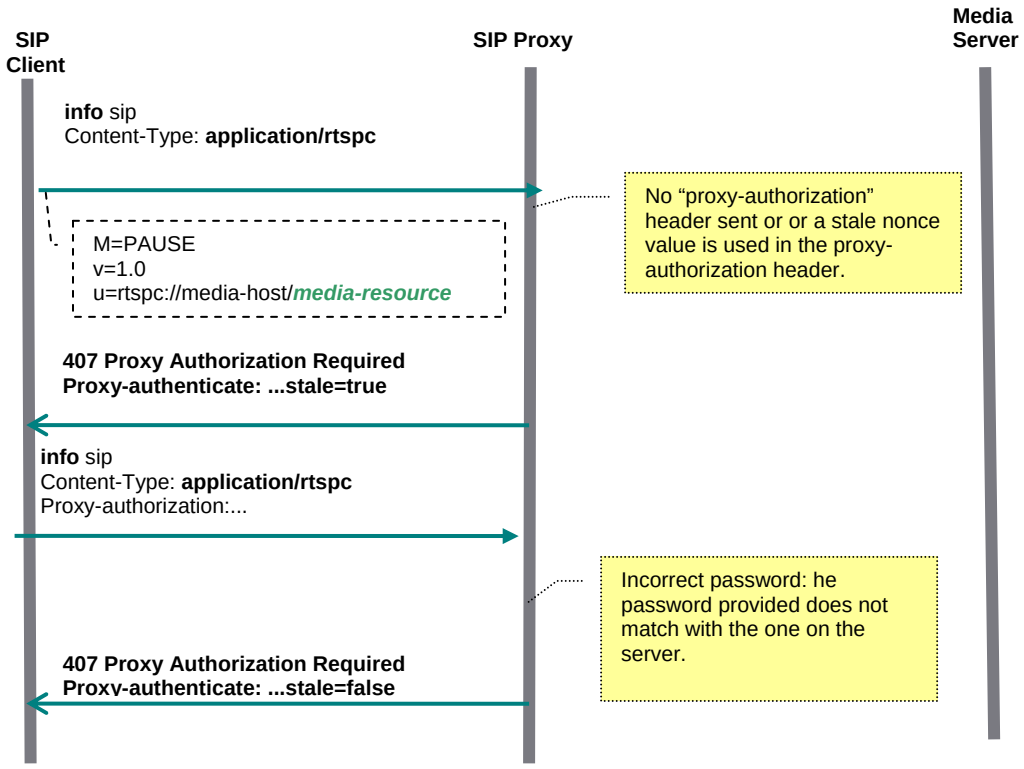

Figure 2.4 : Authentication of Trick-Play's

Figure 2.4 shows two authentication failures:

First INFO message is challenged with 407 as it does not have a "proxy-authorization" header, which contains the credentials to access the system. In order to identify that, 407 Proxy Authorization required header contains the stale flag set to true. A second case is a stale nonce (which is a random character array provided by the server): if the credentials are valid but nonce value on the "proxy-authorization" header does not match the one on the SIP proxy, a 407 challenge is sent back with

stale flag set to true and containing the server nonce [21].

Second INFO message is challenged with 407 as the password used to create the "proxy-authorization" header. For that reason, 407 challenge is sent back with stale flag set to false [21].

Integrity of the message body can also be secured via usage of "authentication with integrity" option defined with "qop=auth-int" parameter in the proxy-authorization header. When this option is used, message body is also added to the digest algorithm; therefore the server is able to determine if the message body is tampered or a proxy-authorization header of a previous message is replayed [21].

Appendix A.2 gives the definitions of HTTP Authentication related terms. More information on HTTP Authentication can be found on [21].

2.2.6 Protocol State Machine

2.2.6.1. Client State Machine

Client states are parallel to those of RTSP [22]. The client can assume the following states:

- Init: SETUP has been sent, waiting for reply.
- Ready: SETUP reply received or PAUSE reply received while in Playing
- Playing: PLAY reply received
- Recording: RECORD reply received

Table 2.1 : Client State Machine

State	Message Sent	Next State (After Response is Received)
Init	SETUP	Ready
	TEARDOWN	Init
Ready	PLAY	Playing
	RECORD	Recording
	TEARDOWN	Init
	SETUP	Ready
Playing	PAUSE	Ready
	TEARDOWN	Init
	PLAY	Playing
	SETUP	Playing (changed transport)
Recording	PAUSE	Ready
	TEARDOWN	Init
	RECORD	Recording
	SETUP	Recording (changed transport)

2.2.6.2. Server State Machine

Client states are parallel to those of RTSP [6]. The server can assume the following states:

- Init: The initial state, no valid SETUP has been received yet.
- Ready: Last SETUP received was successful, reply sent or after playing, last PAUSE received was successful, reply sent.
- Playing: Last PLAY received was successful, reply sent. Data is being sent.
- Recording: The server is recording media data.

Table 2.2 : Server State Machine

State	Message Sent	Next State (After Response is Received)
Init	SETUP	Ready
	TEARDOWN	Init
Ready	PLAY	Playing
	RECORD	Recording
	TEARDOWN	Init
	SETUP	Ready
Playing	PAUSE	Ready
	TEARDOWN	Init
	PLAY	Playing
	SETUP	Playing (changed transport)
Recording	PAUSE	Ready
	TEARDOWN	Init
	RECORD	Recording
	SETUP	Recording (changed transport)

2.2.7 Message Types

2.2.7.1. SETUP

SETUP message is used to initiate a media session controlled by RTSP-C. The network resource referenced by RTSP-C url is requested from the media server.

2.2.7.2. SET_PARAMETER

This method requests to set the value of a parameter for a presentation or stream specified by the URI.

A request SHOULD only contain a single parameter to allow the client to determine why a particular request failed. If the request contains several parameters, the server MUST only act on the request if all of the parameters can be set successfully. A server MUST allow a parameter to be set repeatedly to the same value, but it MAY disallow changing parameter values.

Note that transport parameters for the media stream MUST only be set with the SETUP command.

2.2.7.3. PLAY

PLAY message is used to start streaming on an already initiated media session. It can be enhanced with scale (c), range (r) and speed (s) headers to implement fast-forward, rewind or stream scrolling. Details on these headers canbe found on section 3.2.8.

2.2.7.4. PAUSE

PAUSE request causes the stream delivery to be interrupted (halted) temporarily. If the request URL names a stream, only playback and recording of that stream is halted. For example, for audio, this is equivalent to muting. If the request URL names a presentation or group of streams, delivery of all currently active streams within the presentation or group is halted. After resuming playback or recording, synchronization of the tracks MUST be maintained.

2.2.7.5. RECORD

This method initiates recording a range of media data according to the presentation description. It is referenced for future work.

2.2.8 Response Codes

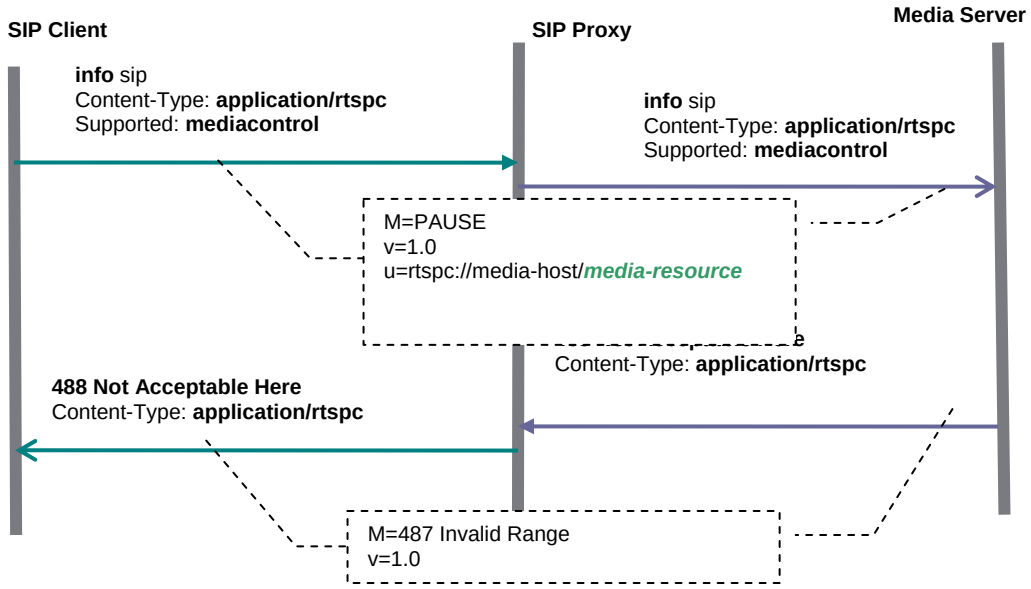

Figure 2.5 : Transmission of Response Codes

In order to support transport of 4xx/5xx RTSP reason codes via SIP in the compact mode, either “488 Not Acceptable Here” or “606 Not Acceptable” used with a RTSP-C message body containing the RTSP reason code. Upon receival of any 4xx/5xx/6xx message RTSP-C state stays unchanged.

Table 2.3 : RTSP Response Code Transportation

Code	Reason Phrase	Methods	Transported By
400	Bad Request	all	RTSP-C
401	Unauthorized	all	SIP
402	Payment Required	all	RTSP-C
403	Forbidden	all	RTSP-C
404	Not Found	all	RTSP-C
405	Method Not Allowed	all	RTSP-C
406	Not Acceptable	all	RTSP-C
407	Proxy Authentication Required	all	SIP (Done on SIP Proxy)
408	Request Timeout	all	SIP
410	Gone	all	SIP
411	Length Required	all	SIP
412	Precondition Failed	DESCRIBE, SETUP	RTSP-C
413	Request Entity Too Large	all	RTSP-C
414	Request-URI Too Long	all	RTSP-C
415	Unsupported Media Type	all	SIP
451	Invalid parameter	SETUP	RTSP-C
452	Illegal Conference Identifier	SETUP	RTSP-C
453	Not Enough Bandwidth	SETUP	RTSP-C
454	Session Not Found	all	RTSP-C
455	Method Not Valid In This State	all	RTSP-C
456	Header Field Not Valid	all	RTSP-C
457	Invalid Range	PLAY	RTSP-C
458	Parameter Is Read-Only	SET_PARAMETER	RTSP-C
459	Aggregate Operation Not Allowed	all	RTSP-C
460	Only Aggregate Operation Allowed	all	RTSP-C
461	Unsupported Transport	all	SIP
462	Destination Unreachable	all	SIP
500	Internal Server Error	all	SIP
501	Not Implemented	all	RTSP-C
502	Bad Gateway	all	SIP
503	Service Unavailable	all	SIP
504	Gateway Timeout	all	SIP
505	RTSP Version Not Supported	all	RTSP-C

Since some RTSP reason codes are transport specific, and since SIP functions as the transport protocol over RTSP-C, some reason codes currently suppported on RTSP are related to SIP in this convergence model. Support of these reason codes are moved to SIP layer while the remaining reason codes can be used in the RTSP-C message body. Message body may also contain supported options to guide the client.

Detailed information on SIP and RTSP response codes can be reached on [22] and [17].

2.2.9 Message Headers

Table 2.4 : RTSPC Message Headers

Header	type	support	methods
Conference (C)	R	opt.	SETUP
Duration (d)	g	opt.	all
Language (l)	R	opt.	all
Media (m)	R	opt.	all
Media (m)	r	opt.	all
Method (M)	R	opt.	all
Method (M)	r	opt.	all
Options (o)	r	opt.	all
Range (r)	R	opt.	PLAY,PAUSE,RECORD
Range (r)	r	opt.	PLAY,PAUSE,RECORD
Require (Q)	R	req.	all
Scale (c)	Rr	opt.	PLAY, RECORD
Session-Id (i)	Rr	req.	SETUP
Speed (s)	Rr	opt.	PLAY
Unsupported (n)	r	req.	all
Version (v)	Rr	req.	all

2.2.9.1. Conference (C)

Identifies an existing conference ID that the RTSP session can join. (Reserved for future use).

2.2.9.2. Duration (d)

Shows the total duration of a media session (length of the media).

2.2.9.3. Language (l)

References the language selection of a media track. Can be referred as audio language or a subtitle language.

2.2.9.4. Media (m)

RTSPC treats media just like SDP. It is probable that the media is composed of more than one track (video, audio, subtitle…). For this case, instead of explicitly referencing each track with an rtspc url, a media description element, “m”, is used as the one in SDP. An “m” element can also have attributes included in a “a” header.

2.2.9.5. Method (M)

Method header identifies the RTSP-C method type. Can be SETUP, PLAY, PAUSE or RECORD.

2.2.9.6. Options (o)

Resides in eitther response or request messages. It defines the allowed RTSP-C methods for that UA.

2.2.9.7. Range (r)

This request and response header field specifies a range of time. This header is an exact mapping of RTSP range header. The range can be specified in a number of units. This specification defines the smpte ([6] Section 3.5), npt ([6] Section 3.6), and clock ([6] Section 3.7) range units.

2.2.9.8. Require (Q)

The Require header is used by clients to query the server about options that it may or may not support. The server MUST respond to this header by using the Unsupported header to negatively acknowledge those options which are NOT supported.

2.2.9.9. Scale (c)

A scale value of 1 indicates normal play or record at the normal forward viewing rate. If not 1, the value corresponds to the rate with respect to normal viewing rate. For example, a ratio of 2 indicates twice the normal viewing rate ("fast forward") and a ratio of 0.5 indicates half the normal viewing rate. In other words, a ratio of 2 has normal play time increase at twice the wallclock rate. For every second of elapsed (wallclock) time, 2 seconds of content will be delivered. A negative value indicates reverse direction.

2.2.9.10. Session-Id (i)

This is the unique identifier of the session. Since SIP Call-ID also does that, it is not required to send this header in mid-call messages. It is negotiated with the server via SETUP message. The usage of this header is reserved for transfer scenarios.

2.2.9.11. Speed (s)

This request header fields parameter requests the server to deliver data to the client at a particular speed, contingent on the server's ability and desire to serve the media stream at the given speed. Implementation by the server is OPTIONAL. The default is the bit rate of the stream.

2.2.9.12. Unsupported (n)

The Unsupported response header lists the features not supported by the server.

2.2.9.13. Version (v)

Specifies the version of the protocol message. Current version is 1.0

2.3 Use Cases

This section defines some of the use cases which will emphasize the advancement of RTSP-C convergement model.

2.3.1 "Watch with Me" Application

Here the hypotetical feature that has been presented in the introduction part will be discussed in more detail. In this feature, a subscriber defines a call processing rule for himself using CPL [25] as:

"When a call is received from anyone on my Personal Address Book, if I am watching video, and the state is playing, push the video URL to the caller"

With the use of RTSP-C, the SIP proxy is now aware of the media url (RTSPC url) and the streaming state (init, ready, playing, …). Using this information the server is able to run the CPL rule above and supply the caller enough information for him to receive the same media.Therefore:

- If the call is at a state other than "playing" the caller will be able to receive the call and the SIP UA will ring. The called subscriber can reject the call, redirect it to another subscriber or device, or answer the call on which the paused media sesion will be automatically put on hold.

- If the caller is at playing state, the server will acquire the media url, map it to a correct format and push this link to the caller. Ideally, the link pushed is of a web page which will be displayed on the media browser (on the client) of the caller. This way the caller will be informed of which media the called subscriber is receiving (it it is his/her favourite program or not) and wil be able to initiate a call through its media browser.

3. IMPLEMENTATION

3.1 Integrated Data

3.1.1 Objectives

This section introduces the external tools/packages that are used on this project implementation.

3.1.2 Platform and Tools

Program code has been developed on Java programming language using Eclipse IDE. The development is done on Windows Vista Operating system.

3.1.3 Java Media Framework

The Java Media Framework API (JMF) enables audio, video and other time-based media to be added to applications and applets built on Java technology. This optional package, which can capture, playback, stream, and transcode multiple media formats, extends the Java 2 Platform, Standard Edition (J2SE) for multimedia developers by providing a powerful toolkit to develop scalable, cross-platform technology. [29]

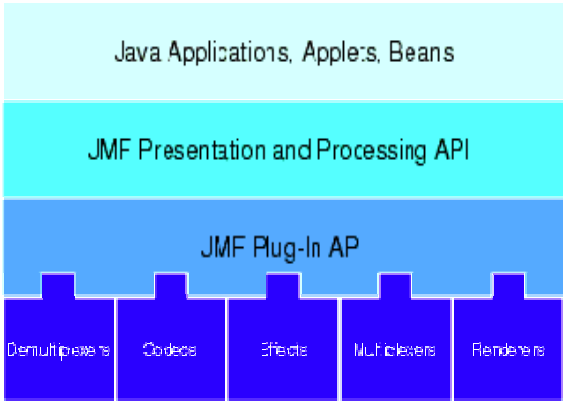

Figure 3.1 : JMF Architecture

JMF uses this same basic model. A data source encapsulates the media stream much like a video tape and a player provides processing and control mechanisms similar to a VCR. Playing and capturing audio and video with JMF requires the appropriate input and output devices such as microphones, cameras, speakers, and monitors. [30]

Data sources and players are integral parts of JMF's high-level API for managing the capture, presentation, and processing of time-based media. JMF also provides a lower-level API that supports the seamless integration of custom processing components and extensions. This layering provides Java developers with an easy-to-use API for incorporating time-based media into Java programs while maintaining the flexibility and extensibility required to support advanced media applications and future media technologies. [30]

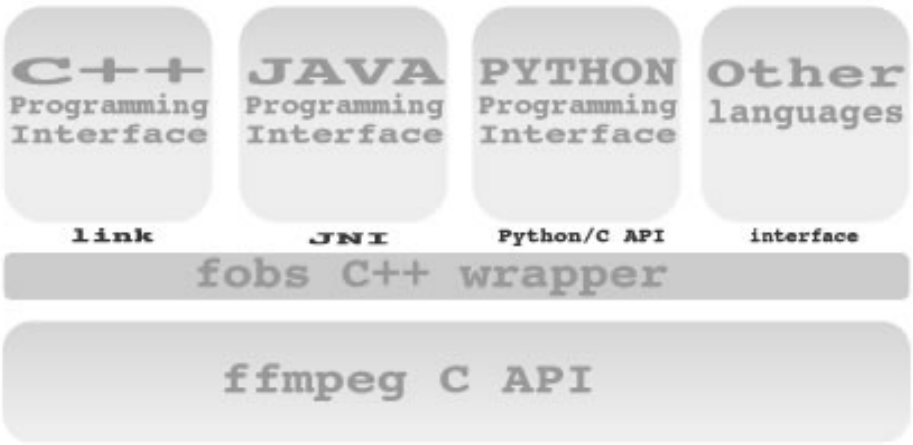

Figure 3.2 : Fobs4JMF Architecture

In order to extend media processing capabilities Fobs4JMF [31] plugin package is installed besides JMF. Fob4JMF is a JNI wrapper to reach native media processing methods of ffmpeg C API [22]. Fobs4JMF includes a preliminary version of java support as a JMF plug-in. With it, JMF is able to open any file supported by ffmpeg in any platform supported by ffmpeg. Fobs4JMF add compatibility in JMF for a lot of new codec's and formats (ogg, theora, xvid, h264, etc) [31].

3.1.4 SIP Stack

mJSip open source SIP stack & client have been adopted for SIP and RTSP-C implementation of this project. Based on mJSip, same stack code has been developped for both client and media server.

MjSip is a complete java-based implementation of a SIP stack.It provides in the same time the API and implementation bound together into the MjSip packages. MjSip is available open source under the terms of the GNU GPL license (General Public Licence) as published by the Free Software Foundation [32].

The MjSip stack has been used in research activities by Dpt. of Information Engineering at University of Parma and by DIE - University of Roma "Tor Vergata" and is currently commercially exploited by CreaLab [32].

MjSip includes all classes and methods for creating SIP-based applications. It implements the complete layered stack architecture as defined in RFC 3261 (Transport, Transaction, and Dialog sublayers), and is fully compliant with the standard. Moreover it includes higher level interfaces for Call Control and User Agent implementations [32].

More information on MjSip can be acquired through [32].

3.1.5 SIP Proxy

As a SIP Proxy, Nortel AS 5200 Release 10.2, which is a commercial product, has been used.

The Application Server 5200 (formerly know as the Multimedia Communication Server or MCS 5200) is designed to manage the complexity of a hyper-connected world by putting the intelligence in the heart of the network[33].

As well as delivering the richest set of voice and multimedia services, the AS 5200 delivers Fixed Mobile Convergence (FMC) applications which using a single number can reach the subscribers on any device or deliver the message to a single voice mailbox, advanced Web Services which deliver standardized Web Services APIs through which an enterprise can include telecommunication services into their web applications, and a powerful customer Web Portal so that the subscribers can customize their services to suit their own particular communication needs and communication style [33].

AS 5200 support various services under the feature groups:

- Voice
- Call Management
- Web Portal
- Web Services
- Fixed Mobile Convergence

- Collaboration Services
- Video

More information on AS 5200 can be reached through [23].

3.2 Network Elements

3.2.1 VoD Client

VoD Client is the receiver of the VoD service. It is capable of processing SIP, RTP, HTTP protocols and RTSP-C protocol. It is also able to receive audio, video and subtitle streams and and play them.

3.2.2 SIP Proxy

As both registrar and proxy, the SIP Proxy keeps the location information of SIP UA's and proxies the SIP messages to/from the SIP UA's. It is also the management center of SIP based services.

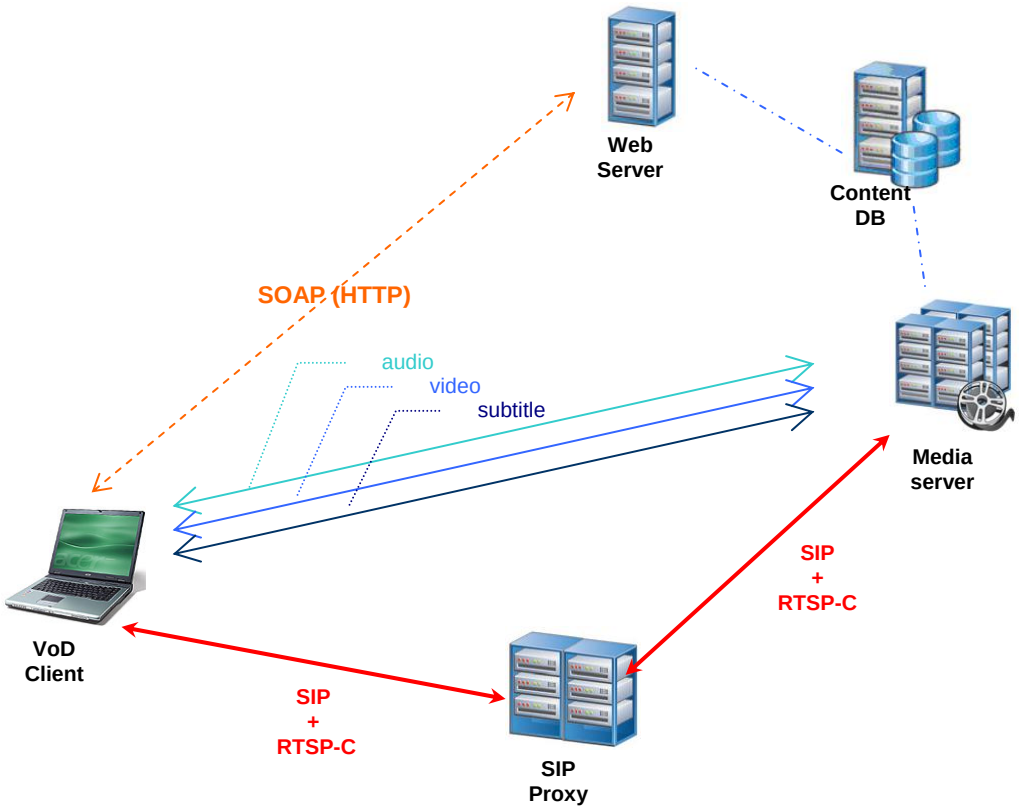

Figure 3.3 : Network Components

3.2.3 Web Server

Web server is connected to the IPTV service core and advertises the VoD media information to the outer world (VoD clients).

3.2.4 Media Server

Media Server hosts, encodes and sends media files. It is capable of processing SIP, RTP and RTSP-C protocols.

3.2.5 Content DB

Content DB provides the media information to the Web Server and is required to be in sych with the Media Server to provide correct information.

3.3 Software Architecture

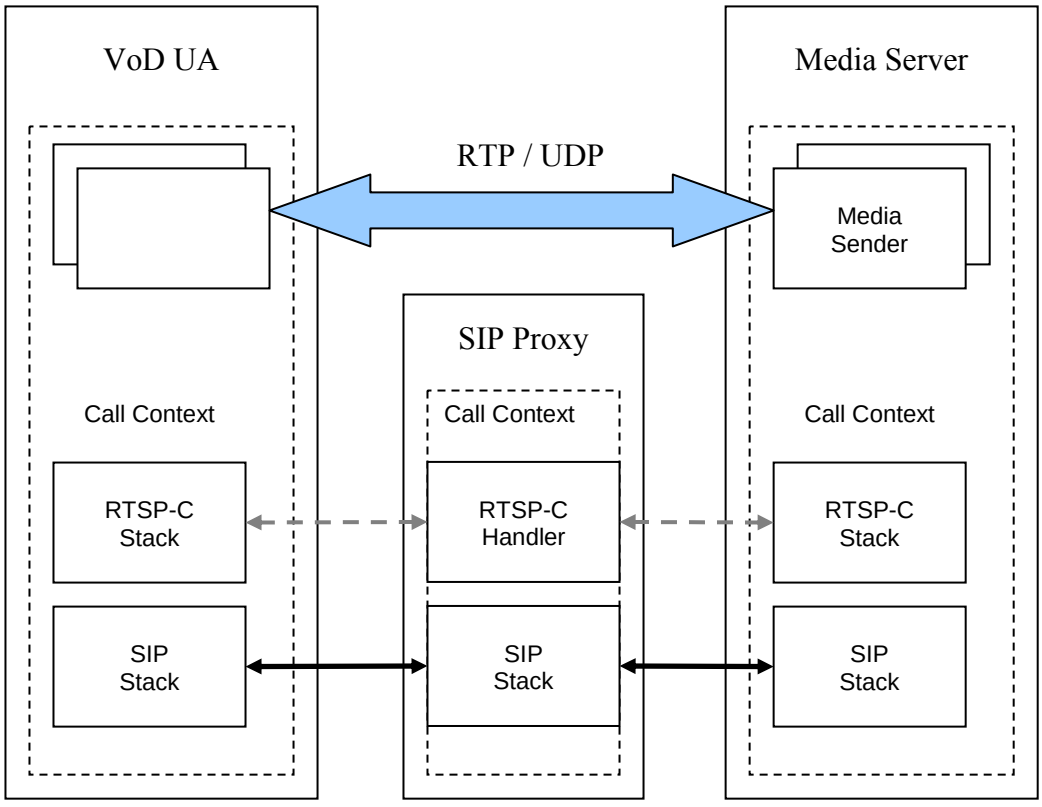

Figure 3.4 : Software Architecture

Figure 3.4 shows the protocol based software architecture of the implementation. As the SIP proxy will never directly handle media, it does not require a complete RTSP-C stack. Therefore a simple service to parse keep track of RTSP-C URL and methods (parse the methods from SIP body and keep track of them) is enough.

Media Sender encapsulates the JMF data structures for media processing and streaming. These structures are Processor, DataSource and RTPManager.

Media Receiver encapsulates the JMF player object which plays the received data streams as audio or video.

More information on JMF objects can be found on [19].

3.4 System Operation

3.4.1 Registration and Service Subscription

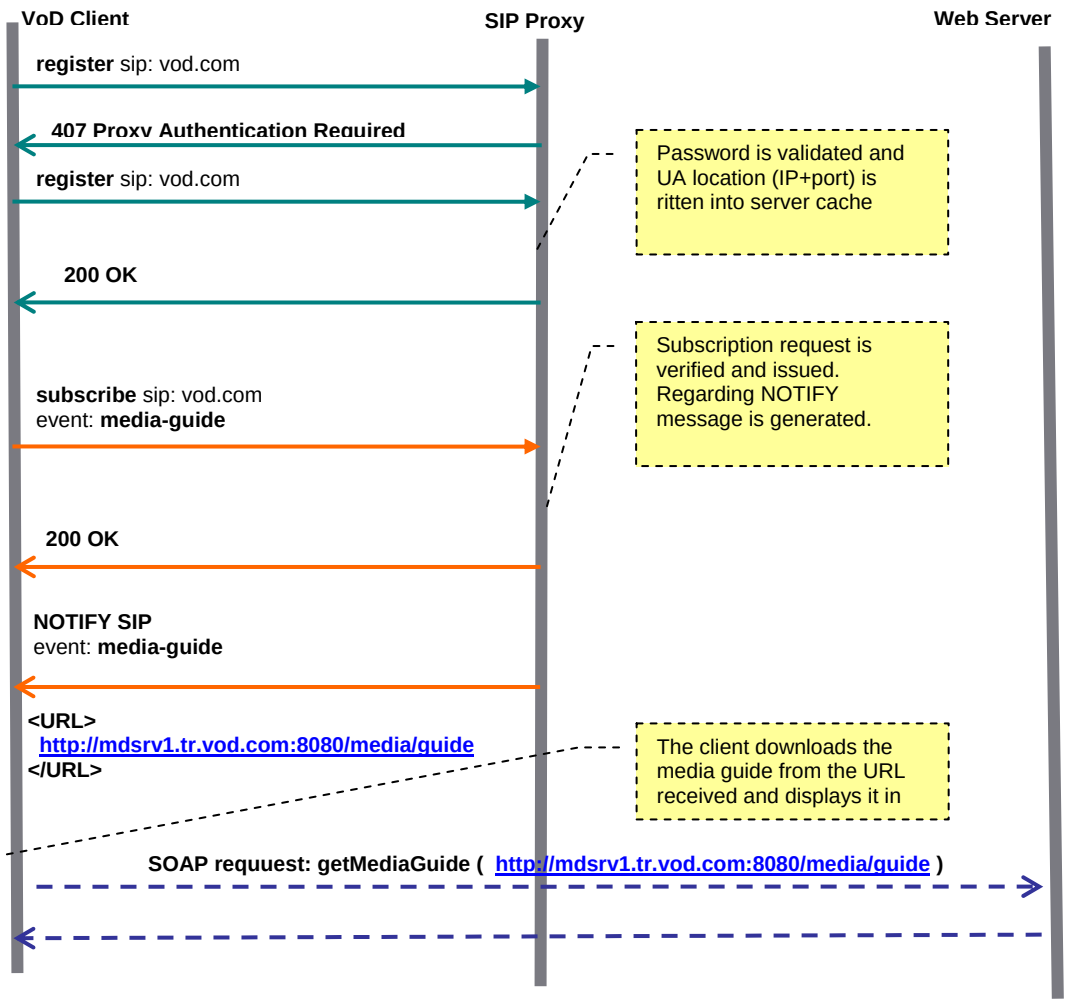

Figure 3.5 : Client Registration & Media Guide Subscription

In order to receive SIP based VOIP services, VoD Client registers to the SIP Proxy using the subscriber password. If the registration is successful, VoD client subscribes for “media-guide” in order to retrieve the media guide. Media-guide subscription is done via SIP SUBSCRIBE message with “media-guide” in the event header. After successful subscription (reported via 200 OK), SIP proxy sends a NOTIFY message to VoD client containing the HTTP/HTTPS URL that the client will be able to download the media guide. Details on SIP-Specific event notification can be reached from [16].

When VoD client downloads the media guide, movies and other media can be displayed on the client GUI.

3.4.2 Call Setup

In the call setup, the rtspc url is constructed based on the selection on the media guide. The SIP url is constructed as “sip:mediaserver@host-IP:5060”, in which “mediaserver” is assumed as an alias for the service. For simplicity, it is implemented as a SIP UA with the name “mediaserver” and registers to the SIP Proxy. However in reality it would be a defined as a static resource.

In the 200 OK response from the mediaserver, the O header (options) shows the usable options on the client. The client uses this header values to construct the GUI capabilities for media control. The response also contains available language capabilities of audio and subtitle streams which are introduced via “a=lang” attribute. The client will later be able to select from those language settings. D (duration) header gives the duration of the media, which is in this case is the movie length. If scrolling is available, the duration can be used in the construction of scroll-bar.

After the cal setup the client is in “ready” state waiting for the PLAY method.

3.4.3 Mid-Call Signalling

The streaming starts with the PLAY method sent through SIP INFO. Here, if no language selection is done prior, the default settings of the client will be used in the play method for the available streams. The GUI also gives capability to select among the provided language settings for audio and subtitle.

Scrolling is implemented via calculating the ratio of the current position on the scroll

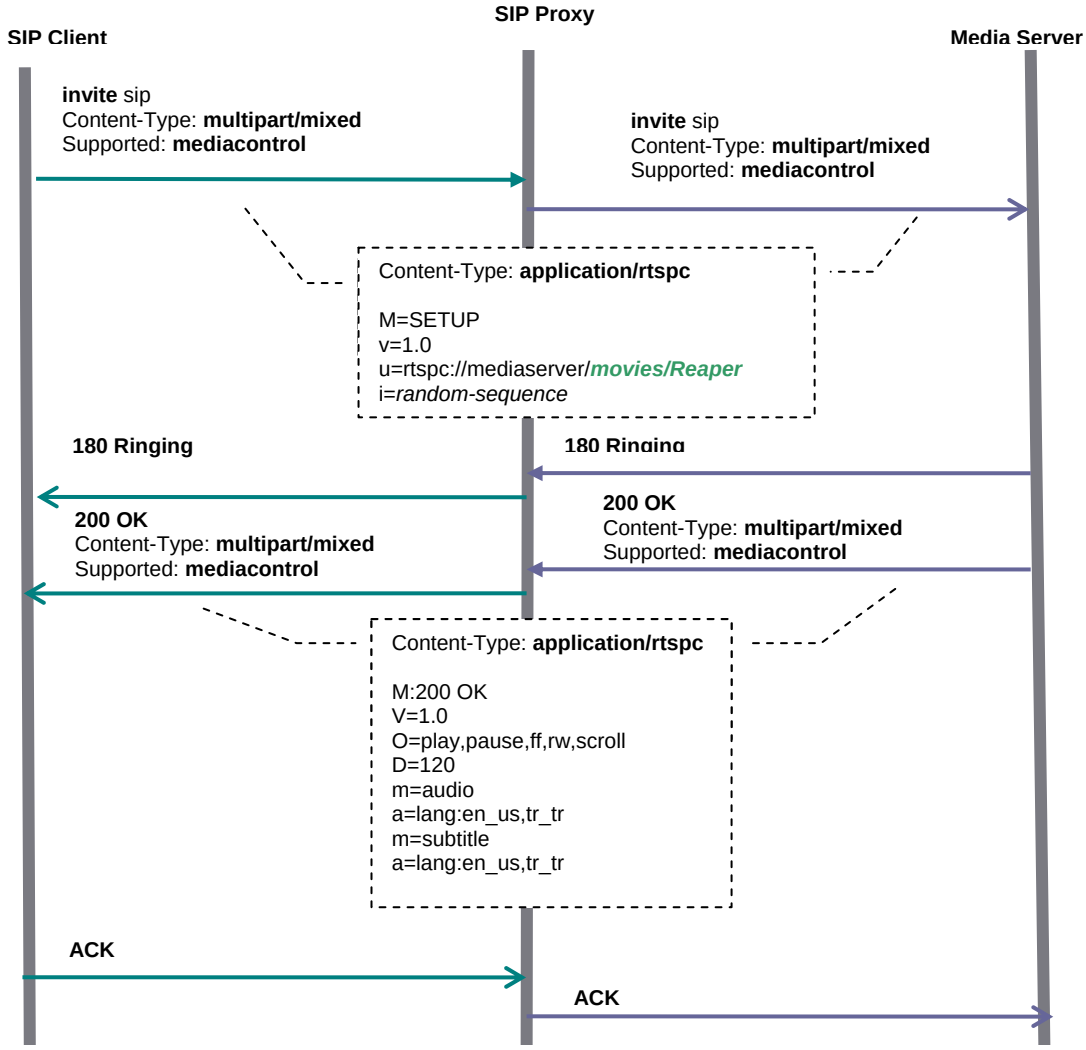

Figure 3.6 : Call Setup

bar on the GUI and multiplying the result with the duration (D) received in the 200 OK. The time value is sent in the r (range) header via the PLAY method.

3.4.4 Subtitle

JMF does not support subtitle streaming. Therefore related modules are created without the use of JMF API. The subtitle stream is designed as a UDP "push" data stream on the server side. Data sysnchronization is done on the server side (media server). Whenever the time control phrase orders, the regarding text is pushed to the output stream. For the moment SRT file format supported.

3.4.5 SIP Proxy Work

Two services are developped on the SIP proxy:

First one is the “media-guide” subscription service which returns the media-guide URL that will enable web method invocation through SOAP.

Second service is bound to basic SIP call handling and the CPL processing. The service saves the rtspc url, keeps track of RTSP-C methods and maps it to a web url whenever the CPL rules permit the “Watch with Me” application explained on section 2.3.

4. RESULTS

4.1 Metrics

This section shows the methods and metrics used in the evaluation of the results. The variables used in calculations are:

X: Total message count in the overall execution.

C: Total commands issued.

M: Total message size of a control command sequence in Bytes

T: Total time the messaging uses the network.

A: Message Content size in Bytes

4.1.1 Messages per Control Command: X/C

This is the total number of messages triggered via a control command. It includes the actual command, proxied clone and regarding answer messages.

4.1.2 Bandwidth Utilization per Control Command: M/(T*C)

This is the control command size transmitted in unit time per a control command (Trick-play).

4.1.3 Data Carried per Control Command: A/C

This is the data transmitted per control command. The sample chose for the calculations is a video file (video only) and corresponding audio file. The details on the encoding formats and file sizes are listed in table 4.1.

Table 4.1 : Media File Properties

Video	
Encoding	H263
Duration	1464 sec
Resolution	352x288
Bitrate	9000 Kb/sec
Framerate	30 Frame/sec
Size	118 MB
Audio	
Encoding	MPEG-2 Audio
Frequency	44100
Channel	Strereo
Bitrate	128 Kb/sec
Size	22.3 MB

The control command does not have a proportional relationship with the actual media data transmitted. In order to calculate the data per control command, an assumtion is made on how many control commands are initiated in a unit time. The worse case scenario selected is 2 control commands per minute: assuming a viever may need to pause and play the media over 1 minute intervals. In the calculations the total media file size: 118+22.3 = 140.3 MB = 140300000 bytes.

4.1.4 Control Command Size per Data Carried: M/A

This is the control command size per data transmitted. The total size of the control messages and the sample media files selected will be used.

4.1.5 Control Command Per Session

This is the count of control command count per session. This covers the call setup, stream initiation, call closure and stream closure.

4.1.6 Optimum Bandwidth Utilization Per Session

Optimum metrics cover the condition that no trick-play is initiated throughout a media session.

4.1.7 Optimum Data / Control Throughout a Session

Optimum metrics cover the condition that no trick-play is initiated throughout a media session. Optimum data per control identifies the media size over the control messaging size throughout a session where no trick-play is initiated. The sample media is used in the calculations.

4.2 Preconditions and Assumptions

As introduced in Literature Work section, there are four approaches to provide SIP and RTSP interworking. However, not all are suitable to conform the considerations mntioned as requirements and are sutiable for basic SIP enabled networks.

Daul Stack approach, though being the most straightforward approach, neither does offer HTTP digest authentication as the way it is, nor provides mechanism to supply information for Streaming based SIP services. In this case, it can be filtered out and replaced with a more improved version as RTSP Event Notification to SIP.

Service Blending has more to offer for security considerations or streaming based SIP services. However, the concept is introduced for IMS networks, to interwork with CSCF nodes. The concept is still new and some nodes introduced are either high level design or porotyping stage. It is not introduced for basic SIP proxy networks, which form the majority of VOIP networks deployed all over the world.

Therefore, further analysis is done based on the two alternatives: All SIP Solution and RTSP Event Notifiction to SIP.

RTSP Event Notification concept introduced as an idea to enable to report RTSP states to SIP network. The concept has some requirements to fullfil security and NAT traversal considerations. Thus, the assumtions on how the regarding network whould be are listed below. The network model will be used in further analysis of the systems.

It is assumed that there would be an RTSP proxy which will lies on the service provider network along with SIP Application Server and Central Database.

RTSP Proxy is assumed to be able to reach subscriber passwords stored on the Central Database. By this capability, RTSP Proxy will have the capability to apply HTTP Digest authentication scheme on RTSP messages.

RTSP Proxy is also assumed to have capability to generate SIP NOTIFY messages per the RTSP control message it proxies to the media server. The subscription method is out of concept; it is another design item whose impact on the following analysis can be minimized. By simplicity, the method is aasumed implicit subscription which does not require SIP SUBSCRIBE messaging.

It is assumed the User Agent is capable of processing two protocols, SIP and RTSP at the same time. It is assumed that RTSP Outbound Proxy is also provisioned on the client, like the SIP Outbound Proxy.

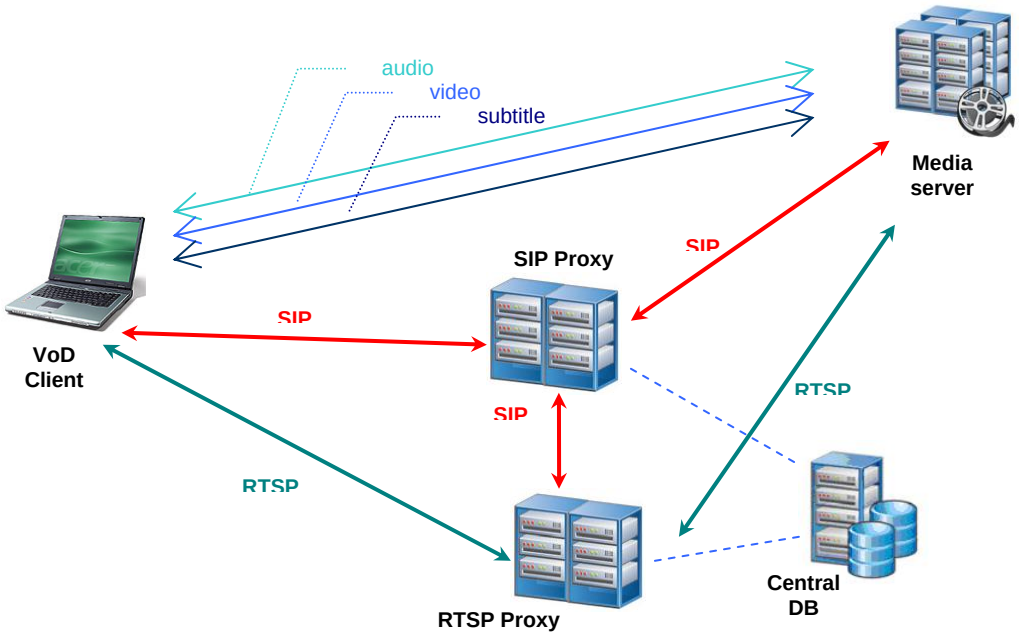

Figure 4.1 : RTSP Event Notification Network Model

RTSP messages used in calculations are derived from RTSP RFC [22]

SIP NOTIFY message is assumed to use xml format in the message body. As similarity considered, the "media-guide" NOTIFY message is used in the calculation considering usage and network similarity. Modifications are done based on the information required from the RTSP event: subscriber, RTSP URL and action are added as content.

Since Transfer related information already transmitted via SDP through SIP, SETUP message signalling is omitted: Whether to use SETUP in dual stack based implementation is a discussion point. It is assumed that the streaming is started via a PLAY command instead which also generates a SIP NOTIFY for RTSP event.

The same way as SETUP, TEARDOWN message is omitted as it has close relasionship with actual call session closure, which is already handled by BYE SIP message. It is assumed that BYE message and regarding 200 OK implies TEARDOWN.

```
NOTIFY sip:47.168.94.252:5060 SIP/2.0
From: "test1" <sip:test1@co10sesm2.com>;tag=1207894873716
To: "test1" <sip:test1@co10sesm2.com>;tag=z9hG4bK38533802
CSeq: 1 NOTIFY
Call-ID: 725532220163@192.168.2.4
Content-Length: 175
Content-Type: application/com.nortelnetworks.applications.rtsp-notify+xml
Contact: <sip:test1@co10sesm2.com:5060;maddr=47.168.47.70>
Max-Forwards: 20
Supported: com.nortelnetworks.firewall,nosec,join
Subscription-Expires: 3273
Event: media-guide
Subscription-State: active;expires=3273

<?xml version="1.0" encoding="UTF-8"?>
        <notify>
                <rtsp-event>
                        <subscriber>test1@co10sesm2.com</subscriber>
                        <URL>rtspc://mediaserver/Avatar_301</URL>
                        <action>PLAY</action>
                </rtsp-event>
        </notify>
```

Figure 4.2 : Sample Notify

4.3 Message Flows

4.3.1 RTSP-C Solution

4.3.1.1. Session Setup and Closure

Session setup is initiated via INVITE transaction. Setup is completed via ACK transaction. Session closure is handled via BYE transaction.

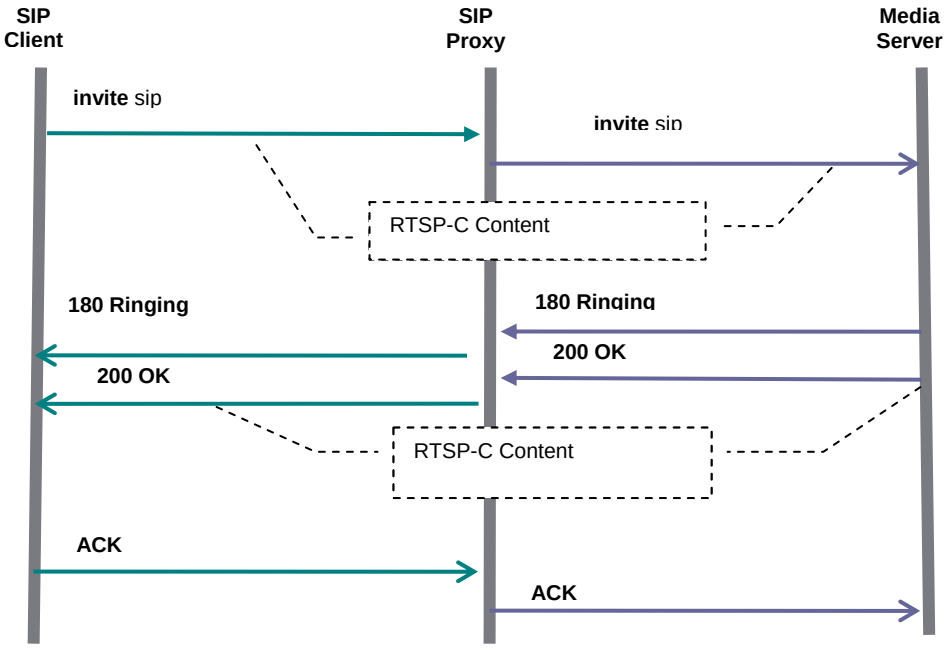

Figure 4.3 : RTSP-C Session Setup

Table 4.2 shows the messages, their size values and occurance counts per signalling flow. The values are taken from test calls.

Table 4.2 : RTSP Messaging Details

Message	Size (Byte)	Count	Size x Count
INVITE	1,038	2	2076
180 RINGING	569	2	1138
200 OK	1,361	2	2722
ACK	577	2	1154
BYE	350	2	700
200 OK	345	2	690
TOTAL	4,240	12	8,480

4.3.1.2. Mid Call (Trick-Play)

A sample call flow is selected for PLAY operation.

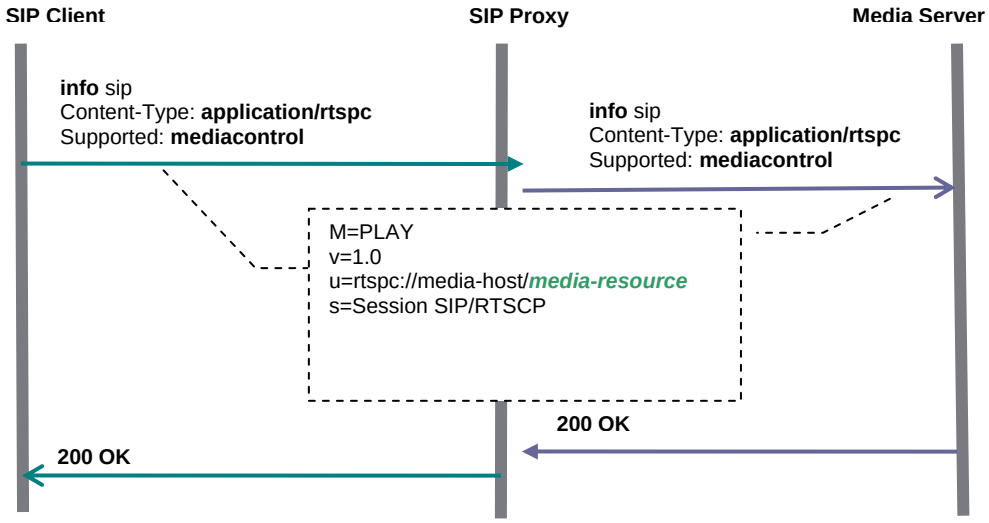

Figure 4.4 : RTSP-C Trick-Play

Table 4.3 shows the messages, their size values and occurance counts per signalling flow. The values are taken from test calls.

Table 4.3 : RTSP-C Trick-Play Details

Message	Size (Byte)	Count	Size x Count
INFO	498	2	996
200 OK	345	2	690
TOTAL	843	4	1,686

4.3.2 RTSP Event Notification to SIP

4.3.2.1. Session setup And Closure

Session setup is initiated via INVITE transaction. Setup is completed via ACK transaction. While RTSP PLAY transaction takes place, the RTSP proxy informs SIP Proxy on regarding RTSP event. Session closure is handled via BYE transaction.

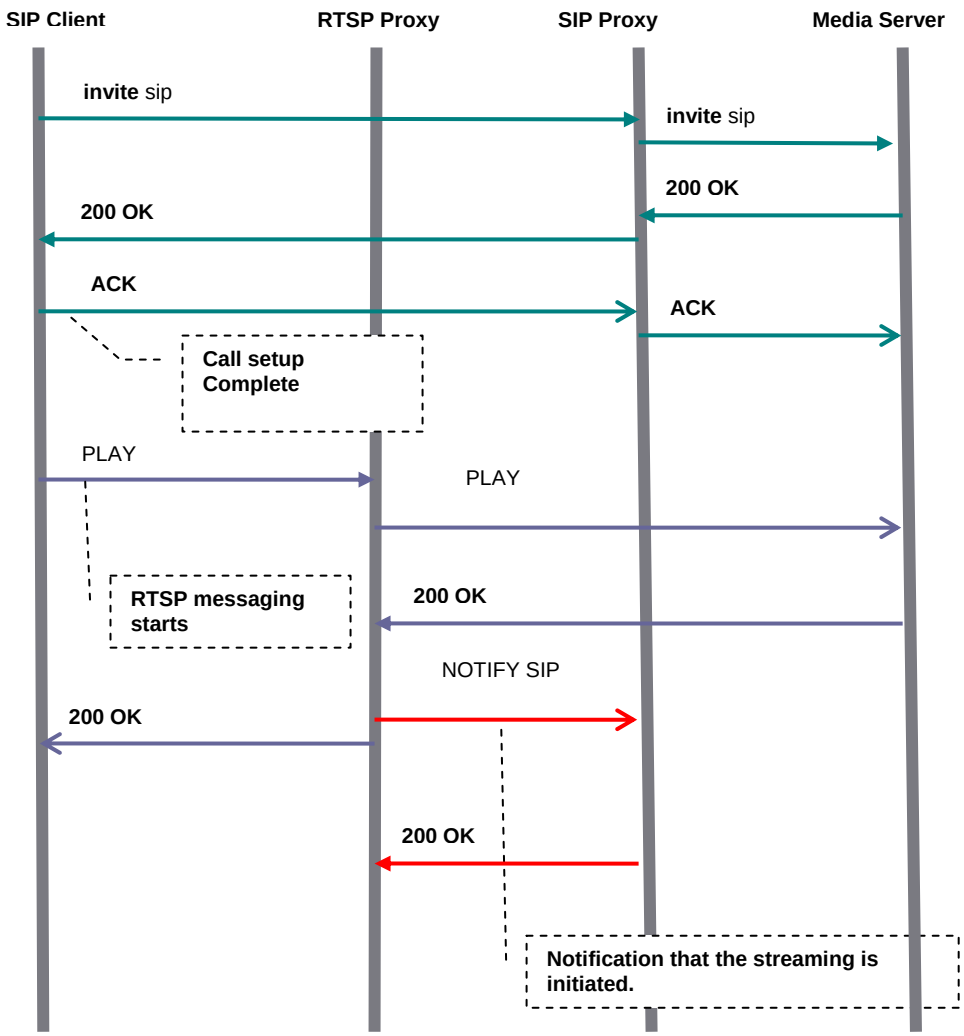

Figure 4.5 : RTSP Event Notification Session Setup

Table 4.4 shows the messages, their size values and occurance counts per signalling flow. The values are taken from test calls.

Table 4.4 : RTSP Event Notification Messaging Details

Message	Size (Byte)	Count	Size x Count
INVITE	741	2	1482
180 RINGING	569	2	1138
200 OK	1,076	2	2152
ACK	577	2	1154
PLAY	74	2	148
200 OK (PLAY)	26	2	52
NOTIFY	813	1	813
200 OK	345	1	345
BYE	350	2	700
200 OK	345	2	690
TOTAL	4,916	18	8,674

4.3.2.2. Mid-Call (Trick-Play)

A sample call flow is selected for PLAY operation.

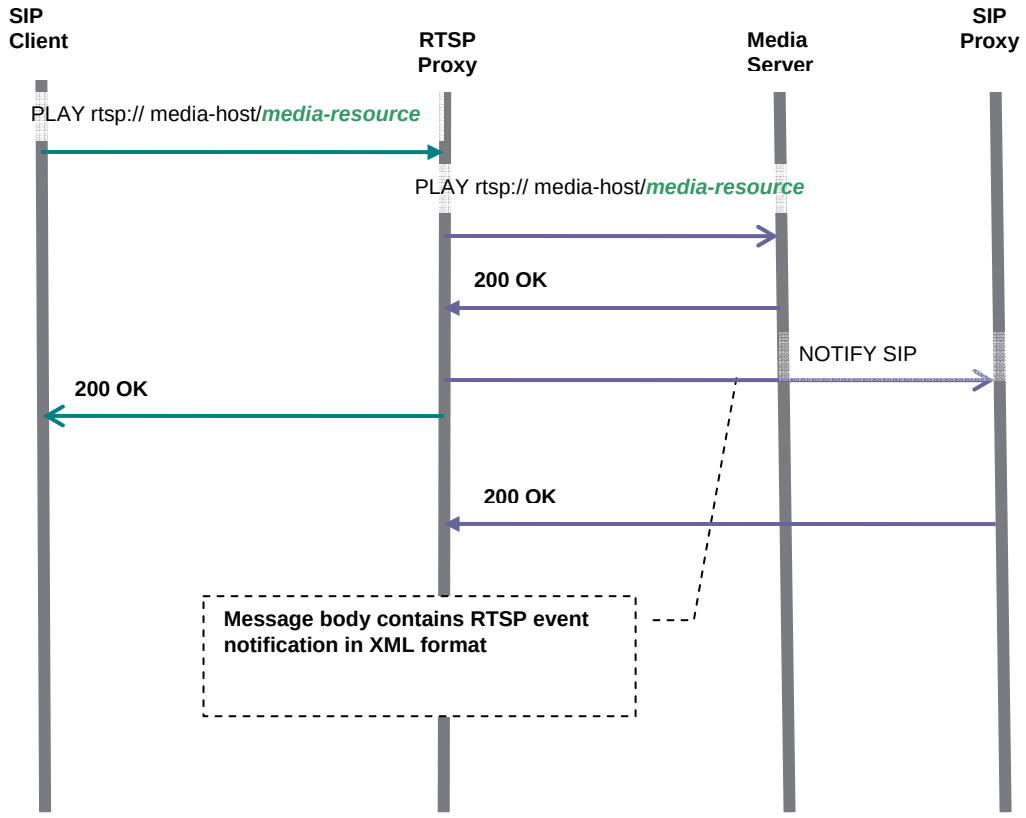

Figure 4.6 : RTSP Event Notification Trick-Play

Table 4.5 shows the messages, their size values and occurance counts per signalling flow. The values are taken from test calls.

Table 4.5 : RTSP Event Notification details

Message	Size (Byte)	Count	Size x Count
PLAY	74	2	148
200 OK (PLAY)	26	2	52
NOTIFY	813	1	813
200 OK	345	1	345
TOTAL	1,258	6	1,358

4.4 Numeric Analysis

Table 4.6 : Numeric Analysis

	RTSP-C	RTSP Event Notif.
Messages / Control Command	4	6
Bandwidth Utilization / Control Command (bytes)	1,686	1,358
Data Carried / Control Command (bytes)	2875000	2875000
Data Carried / Control Command Size	1705.22	2117.08
Control Command / Session	10	16
Optimum Bandwidth Utilization / Session (bytes)	8,480	8,674
Total Data / Session (bytes)	140,300,000	140,300,000
Total Data / Control Through a Session	16544.81	16174.78

Table 4.6 shows the some of the metrics and some associated data.

Some notes:

- In the calculation of Bandwidth Untilication / Control Command the T (time) value is taken 1 for simplicity. Assuming packet processing delay and network propagation is equivalent, the proportion on packet sizes is focused.
- Data Carried / Control Command as the same file and control command interval (1 min) selected.
- Data Carried / Control Command Size reflect the relation between the total control command size in bytes and media size in bytes (transmitted in 1 minute interval between 2 control commands).
- Optimum Bandwidth Utilization / Session use T (time) value 1 for simplicity.
- Control Through a Session / Total Data reflects the optimum values where no mid-call trick play is initiated.

The analysis for the above values indicate the following results.

The bandwidh utilization per media data values for:

- RTSP-C Solution: Varies between 16544.81 and 1705.22 for optimum session startup and selected mid-call worse case scenario.
- RTSP Event Nofification: Between 16174.78 and 2117.08 for optimum session startup and selected mid-call worse case scenario.

The bandwidth utilization rates considered, RTSP Event Notification is more efficient in the mid-call while RTSP-C is more efficient for session setup:

- RTSP-C/ RTSP Evt. Not. = 1.25 for mid call
- RTSP-C / RTSP Evt. Not. = 0.98 for session startup

When message counts considered, RTSP-C uses less messages than RTSP Event Notification to undermine the same functionality:

- RTSP-C / RTSP Evt. Not. = 0.67 for mid call
- RTSP-C / RTSP Evt. Not. = 0.62 for session startup

The results point out that the two methods have approximate values for network utilization and thus approximate network resource consumption. Considering the fact that the data carried is a lot larger than the control command sizes, the two methods can be regarded as equivalent. This shows us that there is no big difference between the two approaches when the actual transmitted media data is put into calculation. The methods should be further evaluated in other categories.

4.5 Usability Metrics and Evaluation

As introduced earlier this new model defines interoperability for SIP and RTSP protocols and has two main goals: security and a framework for streaming based SIP applications.

4.5.1 Security

Dual-stack approach does not define the presence of an RTSP proxy on the service provider network. Therefore, the authentication function of RTSP is left to the media server, which is the open item of this implementation.

RTSP-C, as a part of SIP message body, leaves the authentication function to SIP layer. If the authentication rules on the SIP proxy require info messages to be authenticate, this is done by the SIP proxy with no extra work. The authentication would be done on the service provider network as the SIP Application server is capable to reach the subscriber passwords stored on the central database. Also, it can adapt to TCP/TLS and IPSEC implementations of SIP with no additional work.

RTSP event notification to SIP model with the assumptions and preconditions already mentioned, can support HTTP digest authentication scheme which is already available for RTSP protocol. The authentication can be done on the service provider network as the RTSP proxy is capable to reach the subscriber passwords stored on the central database.

Service Blending options reviewed are designed to have interface with the CSCF, which already takes care of SIP authentication via HTTP Digest scheme. However, there is no reference on the authentication of RTSP signalling in those models.

Considering the facts mentioned, RTSP-C and RTSP event notification to SIP provide the most secure settings on equivalent grade. Service Blending is grey on this area but if it is assumed to use HTTP digest authentication, it presents security level equivalent to the other two.

4.5.2 Framework for Streaming Based SIP Applications

Dual-stack approach keeps SIP and RTSP on separate planes after the call setup. No other interaction is defined for SIP and RTSP; therefore, development of streaming based applications is not covered.

On the other hand, RTSP-C achieved this goal by putting the SIP proxy onto the media control path. “Watch with Me” application proves RTSP-C can be implemented to SIP networks with little modification and is suitable for post-answer streaming based applications. This approach also suitable for pre-answer service interactions like “Call Admission Control” services, in which RTSP content can be used in call admission policies.

RTSP Event Notification to SIP mechanism is specifically suggested for this purpose. SIP Application can keep track of streaming media and associated states by means of the NOTIFY messages it receive. This mechanism is suitable for post-answer service implementations.

Service Blending is specifically brought up to provide service interactions. SSV and MMP applications define some interaction for RTSP processing. Especially MMP, which has integrated RTSP proxy, introduces a solution to IMS network similar to RTSP Event Notification. Service Broker is actually developed for special protocol interactions like that of SIP and RTSP. For media streaming case, "snooping" of RTSP messages and sending the regarding information via NOTIFY messages to SIP Application Servers. By doing that, Service Broker achieves the basic functionality required for streaming based SIP applications. Yet it has not been defined how Service Broker is inserted on RTSP path and how the SIP Applications Servers implicitly or explicitly subscribe to Service broker on RTSP events of a specific subscriber. Furthermore, Service broker is introduced to Next Generation Networks (IMS) and does not offer solution to basic SIP Proxy systems.

RTSP-C and RTSP Event notification provide flexible platform for streaming based applications. However, different form RTSP event Notification, RTSP-C also provides capability for Call Admission type applications, as it lie on call setup path. Service Blending is a successful concept but introduced only to IMS networks yet.

4.5.3 Performance

As RTSP Event Notification keeps the processing of RTSP messages independent of the SIP network, network delay of the RTSP messages are not tied to the processing capabilities of the SIP proxy.

Since, RTSP-C messages are transmitted over INFO messages; they are queued and processed on the SIP proxy. This is a disadvantage of RTSP-C. The delay proven tolerable on normal conditions and is possible to reduce the delay on the SIP proxy by allocating higher priority threads on the SIP proxy.

It is not known how Service Broker operates on RTSP message snooping and if a relevant delay occurs.

In performance RTSP Event Notification has advantage over RTSP-C on the delay. However, with sufficient hardware and software adjustments, this disadvantage can be overcome.

4.5.4 Implementation

Dual stack approach is rather easy to implement as the two protocols are kept mainly in their own space, with minimal interaction.

RTSP Event Notification to SIP implementation has requirements on both the SIP Application server and RTSP proxy. The SIP Application server is required to handle the Sip NOTIFY messages containing RTSP events. It may also require to use a subscription method to trigger RTSP event notification. The RTSP Proxy implemented to the network si required to have an integrated SIP stack through which it will send event notifications. The clients and the media server is required to have dual-stack for SIP and RTSP handling.

As the implementation part of this study proves, RTSP-C is easily realized on existing sip networks, with little work on the SIP proxy, client and media server. This makes the solution more valuable in short term.

Service Blending solutions are only applicable to IMS networks. The solutions require interaction with various service nodes on the IMS network and are not only designed specific to SIP and RTSP but a various set of features on the network.

In conclusion, as RTSP-C has minimum network requirements, it is easiest to implement to an existing SIP system.

5. FUTURE WORK

One part of the future work of this study is the interaction of advanced SIP services and RTSP. In this category, joining a conference through SIP, all kinds of transfer scenarios, call park scenarios and services based on PSTN-SIP convergence can be mentioned.

One other area of future work is the performance improvement of INFO processing on SIP proxies. Enhancements on SIP Proxy can be done to provide higher priority tasks to INFO messages in specifically for VoD interworking.

Integration of RTSP-C with other IPTV variant applcations like Tripple Play is also another future work area.

For RTSP-C prototype subtitle and audia language change during session can be implemented.

A prototype of RTSP Event Notification can be prepared a more detailed comparison of RTSP-C and RTSP event notification can be done. On this concept, propagation and accuracy delays can be compared.

IMS integration of RTSP is another future item. In that work, a network model and prototyping can be done on Service Broker.

6. FINAL CONCLUSION

The results proved that RTSP-C convergence model is applicable on SIP networks to provide interoperability with RTSP. The method has proven effective when evaluated against security, performance, implementation aspects. Though the model requires processing of Trick Play's on the SIP proxy, this has proven to cause tolareable delay which can be improved on adjustments on the SIP proxy. The method especially stands out as the implementation is easier than the other approaches to the best of our knowledge and when providing framework for streaming based services. As the method allows pre-answer feature intractions like Call Admission Control, it provides a more flexible platform than the closest rival option, RTSP Event Notification. With all aspects evaluated RTSP-C provides the more advantages than the rest of the options presented in the thesis.

REFERENCES

[1] **Friedrich, O., Al-Hezmi, A., Arbanowski, S., Magedanz, T.**, 2007 , Next Generation IPTV services for an extended IMS architecture, Autonomous Decentralized Systems, 2007. ISADS '07. Eighth International Symposium held on 21-23 March 2007, Page(s):429 – 436, Digital Object Identifier 10.1109/ISADS.2007.52

[2] **S. Whitehead, M.J. Montpetit, X. Marjou, S. Ganesan, D. Ress, D. Goodwill**, 2006, An Evaluation of Session Initiation Protocol (SIP) for use in Streaming Media Applications,

http://tools.ietf.org/id/draft-whitehead-mmusic-sip-for-streaming-media-02.txt

[3] **Bodzinga, A., White, S.**, 2008, Interworking IPTV Services with IMS

[4] **Ensor, J. R., Hoffman, M., Rimac, I.**, 2006, Blending IPTV Services, IPTV Workshop jointly held with the 15th International World Wide Web Conference in Edinburgh, Scotland on May 24, 2006

[5] **RFC3427**, 2002, Change Process for the Session Initiation Protocol (SIP)

[6] **RFC3325**, 2002, Private Extensions to the Session Initiation Protocol (SIP) for Asserted Identity within Trusted Networks

[7] **RFC4964**, 2007, The P-Answer-State Header Extension to the Session Initiation Protocol for the Open Mobile Alliance Push to Talk over Cellular

[8] **RFC4457**, 2006, The Session Initiation Protocol (SIP) P-User-Database Private-Header (P-Header)

[9] **RFC3966**, 2004, The tel URI for Telephone Numbers

[10] **RFC3261**, 2002, SIP: Session Initiation Protocol

[11] **RFC4904**, 2007, Representing Trunk Groups in tel/sip Uniform Resource Identifiers (URIs)

[12] **RFC2046**, 1996, Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types

[13] **RFC3372**, 2002, Session Initiation Protocol for Telephones (SIP-T): Context and Architectures

[14] **RFC3204**, 2001, MIME media types for ISUP and QSIG Objects

[15] **RFC2976**, 2000, The SIP INFO Method

[16] **RFC3265**, 2002, Session Initiation Protocol (SIP)-Specific Event Notification

[17] **RFC3842**, 2004, A Message Summary and Message Waiting Indication Event Package for the Session Initiation Protocol (SIP)

[18] **Khan, S. Q., Gaglianello, R., Luna, M.**, 2007, Experiences with Blending HTTP, RTSP, and IMS, IEEE Communication Magazine March 2007

[19] **3GPP TR 23.810**, version 9.0, 2008, Study on architecture impacts of Service Brokering

[20] **Lee, Chae-Sub**; 2007, IPTV over Next Generation Networks in ITU-T

Broadband Convergence Networks, 2007. BcN '07. 2nd IEEE/IFIP International Workshop on 21-21 May 2007 Page(s):1 - 18

[21] **RFC2617**, June 1999, HTTP Authentication: Basic and Digest Access Authentication

[22] **RFC2326**, 1998, Real Time Streaming Protocol (RTSP)

[23] C. Boulton, Ed., Rosenberg, J., Camarillo, G., 2007, Best Current Practices for NAT Traversal for SIP, http://www.ietf.org/internet-drafts/draft-ietf-sipping-nat-scenarios-07.txt

[24] **Westerlund, M., Zeng, T.**, 2007, An Network Address Translator (NAT) Traversal mechanism for media controlled by Real-Time Streaming Protocol (RTSP)

[25] **RFC3880**, 2004, Call Processing Language (CPL): A Language for User Control of Internet Telephony Services

[26] URL: http://www.dailyiptv.com/whitepaper/iptv-explained/

[27] **Tanenbaum, A. S.**, 2003, Computer Networks, 4th Edition

[28] **RFC2387**, 1998, The MIME Multipart/Related Content-type

[29] URL: http://java.sun.com/products/java-media/jmf/

[30] URL: http://java.sun.com/products/java-media/jmf/2.1.1/guide/JMFArchitecture.html

[31] URL: http://fobs.sourceforge.net/features.html

[32] URL: http://www.mjsip.org/

[33] URL: http://products.nortel.com/go/product_content.jsp?parId=0&segId=0&catId=A& prod_id=47181

APPENDICES

APPENDIX A.1 HTTP Digest Authentication Scheme

Authentication in HTTP is defined in RFC 2617. There are two authentication schemes mentioned in RFC 2617: “Basic” and “Digest”. “Basic” authentication scheme is not considered to be a secure method of user authentication, as the user name and password are passed over the network in an unencrypted form. Due to the security weakness, its usage in SIP has been deprecated. RFC 3261 suggests “Digest” authentication scheme.

The Digest authentication scheme challenges the UA using a nonce value. A valid response contains a checksum (by default, the MD5 checksum) of the username, the password, the given nonce value, the HTTP method, and the requested URI. In this way, the password is never sent in the clear.

When a proxy challenges a UA request with a “407 Proxy Authorization Required”, it includes a “proxy-authenticate” header containing realm, nonce, stale, algorithm and qop subfields. Below functions of these parameters are described in detail:

realm: A string to be displayed to users so they know which username and password to use. It defines the protection space.

nonce: A server-specified data string which should be uniquely generated each time a request is challenged with a 401 or 407 response. The contents of the nonce are implementation dependent. The quality of the implementation depends on a good choice.

stale: A flag, indicating that the previous request from the client was rejected because the nonce value was stale.If stale is TRUE(case-insensitive), the client may wish to simply retry the request with a new encrypted response, without reprompting the user for a new username and password. The server should only set stale to TRUE if it receives a request for which the nonce is invalid but with a valid digest for that nonce (indicating that the client knows the correct username/password). If stale is FALSE, or anything other than TRUE, or the stale directive is not present, the username and/or password are invalid, and new values must be obtained.

algorithm: A string indicating a pair of algorithms used to produce the digest and a checksum. It's default value is "MD5". If the algorithm is not understood, the challenge should be ignored and a different one used, if there is more than one).

qop: It is an optional directive in order to be backwards compatible with RFC 2069. If present, it is a quoted string of one or more tokens indicating the "quality of protection" values supported by the server. The value "auth" indicates authentication; the value "auth-int" indicates authentication with integrity protection

Whenever a client gets a challenge, it re-sends the request including proxy-authorization header which contains realm, username, nonce, uri, response, algorithm, cnonce, qop and nonce-count (nc) subfields.

uri: Request-uri is duplicated to here.

qop: Indicates what "quality of protection" the client has applied to the message.

cnonce: It is an opaque quoted string value provided by the client and used by both client and server to avoid chosen plaintext attacks, to provide mutual authentication, and to provide some message integrity protection. This MUST be specified if a qop directive is sent (see above), and MUST NOT be specified if the server did not send a qop directive in the proxy-authenticate header field

nonce-count: is the hexadecimal count of the number of requests (including the current request) that the client has sent with the nonce value in this request. It is used against replay attacks. This MUST be specified if a qop directive is sent (see above), and MUST NOT be specified if the server did not send a qop directive in the proxy-authenticate header field.

response: A string of 32 hex digits computed as defined below, which proves that the user knows a password. If the request-digest calculated by the server using the credentials and the given proxy-authentication header matches the response field, the credentials used by the UAC are valid.

Below is the procedure showing how the request-digest is calculated:

In the formulas, the "MD5" and "MD5-sess" digest algorithms are denoted as:

H(data) = MD5(data)

and

KD(secret, data) = H(concat(secret, ":", data))

If the "qop" value is "auth" or "auth-int":

request-digest = <"> < KD (H(A1), unq(nonce-value)
":" nc-value
":" unq(cnonce-value)
":" unq(qop-value)
":" H(A2)
) <">

If the "qop" directive is not present (this construction is for compatibility with RFC 2069):

request-digest = <"> < KD (H(A1), unq(nonce-value) ":" H(A2)) > <">

If the "algorithm" directive's value is "MD5" or is unspecified, then

A1 is:

A1 = unq(username-value) ":" unq(realm-value) ":" passwd

where

passwd = < user's password >

If the "algorithm" directive's value is "MD5-sess", then A1 is calculated only once - on the first request by the client following receipt of a WWW-Authenticate challenge from the server. It uses the server nonce from that challenge, and the first client nonce value to construct A1 as follows:

A1 = H(unq(username-value) ":" unq(realm-value)

":" passwd)

":" unq(nonce-value) ":" unq(cnonce-value)

This creates a 'session key' for the authentication of subsequent requests and responses which is different for each "authentication session", thus limiting the amount of material hashed with any one key. (Note: see further discussion of the authentication session in section 3.3.) Because the server need only use the hash of the user credentials in order to create the A1 value, this construction could be used in conjunction with a third party authentication service so that the web server would not need the actual password value. The specification of such a protocol is beyond the scope of this specification.

If the "qop" directive's value is "auth" or is unspecified, then A2 is:

A2 = Method ":" digest-uri-value

If the "qop" value is "auth-int", then A2 is:

A2 = Method ":" digest-uri-value ":" H(entity-body)

BIOGRAPHY

İbrahim Bilgin was born in Balikesir, TURKEY in 1982. He graduated from Soke Hilmi Firat Anatolian High School in 2000. In 2004, he received B.Sc. degree in Computer Engineering from Istanbul Technical University. In 2004, he started the Computer Engineering M.Sc. program in Istanbul Technical University. He is currently working as a software design engineer in Nortel Networks Netas. His research interests are telecommunication and Voice Over IP systems design, cryptography, and Object Oriented Programming.