Master's thesis

NTNU
Norwegian University of Science and Technology

Fairooz Zafar

IMS Service Orchestration and Interaction Challenges

Master's thesis in Master's of Science in Communication Technology
Supervisor: Mazen Malek Shiaa
June 2019

NTNU
Kunnskap for en bedre verden

Title: IMS Service orchestration and interaction challenges
Student: Fairooz Zafar

Problem description:

IP Multimedia Subsystem (IMS) is one of the most recognized systems to provide blended multimedia services to mobile and fixed networks. It was adopted with the intention to increase interoperability among network elements and decrease complex and non-extensible elements from the network. However, feature interactions, in many cases, make it difficult to achieve these intentions. Even though much work has been done in this regard, service capabilities are still facing interaction issues and many telecommunication operators (telcos) are dealing with the issue of limited service orchestration. The operators have come up with solutions that tend to minimize the problems but there are still many limitations that need to be addressed. Service Capability Interaction Manager (SCIM) is one proposed solution by the 3GPP organization. However, there is no clear specifications on SCIM and there no evaluation criteria are defined, which is potentially the reason why it is not widely used by telcos. This has led to various adaptations of the SCIM by the industry. The release of 5G soon will call for more complex services and compositions to which the existing solutions have a possibility of falling short.

The aim of this thesis is to formulate criteria for evaluation of a SCIM, to define functional and non-functional requirements of a SCIM, and to evaluate existing SCIM products. In addition, a prototype SCIM will also be developed based on the formulated criteria.

To achieve these goals, simple services and a prototype SCIM will be developed as SIP servlets deployed on Metaswitch Rhino TAS (Telephony Application Server) that will aim at solving some of the existing service interactions. The formulation of criteria will be based on research on existing specifications and products in the field by comparing the functional and non-functional properties. Alongside, Rhino SIS, an existing service broker similar to a SCIM, will be used as an experimentation tool. This is also one of the tools that will be evaluated against the proposed criteria of evaluation at the end of this thesis.

The specific tasks outlined for this thesis are as follows:

– Review and derive functional and non-functional requirements for a SCIM

– Study existing standardization documentation and formulate criteria for evaluation of a SCIM

- Define service interaction scenarios to be used as example scenarios in the implementation
- Become familiarized with TAS and Rhino SIS tools and experiment with selected service interaction scenarios
- Develop a prototype SCIM based on the derived functional and non-functional requirements
- Test and evaluate the prototype using the chosen interaction scenarios
- If possible, evaluate existing SCIMs against the formulated evaluation criteria

Responsible professor:	Mazen Malek Shiaa, NTNU
Supervisor:	Bjørn Gulla, Kornschnok Dittawit, Gintel AS / NTNU

Abstract

IP Multimedia Subsystem (IMS) is part of the 3G mobile network that allows deployment and blending of multimedia services for its users. This subsystem was specified as a standard by the standardization organizations such as the 3rd Generation Partnership Project (3GPP), Internet Engineering Task Force (IETF). These organizations also include the standard for the services as well as all other components that are part of the system. IMS is aimed to provide multimedia services to all generations of networks and devices. Service compositions allow the users to get the blended services to subscribe to with IMS. But all compositions do not always lead to desirable outcomes leading to feature interactions. The network operators use their monolithic or man-in-the-middle approach to handle these interaction issues which are often constrained and expensive. To address the issue of feature interaction, the 3GPP proposed to place an entity known as the Service Capabilities Interaction Manager (SCIM) between the S-CSCF and the Application Servers(ASes). With no specifications of this entity provided, many proposals for the specifications came from the research community and SCIM products are developed by the telecom vendors. Due to no standardization, these products are vastly different from one another and often expensive which has made many telecom operators reluctant to include it in their network. This thesis investigates the standards and research available on the topic of feature interaction along with the SCIM products to understand the advantages and limitations. The information from the investigations are then utilized to make proposals for both functional and non-functional requirements that a SCIM should hold. Furthermore, two existing SCIM products are evaluated using the requirement suggestions.

Preface

This master's thesis is a representation of my work conducted in Spring 2019 as part of my M.Sc. degree in Communication Technology from the Department of Information Security and Communication Technology at the Norwegian University of Science and Technology (NTNU). The work presented here is an extension of the work done in the specialization project carried out in Autumn 2018.

It was my personal goal to get familiar with the practicalities of IMS, services and the protocols related to them as I find the technologies and services related to the telecommunication field very interesting and intriguing. The research on the topic and experimentation with tools that are commonly used in this field has been a great learning experience.

Even though the work has been challenging at times, it has taught me a lot about the tools and protocols such as SIP. In addition, the construction of a report to represent the knowledge I have gathered during the semester, has proven to be more demanding than anticipated.

Acknowledgements

I would like to take this opportunity to express my deepest gratitude to my supervisor Mazen Malek Shiaa at NTNU and one of my co-supervisors Bjørn Gulla at Gintel AS. They made this project possible for me and it was an amazing experience to learn on this topic since the specialization project in Autumn 2018. I want to specially thank my other co-supervisor, Kornschnok Dittawit at Gintel AS who has helped me through the entire year with her guidance and motivation for me to reach my goal.

I would further like to thank Jan Wedvik at Gintel AS for being kind enough to help me with technical difficulties during the practical part of the thesis.

The staff at the department of the Information Security and Communication Technology, for taking the time for the additional work to make this project possible, I am grateful.

Last, but not the least, I want to thank my friends and family for their continuous support and encouragement throughout the year.

Contents

List of Figures

List of Algorithms

List of Acronyms

3GPP The Third Generation Partnership Project.

AS Application Server.

CB Call Barring.

CFU Call Forwarding Unconditional.

CSCF Call Session Control Function.

HSS Home Subscriber Server.

I-CSCF Interrogating- CSCF.

IETF Internet Engineering Task Force.

iFC Initial Filter Criteria.

IMS IP Multimedia Subsytem.

ITU-T International Telecommunication Union- Telecommunication.

MRF Multimedia Resource Function.

MRFC Multimedia Resource Function Controller.

MSC Mobile Switching Center.

NTNU Norwegian University of Science and Technology.

OSA-SCS Open Service Access–Service Capability Server.

P-CSCF Proxy- CSCF.

PSTN Public Switched Telephone Network.

SB	Service Broker.
SCIM	Service Capability Interaction Manager.
S-CSCF	Serving- CSCF.
SIP	Session Initiation Protocol.
SIP URI	SIP Uniform Resource Identifier.
SIS	Service Interaction SLEE.
SLEE	Service Logic Execution Environment.
SLF	Subscriber Location Function.
telcos	Telecommunication Operators.
TR	Technical Requirement.
TS	Technical Specification.
UE	User Equipment.

Chapter 1 Introduction

Multimedia services are an attraction to the telecommunication sector. IP Multimedia Subsystem (IMS) was adopted by the 3GPP and ITU-T as a standard to have a common architecture for all devices and networks (e.g., 2G, 3G, 4G, LTE, including the Next Generation Networks (NGN)). It is now used by many telecommunication operators (telcos) to provide combined services to its users. IMS is able to avoid vendor lock-ins which results in the inter-operable and extensible network elements. These properties are making IMS popular everyday.

Composition of several services is required to provide advanced services with IMS. The means for controlling consecutive interaction between services is known as service orchestration [3GP07] which is achieved with the use of Initial Filter Criteria (iFCs) [3GP12b]. IFC is the collection of triggers that determine when a SIP(Session Initiation Protocol) request is forwarded to the Application Server(AS) that will be providing the service [CGM09]. Often service orchestrations face challenges known as *feature interaction*, which means that services when deployed in a session, can be processed separately and independently but creates conflicts when running together [3GP07], which is making service compositions limited. The need for new and attractive service compositions has made it necessary to look into the existing system and find out a solution that will help the case.

This chapter gives an overview of the thesis describing the background and motivation for the thesis. It also mentions the methodologies that will be used to reach the research goals of this thesis work.

1.1 Background

The 3rd Generation Partnership Project (3GPP) in accordance with Internet Engineering Task Force (IEFT) adopted IMS to deliver multimedia services to a wide range of users. This adoption was part of the core network evolution from circuit-switching to packet-switching and since then, IMS has become the core component within 3G, cable TV and next generation fixed telecommunication networks. [3GP00]. Session Initiation Protocol (SIP) is the main signaling protocol for IMS. This system aimed to assist telecos to deliver next generation interactive and inter-operable services cost-effectively and provide flexibility of the internet. The prime motivation in the adaptation of IMS is to avoid vendor lock-ins known from various approaches which lead to 1) lack of interoperability among network elements and 2) complex and non-extendable (monolithic) elements like Mobile Switching Centre (MSC). These resulted in constraints in evolution and innovation.

Even though IMS is a global success, over the years with increase of dependency on multimedia services, complex service orchestrations in many cases, face challenges that cannot be handled by the standard architecture. Hence, this architecture has been modified over time to cope with the complexities and the needs of the operators and to extended service capabilities outside of the framework with ease [HC09]. Serving-Call Session Control Function (S-CSCF) in the IMS control layer invokes the correct Application Server (AS) to provide services to the subscriber. AS is a SIP entity that hosts and executes IMS services [3GP13] , [CGM09]. Service profiles of users are transferred to the S-CSCF from the Home Subscriber Server (HSS) over the Cx interface (Diameter based). [KG08] Diameter is an authentication, authorization, and accounting (AAA) protocol. Service profiles are composed of a list of initial filter criteria (iFC) which are processed in a prioritized order by the S-CSCF. IFC allows a set of services to be invoked in defined circumstances. S-CSCF executes the services through a series of SIP messages(like, INVITE, REGISTER etc.). A powerful mechanism like iFC still has its limitations which makes it unable to solve many feature interactions that started the discussion of having solutions over the iFC that would allow the telcos to still be able to combine services they want by solving the feature interactions. One of the proposed solutions is an entity known as the Service Capability Interaction Manager (SCIM) [Gro] which is still without any standardization. The SCIM is also often known as a Service Broker (SB). This thesis focuses mainly on this entity investigating it as well as proposes some specifications that any SCIM entity should possess.

1.2 Motivation

The work on IMS has been done for over a decade but there is still no standard for SCIM which would specify certain criteria for how services should be handled by the SCIM. Even though a few operators have included SCIM in their system, most are still weighing the pros and cons of investing in a SCIM; if it would allow many flexible service composition, if it is worth investing in, if the new service capabilities will give back enough revenue etc. Without a standard, there is no baseline for the evaluation of any SCIM. This might increase the value of work that can be done in this regard. If successfully conducted, this can be of interest for the stakeholders i.e., telco operators,telco equipment manufacturers/ vendors, service developers, system integrators, etc. A more specific SCIM standardization might increase service compositions and lessen feature interactions. The development of 5G networks will, in future, also demand for services with complexities higher than what have been encountered till now. A look into the future of IMS, its service composition and SCIM seems promising. Hence, the objective of this project is to look into the the IMS architecture and its components. Finding more about service compositions, its orchestration and feature interactions is necessary to identify limitations, its reasons and possibilities to find an alternative standard solution. SCIM/ service broker approach seems to be a promising path forward to overcome feature interactions issues for this generation of telecommunications as well as for the next generations.

1.3 Methodology

This section specifies the methods that will be used to find suitable answers for the research goals set out for this project. Tools to be used and steps to be taken are enumerated.

Literature Review/ Background Research

The project addresses the service interaction challenges, opportunities and existing solutions to increase multimedia service capabilities. A series of 3GPP and IETF standards are studied to investigate the state-of-the-art IMS architecture and service capability standards. More thorough studies on the standards are done to find the technical specifications and limitations of the standards. Academic and technical research papers along with books published on the topic of service capabilities, interaction challenges and solution proposals were also studied. White papers and technical specification guides were used to know and learn the tools that is planned to be used for the experimental implementation. Even through SCIM related standards/ background is limited, efforts were made to understand what is expected of the component and how it can be evaluated.

Technical Evaluation

The first instrument that was used to get information regarding the working approaches for service orchestration was interviewing a technical team from Gintel AS, who have been working in this field for quite some time. They provided technical information regarding the monolithic and man-in-the-middle approach used by the telcos now. Further evaluation of existing SCIM products like Rhino SIS [Met11b] and Lucent Service Broker™ [KRA06] were done by studying the technical specifications as well as white papers. Further work with the Rhino SIS tool was done to identify the techniques used to solve feature interaction. This knowledge was utilized during the formulation of the proposal for a standard SCIM product. These two products were also evaluated on the basis of the proposals for a new SCIM.

Experimental Implementation

To propose and assess any criteria for evaluation, there is a need for testing. For this, SIP servlet applications were created and deployed that pose a feature interaction scenario. The scenario was solved using the Rhino SIS tool. A test/lab environment for this practical part was completed under the supervision of a the technical team of a Norwegian company active in this field – Gintel AS. This experiment gave insight on a functional SCIM and exposed the limitations and strengths of it as well. The results were used as inspiration for the proposals of the functional and non-functional requirements of a standard SCIM.
The problem description also mentions to implement a prototype SCIM using the derived requirements, but due to the lack time and expertise this was not possible. Implementing a prototype SCIM would require magnitudes of technical knowledge on the subject that was not possible with the duration of this thesis.

1.4 Thesis Outline

The entire thesis is divided into the following chapters: the standard architecture of the IMS and its limitations in Chapter 2, moves on to feature interaction scenarios and existing solutions to them in Chapter 3. The implementation of the interaction scenario and a SCIM product to solve the feature interaction is described in Chapter 4, the results of the implementation is elaborated in Chapter 5, the proposals of requirements for a SCIM is drawn out in Chapter 6 and the conclusion of the thesis is in Chapter 7.

Chapter 2 IMS Service Architecture

A tremendous amount of effort has been invested in the development of the technical specifications of IMS. The standard architecture and its service concepts are now accepted across the globe and implemented by many telecos. It is imperative to take a look into the standardization of IMS to understand the working mechanism of this complex system. This chapter explains the architecture layers, protocols as well as service concepts and profiles of IMS.

At the end of this chapter, the limitations of the architecture and the concepts of IMS are highlighted as a precursor for the next chapter while deals with the service interaction issues in IMS.

2.1 Protocols

Session Initiation Protocol (SIP)

[IET02], [3GP12b] An application layer control protocol capable of establishing, modifying and terminating multimedia sessions. It supports five facets of establishing and terminating multimedia communications, such as user location, availability, capabilities, setup and management. But SIP does not provide services, rather it provides the primitives that can be used to implement different services and it works with both IPv4 and IPv6.

SIP was chosen as the session control protocol for multiple reasons; SIP makes it easy to create services; it is based on HTTP allowing developers to use all service frameworks developed for HTTP such as Java servlets [CGM09]. In the IMS, all messages through the network from one terminal to the other are SIP messages.

Diameter

[IET03b] The AAA (Authentication, Authorization, and Accounting) protocol in the IMS was chosen to be Diameter. Diameter is widely popular on the Internet to perform AAA [CGM09]. IMS uses Diameter in a number of interfaces but not all. It is mainly used during the session setup (communication with the HSS) and to perform credit control accounting. The other elements of the IMS network uses Diameter to upload and download to and from the HSS (Home Subscriber Server) /SLF (Subscription Locator Function) [KG08]. Diameter has two interfaces call Cx and Dx that are used to communicate between the databases and IMS core network to access subscriber-related information.

Other Protocols

[CGM09] Among the other protocols, RTP (Real-Time Transport Protocol) [IET03a] and RTCP (Real-Time Control Protocol) are used to transport real-time media, such as video and audio.

2.2 Databases

2.2.1 Home Subscriber Server(HSS)

The HSS is the central repository for user-related information which is technically an evolution of the Home Location Registrar (HLR) and the Authentication Center (AUC) of the GSM node [CGM09]. This database stores all subscription-related information (user/ service profiles) of all users including location, security (both authentication and authorization information), S-CSCF assigned to the user. It also contains the required information to handle any multimedia session for the user.

It is very common for a network to contain a single HSS where all user information is stored, unless the number of subscribers is too high to be handled by a single HSS. In any case, the information related to a single user is stored in one particular HSS [CGM09]. If the network is composed of multiple HSS, the network also contains a Subscribed Locator Function (SLF).

2.2.2 Subscriber Locator Function(SLF)

An SLF is included in the network when there exists multiple HSSs. The SLF is a database that maps the users' addresses to HSSs that hold the subscriber data for that user [Acc07]. The SLF also does not perform any logic on its interfaces, instead replies with a redirect message specifying the address of the HSS.

2.3 IMS Core Architecture

The detailed outline of the service architecture of IMS has been described in the 3GPP technical specification TS 23.002 on Network Architecture. Service architecture refers to the mechanisms, configurations and interfaces between and within the core network in the IMS needed to provide multimedia services. The entities related to IMS such as CSCF, MGCF, MRF, etc. as defined in the stage 2 of the IM subsystem TS 23.228.

The architecture of IMS is divided into three layers, namely, access and transport layer, control layer and service layer. A simplified architecture diagram has been provided in Figure 2.1 showing these layers and its components. As mentioned earlier, IMS is intended to provide multimedia services to all devices with IP domain. For this reason, the separation between transport layer and control layer was necessary. This separation allows the exploitation of the IP infrastructure which leads IMS to be an all-IP solution [Acc07]. The description of the layers and it components are provided in the following subsections.

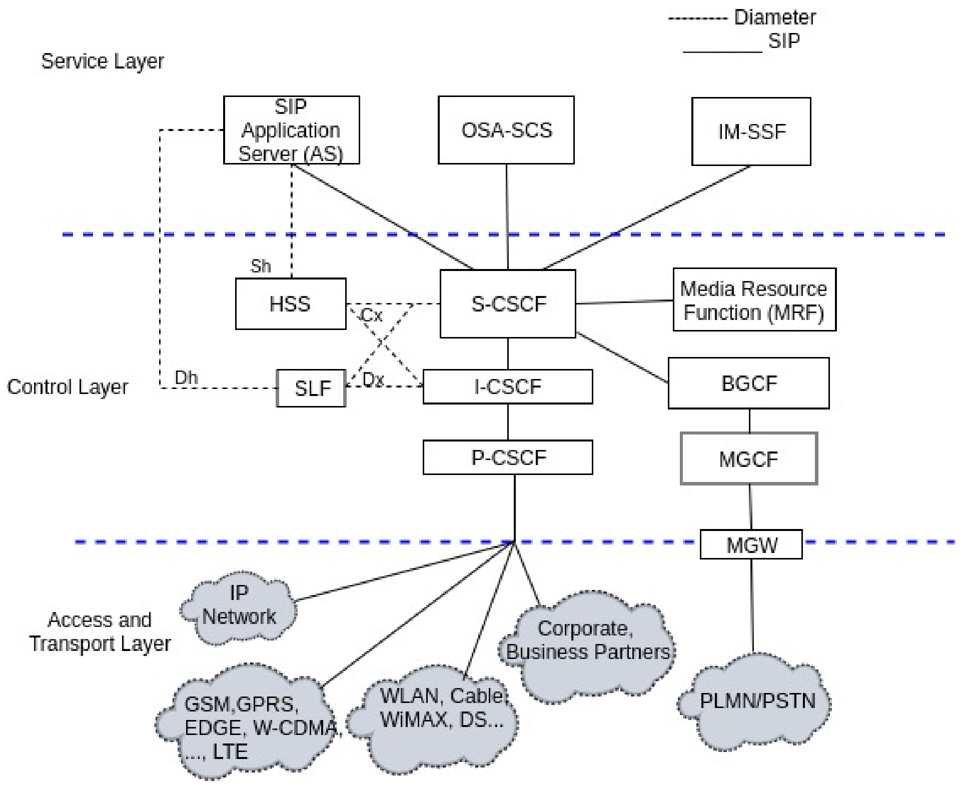

Diameter
SIP
Service Layer
SIP Application Server (AS)
OSA-SCS
IM-SSF
Sh
HSS
Cx
S-CSCF
Media Resource Function (MRF)
Control Layer
Dh
SLF
Dx
I-CSCF
BGCF
P-CSCF
MGCF
MGW
IP Network
Access and Transport Layer
Corporate, Business Partners
GSM,GPRS, EDGE, W-CDMA, ..., LTE
WLAN, Cable, WiMAX, DS...
PLMN/PSTN

Figure 2.1: Standard Architecture of IMS (Simplified)

2.3.1 Access and Transport Layer

This is the network-access layer that allows all devices and user equipment to connect to the IMS network by establishing IP connectivity for the devices.[KG08]. The transport layer consists of various technologies, for example, fixed access (DSL, cable modems, Ethernet), mobile access (wide-band code-division multiple access [WCDMA], CDMA-2000, Global Packet Radio Service [GPRS], LTE), wireless access (such as wireless local area network [WLAN] or WiMax) as well as Public Switched Telephony Network [PSTN].

2.3.2 Control Layer

The control layer orchestrates all logical connections between various network elements. It provides registration of end points, routing of SIP messages and overall coordination of media and signalling gateways [Dia09]. The two most important elements of the control layer are the Call Session Control Function (CSCF) and Home Subscriber Server (HSS).

Call Session Control Function (CSCF)

CSCF is composed of several types of SIP servers that process SIP signals in the IMS infrastructure. There are three types of CSCFs in the IMS domain depending on the functionality they provide, which are described as follows:

Proxy- CSCF (P-CSCF)

This is the first point of contact between the IMS terminal and IMS network [CGM09]. It can be located either in the home network or the visited network. This acts as both inbound and outbound proxy server and all SIP message request traverse through the P-CSCF. This also forwards the messages to appropriate destinations which can either be the IMS terminal or the network. A proxy is assigned to an IMS terminal during the registration process and it stays the same during the registration process but the network can have multiple of these for scalability and redundancy. The P-CSCF has multiple functions including security and authentication of the SIP messages. It establishes some IPsecurity associations towards the IMS terminal which offer integrity protection. The proxy also authenticates the user during the registration process, hence the identity of the user does not need to be authenticated again. In addition, it also authenticates the SIP messages that allows to discard messages that are not build following SIP rules.

Interrogating- CSCF (I-CSCF)

This is another SIP proxy server that is mainly located in the home network, but in some special cases may also be located in the visiting network. The I-CSCF has interfaces with the SLF using the Dx interface and the HSS using Cx interface of the Diameter protocol. It retrieves information from these entities to correctly forward incoming messages from the P-CSCF to a destination, typical an S-CSCF [CGM09]. It is common to have multiple I-CSCF in the network serving various services, such as partial encryption of SIP messages containing sensitive information about domains, their capacity etc.

Serving- CSCF (S-CSCF)

[CGM09] This is the central node of the signalling plane which is a SIP server that also acts as a SIP registrar. There are multiple S-CSCFs in the IMS network serving a number of IMS terminals, depending on the capacity of each node. The I-CSCF is responsible for assigning an S-CSCF to the IMS terminal to serve. One of the many functions of the S-CSCF is to maintain an interface with the HSS (like the I-CSCF) using the Cx interface and with the SLF using the Dx interface of the Diameter protocol to achieve the following:

- Downloading authentication vectors from the HSS to authenticate an user.
- Fetching and storing user or service profiles from the HSS that include the service profiles.
- Communicating with the HSS to inform that this is the assigned S-CSCF for the duration of registration.

Another important function of the S-CSCF is providing SIP routing services, i.e., if a phone number is dialed instead of a SIP URI, the S-CSCF will do the necessary translations.

Any originating or terminating SIP message traverses through the assigned S-CSCF. The S-CSCF accesses the SIP message to determine the number of ASes that need to be triggered to provide the requested services on its way to the final destination.

Media Resource Function (MRF)

[3GP13] The MRF is a source for the network to initiate media resources in the home network. It is used for playing media announcements, real-time transcoding of multimedia data etc. The MRF is further divided into the Media Resource Function Controller (MRFC) which is a signalling plane node that acts as a SIP User Agent to the S-CSCF and Media Resource Function Processor (MRFP) which is a media plane node that implements all media-related functions [Acc07].

Break Out Gateway Control Function (BGCF)

[3GP13] This IMS element selects the network in which PSTN breakout has to occur. It is used for calls from the IMS to a phone in a Circuit Switched network, such as the PSTN or the PLMN; it forwards the signaling to the selected PSTN/PLMN network. If the breakout occurs in the same network as the BGCF then the BGCF selects a MGCF (Media Gateway Control Function) that will be responsible for inter-working with the PSTN, and forwards the signaling to MGCF. Otherwise it forwards signaling to BCGF of another operator network. The MGCF then receives the SIP signalling from the BGCF and manages the interworking with the PSTN network

Public Switched Telephony Network (PSTN) Gateways

These gateways are used for inter networking with the Circuit Switched network. These gateways include Signalling Gateway (SGW), Media Gateway Controller Function (MGCF) and Media Gateway (MGW) [Acc07]. This thesis is not focused on PSTN, hence these gateways are not further explained.

2.3.3 Service Layer

The service layer consists of application servers (AS) which are SIP entities, hosting and executing services. ASes provide services to the end users. Services can be of various type, for example, video conferencing, messaging, presence etc. Depending on its implementation, one AS can host one or many services [KG08]. The ASes interface with the S-CSCF using SIP signalling and the HSS using Diameter. This interface allows the S-CSCF to get the name and address of more than one AS and the order in which each AS should be contacted. A SIP addressing scheme to the AS is known as SIP URI. The AS decides which services should be deployed in any particular session using filter rules provided by the HSS. If any extra information is required for the execution, the AS is able to communicate with the HSS to learn about the service profile of the subscriber[KG08]. There are three types of ASes [CGM09], namely SIP AS, OSA-SCS (Open Service Access–Service Capability Server) and IM-SSF (IP Multimedia Service Switching Function). Multiple ASes can be deployed in the same domain; they can also be of different types. ASes other than SIP AS is outside the scope of this thesis.

SIP AS hosts and executes IMS services based on SIP. They can act as redirect servers, proxy servers, originating/ terminating user agents (UA) as well as Back-2-Back UA [KG08]. An IMS service execution is a sequence of SIP procedures, such as INVITE requests and responses.

2.4 Service Profiles

Service profiles or user profiles are SIP routing information stored in the network database, HSS. Routing information is associated with Public User Identities of users and the Public Service Identities for services and service related resources [Gou07]. Service profiles are transferred to the S-CSCF from the HSS over the Diameter interface, Cx. Service profiles are composed of user identities, name of the S-CSCF allocated to the user, registration and roaming profile, authentication, control and service information[Acc07]. Service information includes a list of initial filter criteria (iFC) [3GP12a] which are processed in a chronological order by the S-CSCF to serve the services the user is subscribed to.

Initial Filter Criteria (iFC)

Often users are subscribed to multiple services which can be hosted on the same or different ASes. These services are composed together using a service chaining feature known as the iFC. IFC allows a set of services to be invoked in defined circumstances. IFCs are at the core of service compositions and execution. The following representation in Figure 2.2 shows how service chaining works when services are hosted on different ASes.

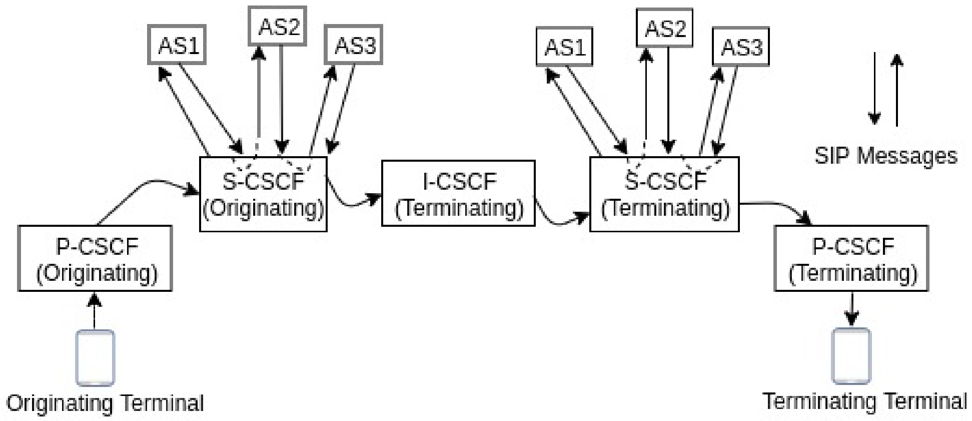


Figure 2.2: Working representation of iFC

Initial Filter Criteria is composed of the following elements, according to [Gou07], [Ber16]:

Priority: The priority of each iFC is set by an integer where 0 is the highest priority.

Trigger Point: A set of criteria that needs to be fulfilled by any SIP request to be routed to an AS, often known as Service Point Triggers (SPT). These are linked through logical operators like AND, OR and NOT. Trigger point conditions can be related to:

– Request-URI

– SIP Method

– SIP Header

– Session Case

– SDP (Session Description protocol)

Hence, Trigger Point is a set of conditions and Application Server is the action. **Default Handling:** There might be times when the AS does not respond to a request, the S-CSCF follows the directions from default handling to find out the next move. The iFC may only address a single AS but, an end-user is very likely to be subscribed to multiple services that are served by various ASes. Each iFC might not address a complete individual service in an AS as multiple services can be host in a single AS through service chaining.

2.5 Limitations of iFC

Even though iFC is a powerful mechanism to chain services, many service orchestrations face challenges as 1) message contents cannot be modified [KT07] e.g. SIP headers, neither before nor after invocation, 2) results from the invocations of several services cannot be 'merged', 3) limited expressiveness of conditional invocations and 4) SIP messages are unable to use any other information like network information to forward a SIP message [KRA06]. To overcome these limitation, the telcos are using some approaches that allow them to combine services creatively. Existing practical solutions from the industry and 3GPP proposed solutions are mentioned in the following chapter.

Chapter 3
Service Interaction in IMS

3.1 Introduction

The ability of IMS to provide multimedia services is one of its most important strengths. It is common to encounter multiple services instead of single services that are subscribed by the users. Each application server can host one or more services. Multiple services are chained using iFC which is known as service composition. Service compositions work well centering around particular composition choices as well as service deployment rules. IFC also plays a significant role in successful service compositions. But as it was seen in the previous chapter, iFCs do come with their own limitations. Often times the telcos come across services that are not possible to combine together. Services that work correctly when deployed alone but create conflicts when deployed together with other services is known as service interaction or feature interaction. Feature interaction is an active issue and many solution proposals have been put forward till now to minimize conflicts. This chapter gives an insight on a feature interaction with two basic services that the telcos provide, how the issue is being handled in the existing systems and the necessity of a new and better solution is discussed.

3.2 Feature Interaction Scenario

Feature interactions can be of two kinds: static or dynamic [3GP09b]. Static interactions occurs when the application invocation order is fixed in each communication session, whereas dynamic interactions are due to the application invocation order that is changed dynamically in different communication sessions based on the dynamic information other than normal Service Point Trigger (SPT); SPT is each of the conditions in the iFC list. A static feature interaction scenario is described as follows:

Call Forwarding Unconditional (CFU) [21] and Call Barring (CB) [20] are two of the most common services that are provided by all telcos. The feature interaction scenario chosen for this thesis is a combination of these services. The scenarios regarding these services are described as follows:

Alice, a subscriber, is subscribed to a the CB service, which allows the caller to block or screen particular or all incoming or outgoing calls to and from particular users. The subscriber has freedom over the type of CB they want to subscribe. In this case, Alice has only blocked a user called Charlie such that all calls to Charlie from Alice will be screened. However, this means that user Charlie is able to make calls to Alice.
Another user, Bob is subscribed to the service CFU. This service allows Bob to forward any calls made to him to be forwarded to another user of his choice. In this scenario, he has chosen the user Charlie where all of Bob's calls get forwarded.
The two services are deployed in two separate application servers in the IMS which may belong to the same or different vendors depending on the choice of the telcos.

The possible outcomes for this scenario are described as follows:

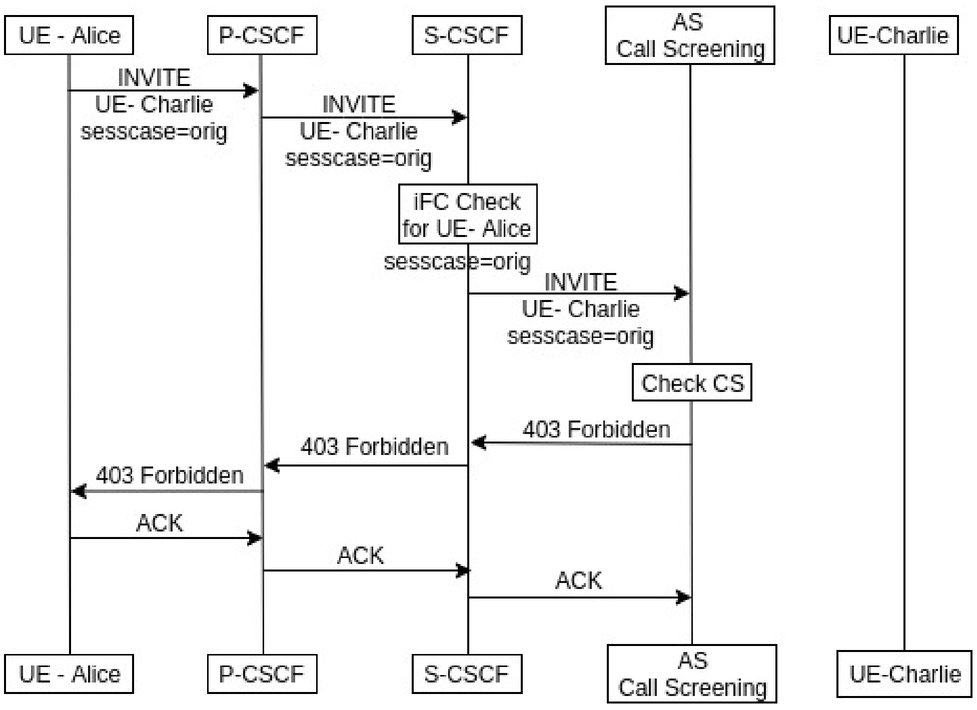


Figure 3.1: Typical Call Barring Scenario

Scenario 1: A call is being made from Alice's User Equipment (UE) to Charlie (for some reason). The INVITE reaching the CB service evaluates the Request URI and find the call is to Charlie, thus sends a response 403 Forbidden [3GP17], which is the desired result from the service. As Alice is the originator of the call, this service is an originating service, referred in the session value as 'orig'. Figure 3.1 represents the behavior of the service.

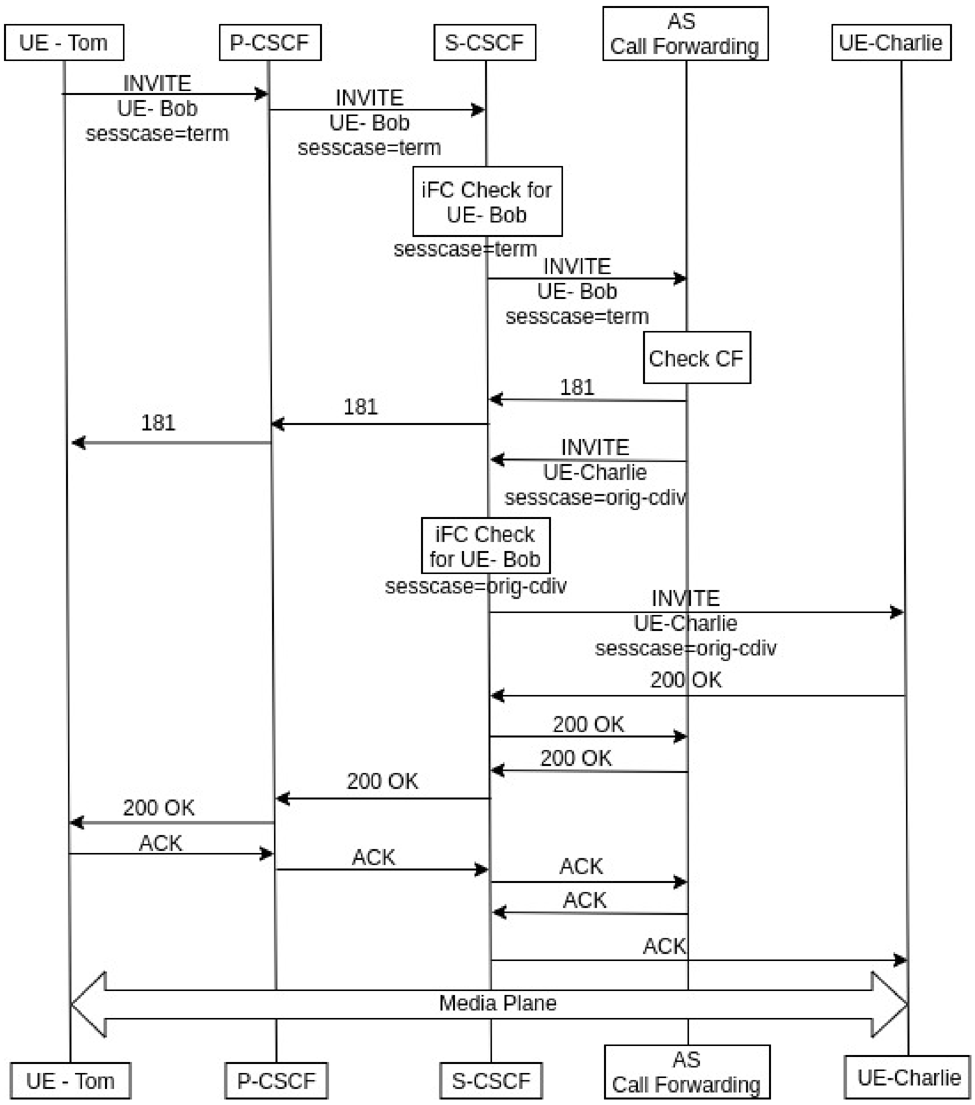


Figure 3.2: Typical Call Forwarding (Unconditional) Scenario

Scenario 2: Bob has chosen all his calls to be forwarded to Charlie. As a call made to Bob will be forwarded, Bob is the terminating party of the call making the service a terminating service, referred in the session value as 'term'. The CFU service adds a 'Diversion' in the header of the INVITE request which identifies the party that is doing the call transfer. The service also checks if the user is allowed to make this call transfer. The session case is also changed to 'orig-cdiv' [3GP10] when the 'Diversion' is added. CDIV means Communication Diversion. This makes the S-CSCF treat this INVITE as an originating INVITE and looks for originating services of the user, if any. A typical behavior for this service is provided in the Figure 3.2.

Scenario 3: When a call is made between the user agents Alice and Bob, the services interact with each other leading to a feature interaction. Anybody trying to call Bob gets forwarded to Charlie - this also includes Alice. Now, this is the correct outcome for the CFU service and even though the CB service is working properly, it has failed to deliver the desired result for Alice because this user does not want to be connected to Charlie. If the services were to perform correctly together, Alice calling Bob should be barred since Alice has barred outgoing calls to Charlie.
It is necessary to look into the details of this scenario. The exchange of SIP messages for scenario 3 is represented in Figure 3.3. In the diagram, it is assumed that there is only one S-CSCF in the network, thus all the users - Alice, Bob and Charlie are registered in the same registrar. Hence, the S-CSCF is aware of user locations for all three and does not need to query the I-CSCF for the locations. After the first iFC check for the originating part, Alice, the INVITE of the call reaches CB service where the Request URIs of Alice and Bob are compared resulting in no barring of Bob, hence the call can be made between them. The INVITE now contains the session value 'term' meaning that the S-CSCF will check for any terminating services for Bob, the terminating party. CFU is activated for Bob which lets the INVITE reach CFU service. This service evaluates to find that Bob has his calls forwarded to Charlie. The service changes to Request URI to Charlie, adds Diversion with Bob's address and session value of 'orig-cdiv'. The INVITE is sent back to the S-CSCF again and now due to the 'orig-cdiv' session value, the originating services of Bob, the value in the Diversion header, is checked including if Bob is allowed to make this call to Charlie. There is no originating service for Bob in this scenario and Bob is allowed to transfer this call. Hence, the call is being connected between Alice and Charlie.

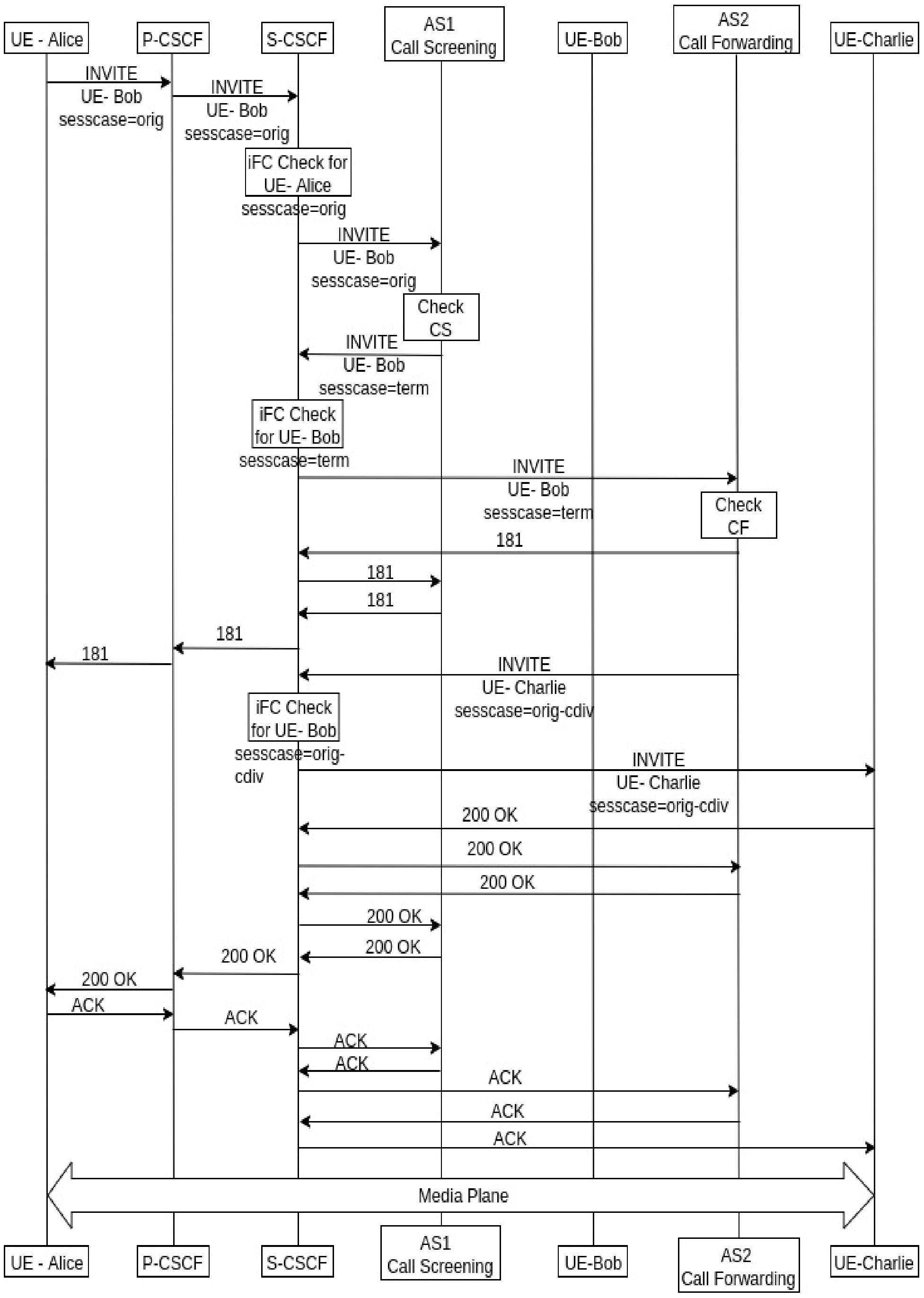

UE - Alice
P-CSCF
S-CSCF
AS1 Call Screening
UE-Bob
AS2 Call Forwarding
UE-Charlie
INVITE UE- Bob sesscase=orig
INVITE UE- Bob sesscase=orig
iFC Check for UE- Alice sesscase=orig
INVITE UE- Bob sesscase=orig
Check CS
INVITE UE- Bob sesscase=term
iFC Check for UE- Bob sesscase=term
INVITE UE- Bob sesscase=term
Check CF
181
181
181
181
181
INVITE UE- Charlie sesscase=orig-cdiv
iFC Check for UE- Bob sesscase=orig-cdiv
INVITE UE- Charlie sesscase=orig-cdiv
200 OK
200 OK
200 OK
200 OK
200 OK
200 OK
200 OK
ACK
ACK
ACK
ACK
ACK
ACK
ACK
Media Plane

Figure 3.3: Feature Interaction Scenario

3.3 Interaction Management in Existing System

The scenario presented in the previous section is an example of one of the many feature interactions in the telco industry. There is a formal model that describes how services are to be delivered in IMS, but in many cases like the mentioned scenario, the model does not seem to function properly. According to the standard, the ASes only communicate with the S-CSCF that handles all required operations; this includes responding to error conditions. As an attempt to support various kinds of service compositions and decrease feature interactions, telcos have come up with different solutions with slight enhancements in the service layer. There are three types of approaches that are actively used in the telco industry to achieve the desired service outcomes which are mentioned in the following subsections.

Monolithic Approach

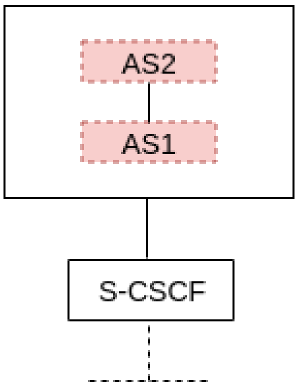


Figure 3.4: Monolithic Approach

This approach is also commonly used in the Intelligent Network (IN) architecture. Here composite services are made in a monolithic way and the service composition(s) are built for one particular telco operator by one vendor. These form large services that perform service chaining inside one big box and the S-CSCF sees it to be a single AS. For this reason, feature interaction issues can be solved with logic placed inside the big box acting as one large service. This solution is rather constraining since the service additions, extensions or modifications can only be done by the same vendor. It is often difficult to introduce services from other vendors. This leads to limitation of services as it is expensive and inflexible for the telecos. Figure 3.4 illustrates this approach. This approach uses the standard iFC to deliver composite services.

The feature interaction scenario described in the previous section is handled in this approach such that both the services will be deployed inside one big box in one or two ASes. The iFC will be in the S-CSCF and the interaction logic will be placed in the big box such that the correct outcome of the services is provided to the subscriber.

The Man-in-the-Middle Approach

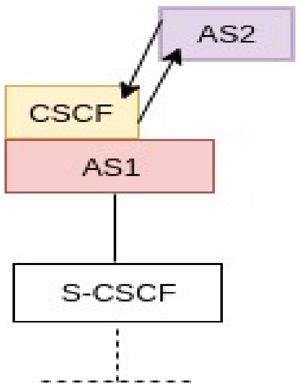


Figure 3.5: Man-in-the-Middle Approach

Another popular approach to the issue is “man-in-the-middle”. In this approach, one vendor makes a standalone service along with an interface that enables the service to be used with another service, potentially from a different vendor. In this situation, the first service acts like the man-in-the-middle. Figure 3.5represents this approach. Different colored AS boxes represent services from separate vendors and the ’CSCF block’ is the layer that enables the integration. This method is comparatively cheaper and more flexible than the previous approach. But this approach has limitations as it is unclear which vendor is responsible for modifying the services to implement a new feature or a change in one service may also require some changes in another service.

With this approach, the CB and CFU services are deployed in separate ASes. The CB service will have the ’CSCF block’ which will call the CFU service. In such a case, the S-CSCF will get the combined response from both the ASes and the call will be forward or barred correctly.

Service Broker/SCIM Approach

This is the solution proposed by 3GPP which describes a two-tier architecture introducing Service Capability Interaction Manager (SCIM) [3GP09a]. SCIM acts as a service broker in the architecture. The architecture diagram in figure 3.6 illustrates the proposal. SCIM as a standalone entity is to be located between the S-CSCF and ASes for managing interactions among the application servers to control conflicts and interaction between services [GCB06]. Even though the entity was proposed, there were no explicit specifications mentioned about how the system would handle feature

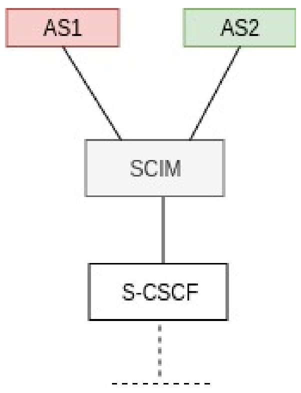


Figure 3.6: Service Broker/ SCIM Approach

interactions and incompatibilities [GCB06]. This gave the vendors an opportunity to come up with creative solutions. According to the proposal, services do not need to know about other services since all coordination of services is managed by the SCIM. There is no interaction between the S-CSCF and ASes unless through the SCIM. This method, however, is not integrated widely by the operators due to its lack of standardization and expenses. This solution is the target of this thesis.

3.4 3GPP Standardization

The solution proposed by 3GPP is a two-tier architecture introducing Service Capability Interaction Manager (SCIM). 3GPP included this entity in the Network Architecture technical specification assuming that it will be involved in the architecture to facilitate feature interaction solutions. The architecture diagram in Figure 3.7 showcases the proposal. SCIM was envisioned to be an optional component of a SIP AS. This entity is to act between the S-CSCF and various AS over the ISC (IMS Service Control) reference point. SCIM is expected to do coordinated execution of potentially conflicting services. The original purpose of SCIM was the coordination of service interactions, however, its compositional capacity has increased flexibility of the system [GC08].

It has been over a decade since 3GPP defined SCIM in 2002 but no particular specifications for the entity was provided. In the technical specification TS 23.002 ([3GP09a]) there is a mention that the functional architecture of the SCIM is outside scope of the standards. Eventually in 2007-2008, a technical report TR 32.810 [3GP07] was released with multiple suggestions on the entity but no standards were specified and the report item was considered to be complete with no further works in the discussion. This thesis will take into account much of these suggestions and other research studies on the topic to propose a solution to the feature interactions.

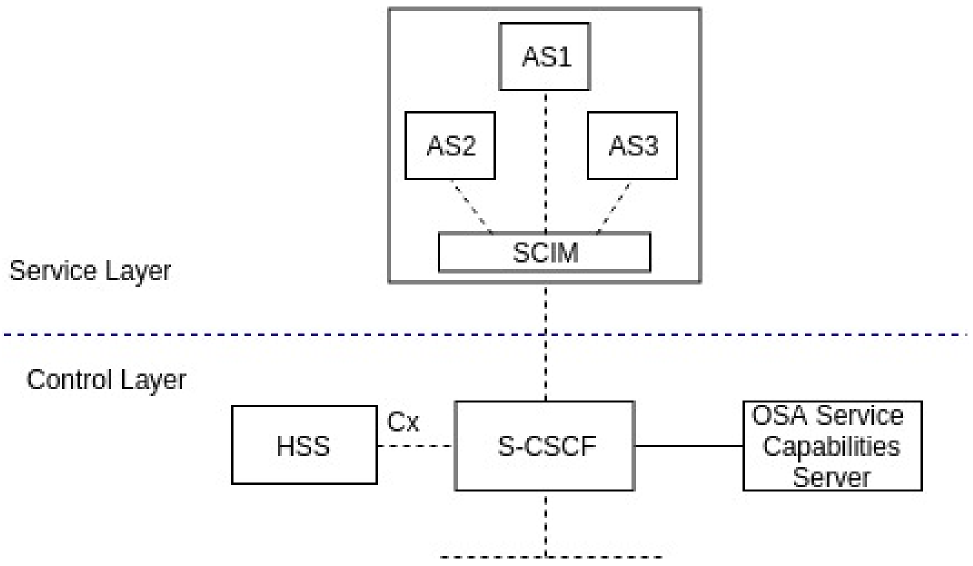


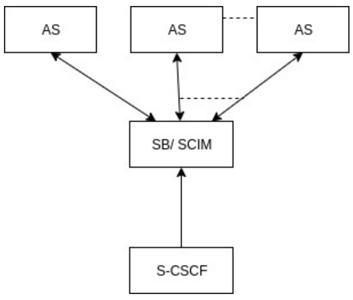
Figure 3.7: Two-tier architecture with SCIM
[3GP09a]

For the role of interaction management, the 3GPP has also introduced a 'Service Broker' (SB) as part of the OSA framework that facilitates third party access to services and network features in a managed and controlled manner [3GP09c]. Similar to the SCIM, specifications has not been defined for the service broker as well. In multiple research papers on the topic and 3GPP technical reports, the terms SCIM and service broker has been used interchangeably. Hence, this thesis will also use both the terms to make appropriate references.

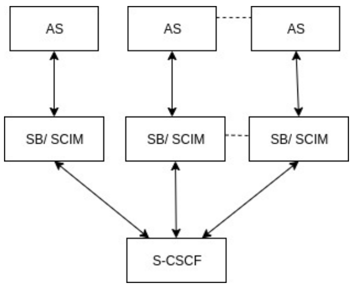
3.5 3GPP Technical Report: TR 23.810

After the SCIM was defined, there were no subsequent publications related to the specifications of the entity. However, a work item was initiated by TSG Service and System Aspects (TSG-SA) [3GP19], to investigate the impacts of inclusion of a SCIM entity in the existing network. The study was concluded with a few architectural suggestions, interaction logic, architecture reference model. This section describes the architectural suggestions only.

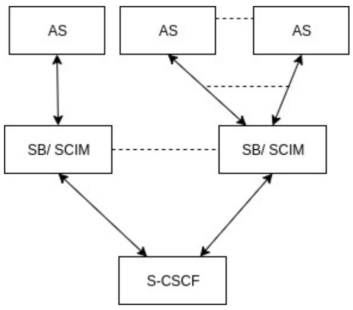
This report aims towards the successful functioning of the service brokers that are to be introduced in the network. It mentions the two categories of service brokering functions: off-line and on-line. Off-line service brokering is out of the scope of this report. On-line brokering functions are aimed to resolve both static and dynamic



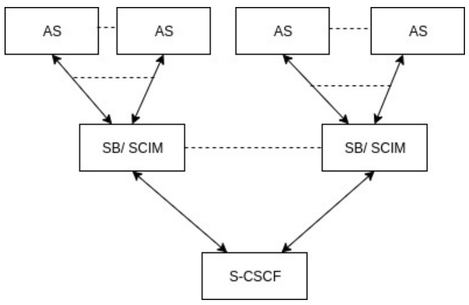
(a) Centralized SB



(b) Distributed SB



(c) Hybrid SB(1)



(d) Hybrid SB(2)

Figure 3.8: Architecture Alternatives for Service Broker [3GP09b]

feature interactions. The architectural requirements for on-line service brokering includes, but not limited to, the following:

- The impacts of introducing the service brokering function to IMS core network and AS should be minimized.
- The service brokering architecture should be flexible enough to process the potential interaction requirements for new applications.
- The service broker shall efficiently interact with the AS and avoid unnecessary interaction.
- The service broker should support service integration across network hosted applications where the applications can reside either in the same AS or in different ASes.
- Allow service integration between SIP and non-SIP applications available via the IMS service architecture.

The report further mentions the following three architecture alternatives that can be possible to incorporate a SCIM/ SB in the network represented in figure 3.8. The alternatives are:

Centralized Service Broker

The SB/SCIM entity is invisible to the ASes involved and the S-CSCF views the SB as an AS supporting the ISC (IMS Service Control) interface. The SB Functions can be located outside S-CSCF, or embedded in S-SCCF. This is represented in Figure 3.8a.

Distributed Service Broker

Each AS involved is connected to one SB. To allow coordination of services the SBs can be located independently or embedded in the AS. To the S-CSCF the SB and AS appears to be one entity supporting the ISC interface. The S-CSCF relays the messages among the SB until all AS finish their functions. This architecture is represented in Figure 3.8b.

Hybrid Service Broker

The SBs have to manage service interactions among the ASes under its direct control as well as with its peer SBs. The two architectures as shown in the diagram are just examples of the hybrid architecture. Architecture in Figure 3.8c shows that one of the SBs may act as both centralized and distributed SBs. Architecture in Figure 3.8d illustrates multiple SBs interfaced with the S-CSCF and they act as both centralized and distributed brokers.

3.6 Existing SCIM Products

Even though in the technical specification report TS 23.002 [3GP09a] there is a mere mention of the SCIM and no defined specification for it, there are multiple SCIM products that have been developed by various groups invested in the telco industry. These products vary vastly from each other as there are no minimum requirements that need to be filled. To demonstrate that, this thesis looks into existing SCIM products. Of the available SCIMs, two of these products are reviewed and the descriptions are stated below:

3.6.1 Lucent Service Broker™

[KRA06] The core of the broker, Lucent Service Broker engine, can dynamically load Java* code fragments, called 'steplets'. Steplets work with the engine to handle SIP messages that it gets from the S-CSCF. By 'handling' it means examining incoming

SIP messages, forwarding requests to application servers, modifying messages before forwarding (if necessary) or reply to requests instead of forwarding them. Since, steplets are Java code, existing Java libraries to read/ write files, connecting to databases/ web servers etc. can be done. The service broker engine is represented in the following Figure 3.9 from the Lucent Service Broker™ paper in [KRA06].

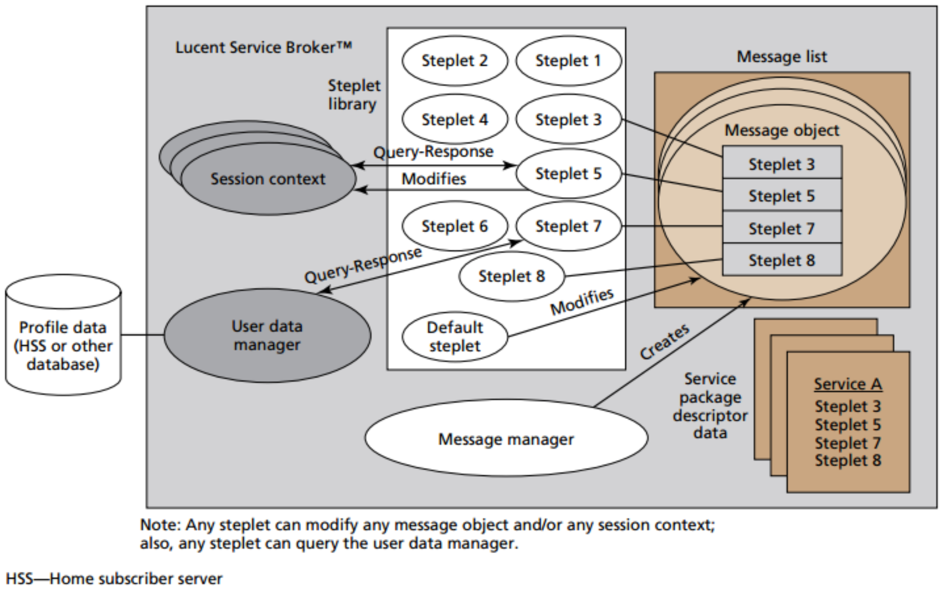


Figure 3.9: Lucent Service Broker™ Engine
[KRA06]

The Broker is message-centric. Once it receives a SIP message, it creates a message object and adds it a message list of pending messages. The broker processes the message by invoking steplets on it. The first invoked steplet is the *default steplet* which determines a list of steplets that need to be invoked for each message dynamically, i.e. any steplet can add steplets to the list at anytime, to complete the request. The steplets processes the message, determines policy issues (e.g., which AS to forward to) by examining the iFC and handles the required service logic. An incoming SIP message is usually handled by a succession of steplets. The default steplet uses information from the HSS or the subscriber database to invoke the rest of the steplets. All the steplets use the same message and session attributes to communicate between each other and keep track of the messages.

Benefits

Steplets can access message contents and modify them if needed based on the response from the ASes. This allows service blending easier and should allow resolution of feature interaction for many complex services. As mentioned earlier, the list of steplets is dynamic, which makes it very flexible to handle messages from many users. The steplets can also access additional information from the providers user database or any other database.

3.6.2 Rhino Service Interaction SLEE(SIS)

[Met11b] SLEE is short for Service Logic Interaction Environment. SIS is OpenCloud's script-driven, JAIN SLEE compliant, multi-protocol, declarative service interaction engine that lets developers script service interaction logic among any ASes, networks or protocols. This SCIM can support both local and external services as well as combine them to make complex services. With the SIS, developers can manage interaction by isolating and controlling how services interact.

Figure 3.10 represents the internal structure of the Rhino SIS (taken from the official Rhino SIS web-page). This illustrates that the SIS is composed of scripting, service-interaction engine, services and management. *Scripts* control the services

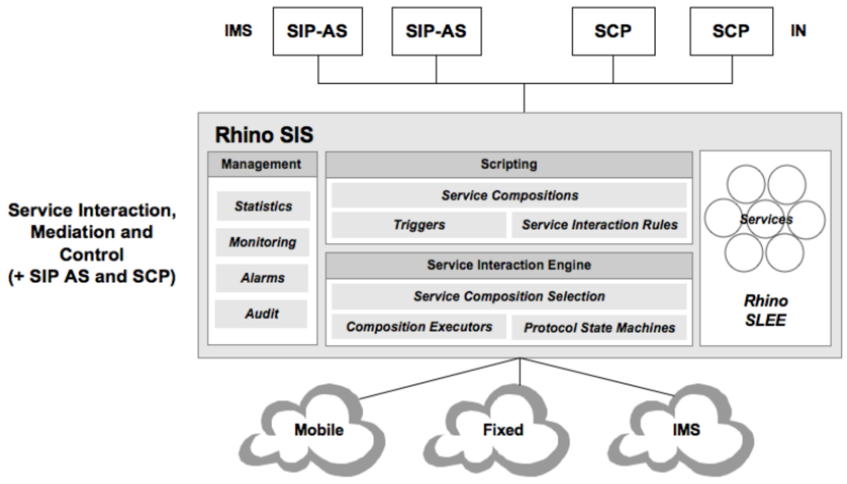


Figure 3.10: High Level SIS Architecture
[Met11b]

that the SIS invokes for each call. A script syntax is composed of services available and sequence or priority of services that should be invoked and triggered. Trigger and composition scripts allow various kinds of service composition. Even though the service interaction rules script contains some predefined service interaction rules, these can be easily influenced by the composition scripts. SIS also has interceptors which is a set of script elements that may modify the parameters of the messages it intercepts (or may perform some other action). Interceptor scripts are typically embedded directly within a composition but can also installed as a SIS component.

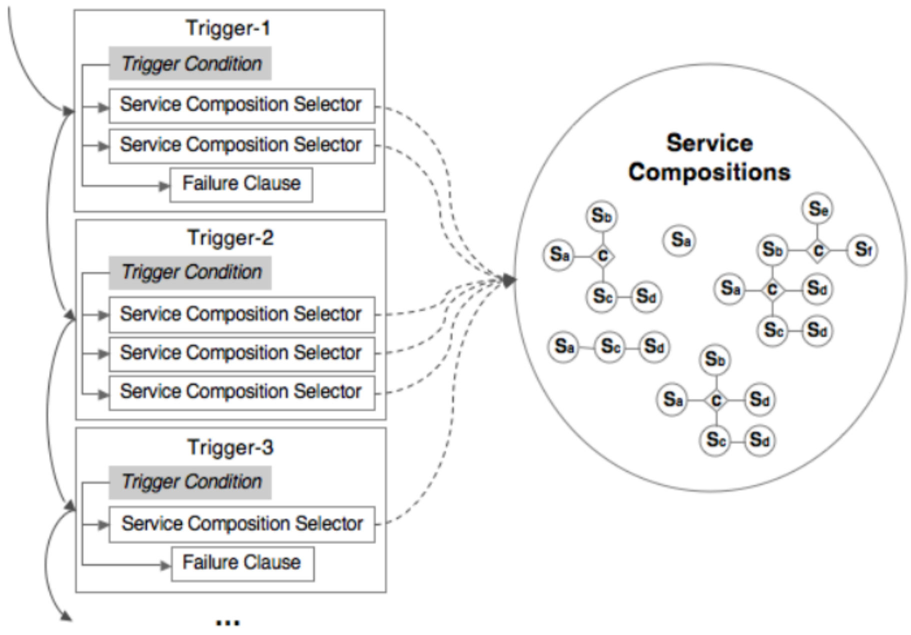


Figure 3.11: Managing Service Interaction
[Met11b]

The *service-interaction engine* evaluates the scripts related to the trigger, service compositions, invokes services at appropriate time and evaluates outcomes from the invocations. When invoking services in a composition, the request is passed to each service according to priority and the outcome is considered. The outcomes are protocol driven. Depending on the outcome, next services are invoked until all necessary services have been invoked and sends a response to the original request. Figure 3.11 demonstrates how triggers and service compositions work. Triggers are evaluated according to priority and moves along the list of triggers evaluating each of them unless on evaluates to be true and a composition of the corresponding trigger is selected.

Services can be of two types: local (implemented in the Rhino platform) or external (from other SIP ASes). These can be triggered by the engine or combined together

to achieve complex services. During the time of invocation of complex services, the engine isolates the services from each other and ensures each service gets the request in the correct order.
Management helps monitoring the entire SIS including triggers, outcomes of invocations as well as predefined alarms for network information.
The Rhino SIS is also expressed as 'extensible' using Java API to extend service composition. This allows the SIS to access and modify signalling parameters.

Benefits

It is possible by the Rhino SIS to manipulate the outcomes of each invoked service which gives a lot of flexibility to combine services and manage feature interactions. Since both local and external services can be combined and invoked, it is possible to make complex services with ease. The SIS is not limited to SIP related services, rather can handle different protocols. The management tool makes the monitoring the system efficient with the predefined alarms.

3.7 Necessity for SCIM Standardization

With the increasing complexity in the nature of service compositions, a creative solution is necessary. For larger and complex networks, SCIM is a better choice for interaction management over the other two approaches that are in use now. However, due to the lack of any defined specifications, the SCIM products, however vastly creative, lacks perspective in most cases. This in return has made the telcos more constrained in their views about integrating a SCIM in their networks. The lack of standardization also entails that there is no base line of what kind of feature interaction is being addressed and how. The SCIM products described in the previous section exhibit the major differences in the products that is targeting to solve the same problem. Hence, SCIM products available now are entirely depended on the developers. The cost of the entity along with the integration cost in the network is also an important concern for the telcos. Thus it is difficult for the operators to evaluate any SCIM, let alone incorporate it in their network.

The following chapter illustrates the service interaction scenario described here as well as implements it in a test bed environment along with a SCIM to find out how the SCIM performs and if the interaction can be resolved.

Chapter 4

Experiments with Feature Interaction and SCIM

Out of the three existing approaches to solve feature interactions, two of them have been widely adopted by the telco operators. Chapter 3 discussed how these solutions have been integrated into the IMS network along with the limitations they bring. The third approach in 3.3 which is the Service Broker/ SCIM approach, however, has not gained so much popularity due to multiple reasons. But with the increase of complexity in service compositions and feature interactions, this entity, which declared as an 'optional node in the service architecture' by the 3GPP [3GP09a] can lead to a solution that is better suited for the complexity presented now. To illustrate this solution, this thesis experimented with one of the existing SCIM products in the market known as the Rhino SIS, described in section 3.6, as an attempt to solve the feature interaction scenario presented in section 3.2. This chapter describes the implementation and interaction management that is achieved using the SCIM.

4.1 Implementation Choice

The only SCIM that was available to be implemented for use in this thesis was the Rhino SIS, which narrows down the choice of SCIM. Rhino SIS is able to interact with multiple kinds of ASes, so the choice comes down to the selection of technology to make the ASes. Two of the popular technologies to choose from are JAIN SLEE and SIP Servlets.

Multiple research papers has been published on benefits of using JAIN SLEE and SIP Servlets for the development of ASes. In the paper [CLP08], the authors compare the two technologies on the basis of the degree of code re-use,concurrency control, support for multiple protocol and management support. After providing in depth discussion on these, they conclude that SIP Servlet is better to use in case of simple services whereas JAIN SLEE is more apt to be used for sophisticated and complex services. Another paper by Bessler et al. is seen to choose SLEE and mentions that the powerful internal event model and the added value that is derived from being able to integrate other protocols besides SIP have led them to incline toward JAIN SLEE [SB07]. In a blog post by Ivelin Ivanov, co-founder Telestax, Inc. [Iva05], mentions that even though JAIN SLEE is a powerful environment, to develop simple applications, it is easier to use SIP Servlet. Weighing these information, for this thesis, SIP Servlet is selected as the technology to develop the applications.

To mimic the IMS network, for this test-bed, OpenSIPS was chosen. OpenSIPS is an open source implementation of a SIP server, which is not only limited to a registrar/ proxy [Ope16]. Section 4.2 provides information on all the tools that were used for this test-bed implementation.

4.2 Implementation Tools

Implementation of the feature interaction in a local setting was done using three main tools which are described in the following subsections.

4.2.1 Rhino TAS

[Met11a] Rhino Telecom Application Server (TAS) is a real-time telecommunication signalling platform with focus on scalability, low latency, fault tolerance, and high availability. It provides native support for standard-based IP and SS7 protocols. Rhino TAS is fully compliant with JSLEE [SM08] and SIP Servlet standards [EP06]. To deploy the Call Forwarding application, which is a SIP servlet application, on Rhino TAS, a SIP servlet Resource Adaptor [Oped] is required; similar for the Call Screening application. The SIP servlet applications deployed on Rhino TAS can be managed using a sipservlet console via a JMX interface. The Rhino SDK allows to create the TAS environment. This environment was used in the work to deploy applications in the SIP Servlet entity of the TAS. The following Figure 4.1 represents the TAS environment. This diagram is from the Opencloud website.

4.2.2 Rhino SIS

[Met11b] This tool has been elaborately described in the second subsection of section 3.6. This tool was chosen to be implemented as this was the only SCIM product available for use during the work. Rhino SIS is not an open source product like any other SCIM products out there. The SIS was deployed in between the OpenSIPs (S-CSCF) and the application servers (Rhino SIP Servlets). The SIS contained the triggers and composition selection for each trigger.

4.2.3 OpenSIPS

[Ope16] Open SIP Server (OpenSIPS) is an open source implementation of a SIP server. It functions as the proxy/ router or the registrar/ S-CSCF in case of this

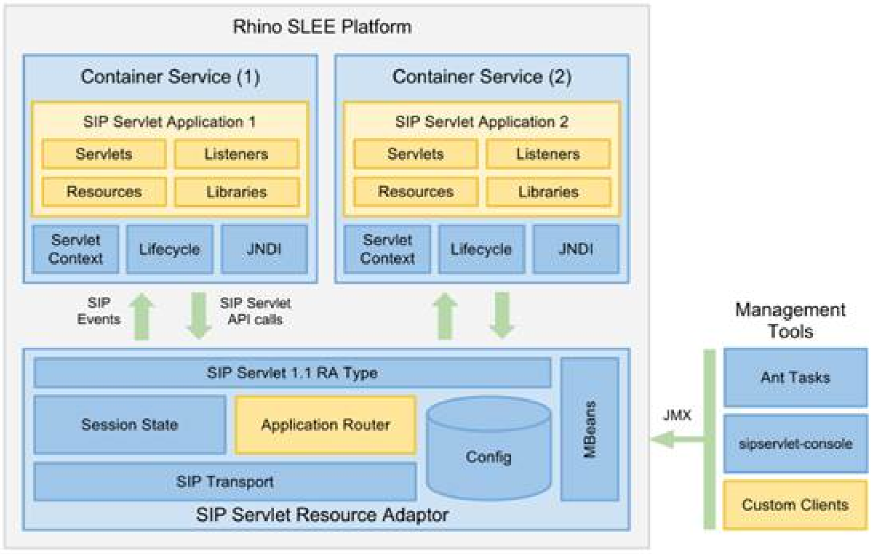


Figure 4.1: Rhino Telecom Application Server (TAS) [Met11a]

thesis. But it also contains application level functionalities that were not realized. This tool was chosen as it covered the functionalities of the P-CSCF as well as S-CSCF. As all the users were registered in the same domain sharing the same S-CSCF, no I-CSCF was needed in this case. This made OpenSIPS the ideal choice for this experiment.

To realize the complete implementation, some other tools that were essential were soft phones and a Java IDE platform to create the services. The next section describes the entire setup for the realization of the scenario.

4.3 Creation of Services

The two services that are created for this are the CB and CFU services using a Java platform. The services use SIP and Servlet resources to implement the desired actions. The interaction scenario of this thesis is limited to three users, hence for simplicity the services work with these three SIP URIs only, but with the introduction of databases to manage users, these services can be catered for a similar output. In the following subsections, the algorithms of the services are provided and the full service codes can be found in Appendix A.

4.3.1 Call Barring Service

This service allows the prevention of certain outgoing calls based on conditions. For this scenario, this condition is if the call is made to the user Charlie. There are three situations that would require this service to act differently. The algorithms for the tree scenarios are provided in Appendix A.3. The 'set' and 'add' commands are to add and set values to the Request header of the INVITE in order to manipulate the INVITE requests. The 'User Agent' field is used as triggers for the OpenSIPS (Appendix C).

4.3.2 Call Forwarding (Unconditional) Service

The CFU service is simpler than the CB service since it only has to either forward the call by changing the 'To' header field and setting the 'User Agent' or just changing the 'User Agent' and no forwarding. The algorithm is provided in Appendix A.1 and the full program is available in Appendix A.2.

4.4 Experimental Setup

Implementation of the test-bed had three main parts, namely, OpenSIPS, Rhino SIS and the Rhino TAS for two SIP Servlets. OpenSIPS and Rhino TAS was installed on a computer that was running Ubuntu 18.04.2 TLS operating system with Intel® Core™ i5-6200U CPU @ 2.30GHz × 4. The implementation on this scale needed to use the localhost address with different ports for all the entities. By trial and error, it was realized that deploying the two services in the same Rhino TAS was interfering with each other, hence, another Rhino TAS was installed in a Virtual Box with one of the services deployed.

To test the functionalities it was necessary to make calls between users, hence two different soft phones, namely Linphone [Lin19] and Zoiper [Zoi19], were used. Since the implementation was on a Linux setting, the choice options for the soft phones were limited as many of the open source soft phones were Windows OS based and did not function in a Linux system. Two different soft phones were used as neither of these phones allow multiple instances at the same time.

The services were written using IntelliJ as the service codes are Java-based. Prior experience in IntelliJ motivated this choice.
Figure 4.2 represented the deployment of the services and how the scenario is realized in the local environment. In this implementation, the SIS does not provide any functionality other than triggering the services in the order of priority that is defined in the composition scripts installed in the SIS.

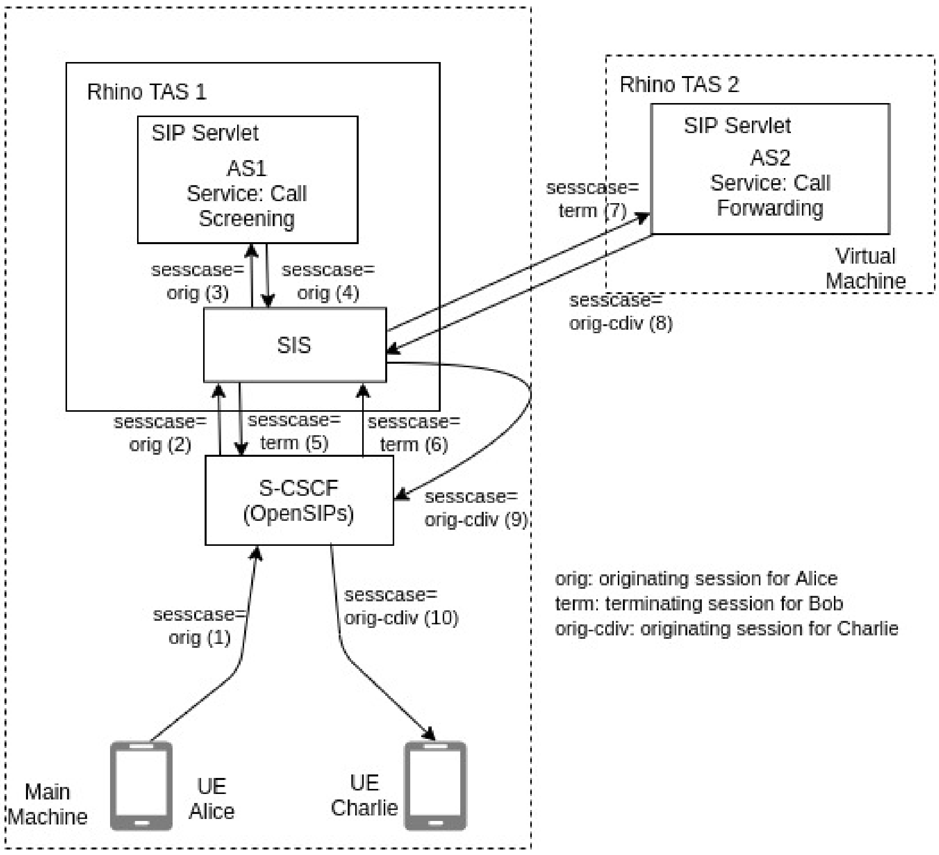


Figure 4.2: Deployment of Services with SIS and Rhino TAS

4.5 SIP Messages in the Experiment

Figure 4.3 illustrates the exchange of all the SIP messages that take place during a call made from User Alice to Bob. In the end, it is clear that the call goes through to Charlie even though Alice has barred Charlie from all of her outgoing calls.
In this scenario, there is only one S-CSCF all the users are registered to; OpenSIPS is acting as the registrar/ S-CSCF. This means that the S-CSCF knows the location of all the users and does not need to query the I-CSCF. In the diagram, a call is being made from the user Alice to Bob. Alice is the originating (orig) and Bob is the terminating (term) parties in this call. How the messages traverse and what they signify is discussed below:

When an INVITE request is received by the S-CSCF from Alice, the session header field is 'orig'; the S-CSCF checks Alice's user profile to check for iFCs. Alice is subscribed to Call Barring service and the user has barred outgoing calls to the user Charlie. The S-CSCF forwards the INVITE request to SIS that triggers the composition for AS1 hosting the CB service. Here, the SIS is used only to trigger the services and no manipulation of any sort is done, to mimic the scenario in the actual network. Next, the CB service checks the Request URI field in the header and finds that the call is to Bob, so the CB service does not take any action and forwards the request back to the S-CSCF for the call to be executed through the SIS.

Once the INVITE comes back to the S-CSCF, the session now is 'term' which implies that the S-CSCF will check the user profile of Bob for any iFC. Bob is a subscriber of Call Forwarding Unconditional service which gives him the ability to forward all his calls to another user, Charlie in this case. Hence, the INVITE is forwarded to AS2 through the SIS (similar to the previous CB service) which is hosting the CFU service. The CFU finds Charlie's user information to which the call should be forwarded. The CFU service changes the Request URI to Charlie's address, changes the session value to 'orig-cdiv' [Gro], adds a 'Diversion' field in the header and sends the INVITE back to the S-CSCF. Simultaneously, the CFU service also sends a response, 181 Call is being Forwarded, back to Alice. The response traverses back to Alice the same way it reached AS2, since record-route is inserted, no steps are skipped.

The S-CSCF one last time checks the iFC of Bob, due to the 'orig-cdiv' value in the header as this field marks that the originating call is from Bob. Bob is not subscribed to any originating services in this scenario. Hence, the INVITE is sent to the UE of Charlie and the session is established.

As described in the iFC subsection of section 2.4, iFC is only able to handle boolean replies from the ASes, i.e., the conditions in the iFC list can only be true or false, this is the nature of condition handling in the iFC. The iFCs are also unable to modify or use any information from the AS to take corrective actions to execute the proper result of the compositions. In the scenario, CFU service changes the Request URI to Charlie, but the CB service is unaware of it as the INVITE is not sent to the CB service again. Even if it was sent back to CB again, as the CFU service adds a 'Diversion' header field [Sys] that indicates that the call is being diverted from Bob, the service would have been compared between Bob, the address in the 'Diversion' header and Charlie, the terminating party now.

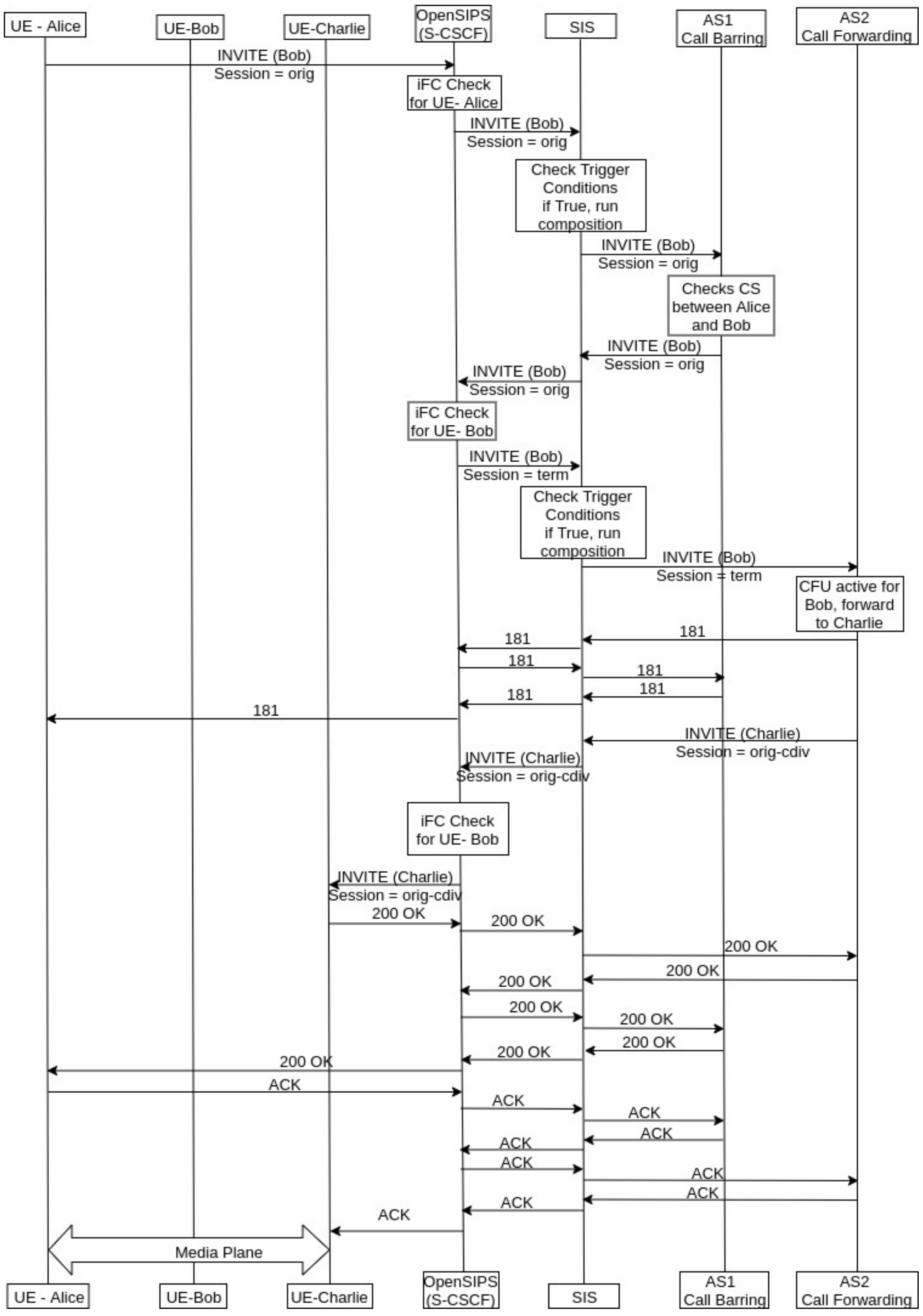

UE - Alice
UE-Bob
UE-Charlie
OpenSIPS (S-CSCF)
SIS
AS1 Call Barring
AS2 Call Forwarding
INVITE (Bob) Session = orig
iFC Check for UE- Alice
INVITE (Bob) Session = orig
Check Trigger Conditions if True, run composition
INVITE (Bob) Session = orig
Checks CS between Alice and Bob
INVITE (Bob) Session = orig
INVITE (Bob) Session = orig
iFC Check for UE- Bob
INVITE (Bob) Session = term
Check Trigger Conditions if True, run composition
INVITE (Bob) Session = term
CFU active for Bob, forward to Charlie
181
181
181
181
181
181
181
INVITE (Charlie) Session = orig-cdiv
INVITE (Charlie) Session = orig-cdiv
iFC Check for UE- Bob
INVITE (Charlie) Session = orig-cdiv
200 OK
200 OK
200 OK
200 OK
200 OK
200 OK
200 OK
200 OK
200 OK
200 OK
ACK
ACK
ACK
ACK
ACK
ACK
ACK
ACK
ACK
ACK
Media Plane
UE - Alice
UE-Bob
UE-Charlie
OpenSIPS (S-CSCF)
SIS
AS1 Call Barring
AS2 Call Forwarding

Figure 4.3: Implementation Scenario of Services

This is one of the many feature interaction issues that come up when multiple services that work accurately on their own (Section 3.2 illustrates this) but does not function well when placed together. Manipulation of the headers would be make this scenario work better with minimum effort. The next section demonstrates how this can be put into action.

4.6 Interaction Management with Rhino SIS

One of the strengths of the Rhino SIS is its script driven triggers and compositions which facilitates feature interaction solutions. In the previous section, the SIS was only used to trigger the services, but this section uses the interaction management aspect of the SIS by using script logic to re-write the request headers such that the INVITE request can be sent back to the CB service to check for any call barring that might occur. Figure 4.4 represents the flow of messages in the network when a call is made from Alice to Bob. The setup of the scenarios is same as in section 4.4.

As described in section 3.6, the Rhino SIS allows the developer to write XML scripts for triggers and compositions. To tackle the scenario discussed, two trigger scripts and two corresponding compositions are scripted and installed in the SIS; Appendix C includes the scripts for them. The first trigger, Trigger1.xml (Appendix B.1) has the conditions of ’method’ = ’INVITE’ and ’Session’ = ’orig’. When both the conditions are evaluated to be ’true’ the composition ’S-Trigger Handling’ (Appendix B.3) is executed. This composition only has the service CB for originating calls. The service code (Appendix A.4) is executed and one of the three situations of CB service can be true. First INVITE reaching AS1 evaluates algorithm A.4 true. With Session = term and User Agent = ’someStringValue’,the INVITE reaches OpenSIPS where this information is utilized to send the INVITE back to the SIS (Appendix C.1). Now, the second trigger, Trigger2.xml (Appendix B.2) evaluates ’true’ and the conditions ’method’ = ’INVITE’ and ’Session’ = ’term’ invoke the composition ’FS-Trigger Handling’ (Appendix B.4) which has CFU and CB as a composition one after the other. CFU algorithm A.1 is carried out with changes the Request URI and Session as well as addition of the Diversion header field. The INVITE is next sent to the SIS for the invocation of the service following the CFU in the composition, which is the CB service. Before invoking, the composition has some interceptors in place that allows header manipulation (Appendix B.4). The interceptor first stores the value of the Diversion header in a variable, next removes it and then invokes the CB service. This lets the service compare between Alice and the new Request URI, Charlie for the matter, according to the algorithm A.2, which in turn bars the call. The *SC_FORBIDDEN* SIP response is sent back to Alice, following all the routes of the route-reader and Alice sees a 403 Forbidden on the equipment screen.

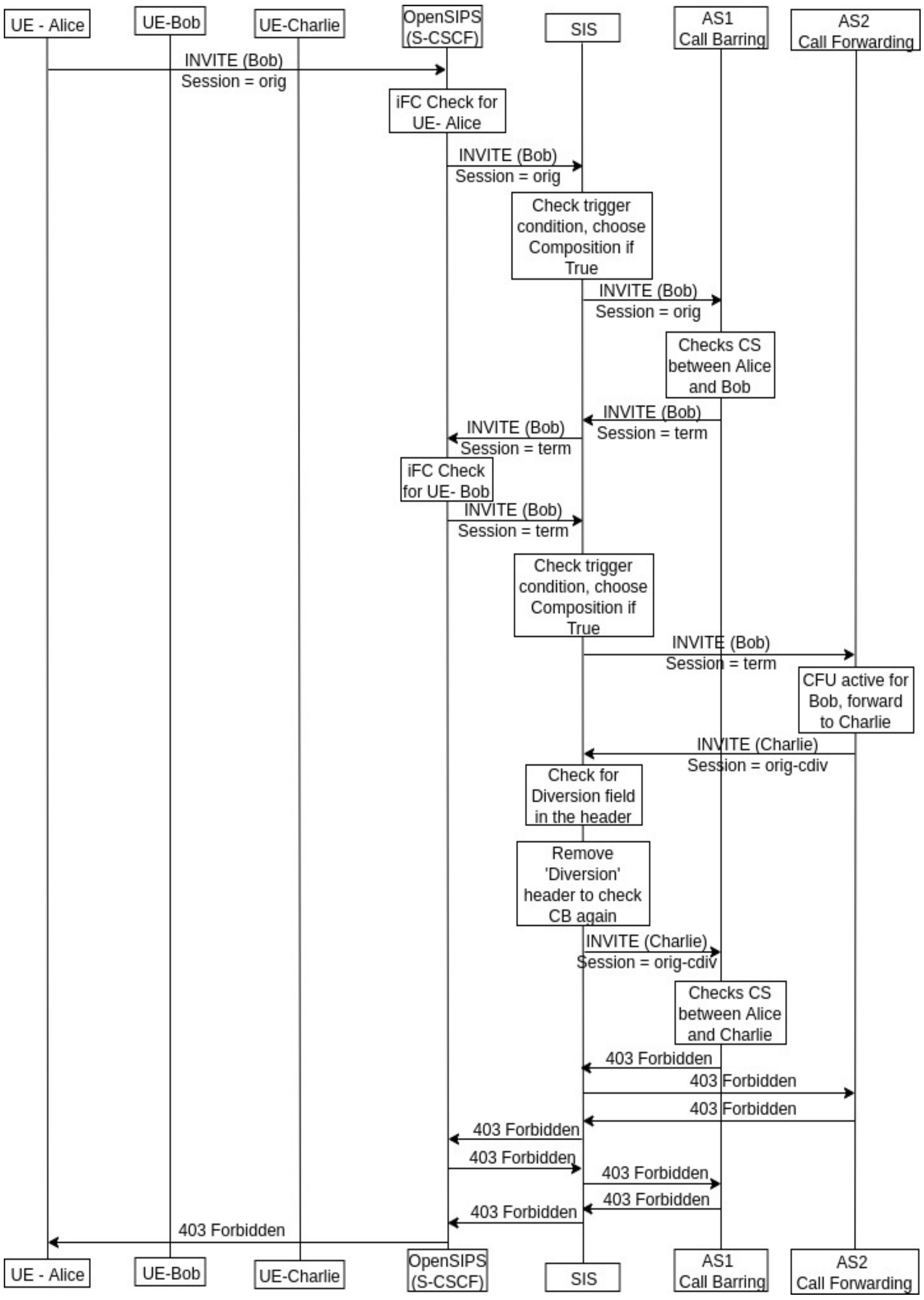

UE - Alice
UE-Bob
UE-Charlie
OpenSIPS (S-CSCF)
SIS
AS1 Call Barring
AS2 Call Forwarding
INVITE (Bob) Session = orig
iFC Check for UE- Alice
INVITE (Bob) Session = orig
Check trigger condition, choose Composition if True
INVITE (Bob) Session = orig
Checks CS between Alice and Bob
INVITE (Bob) Session = term
INVITE (Bob) Session = term
iFC Check for UE- Bob
INVITE (Bob) Session = term
Check trigger condition, choose Composition if True
INVITE (Bob) Session = term
CFU active for Bob, forward to Charlie
INVITE (Charlie) Session = orig-cdiv
Check for Diversion field in the header
Remove 'Diversion' header to check CB again
INVITE (Charlie) Session = orig-cdiv
Checks CS between Alice and Charlie
403 Forbidden
403 Forbidden
403 Forbidden
403 Forbidden
403 Forbidden
403 Forbidden
403 Forbidden
403 Forbidden
403 Forbidden

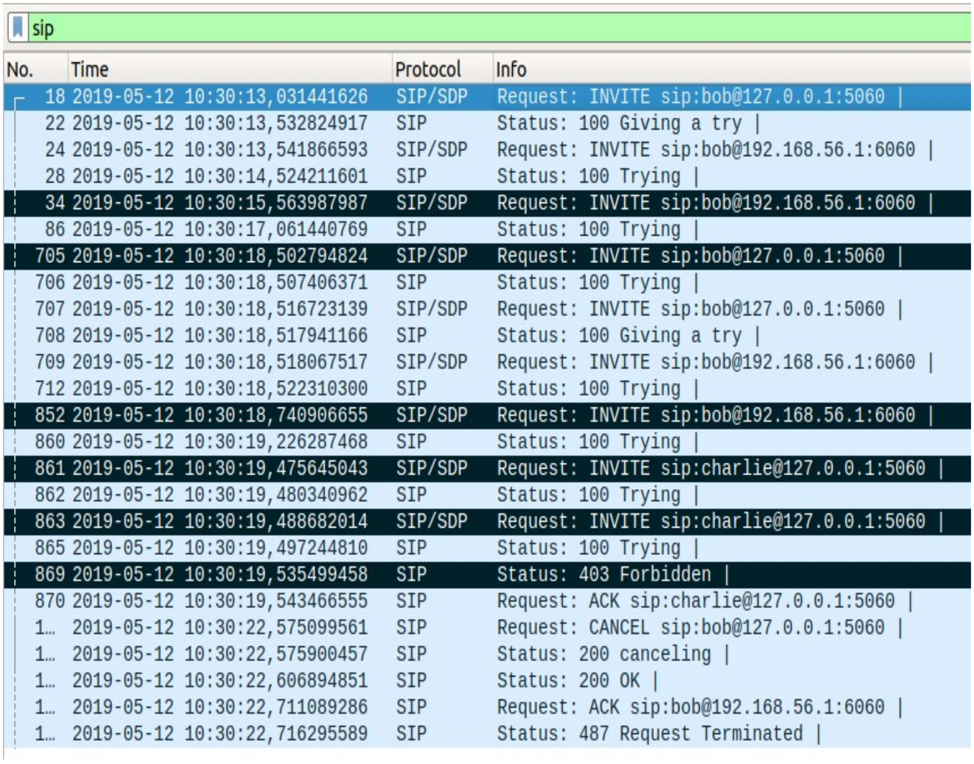
Figure 4.4: Solution Implementation Scenario

4.7 Limitations of Implementations

Implementation of services and the scenario in a smaller scale like this one comes with a few limitations. No database was deployed to maintain the subscriptions and locations of the users as it would be in an actual IMS network. For this reason, the services were hard coded with URIs of the users. The implementation also only focuses on two services and its feature interaction. The compositions hence, are very simple which makes it difficult to predict how they would function and how the logic would be handled for a larger subscriber base and more services in the composition. Nevertheless, these limitations do not affect the overall purpose of this which is to exhibit the benefits of using a SCIM.

Chapter 5 Results

5.1 Wireshark Trace



```
sip
No.  Time                           Protocol Info
  18 2019-05-12 10:30:13,031441626  SIP/SDP  Request: INVITE sip:bob@127.0.0.1:5060 |
  22 2019-05-12 10:30:13,532824917  SIP      Status: 100 Giving a try |
  24 2019-05-12 10:30:13,541866593  SIP/SDP  Request: INVITE sip:bob@192.168.56.1:6060 |
  28 2019-05-12 10:30:14,524211601  SIP      Status: 100 Trying |
  34 2019-05-12 10:30:15,563987987  SIP/SDP  Request: INVITE sip:bob@192.168.56.1:6060 |
  86 2019-05-12 10:30:17,061440769  SIP      Status: 100 Trying |
 705 2019-05-12 10:30:18,502794824  SIP/SDP  Request: INVITE sip:bob@127.0.0.1:5060 |
 706 2019-05-12 10:30:18,507406371  SIP      Status: 100 Trying |
 707 2019-05-12 10:30:18,516723139  SIP/SDP  Request: INVITE sip:bob@127.0.0.1:5060 |
 708 2019-05-12 10:30:18,517941166  SIP      Status: 100 Giving a try |
 709 2019-05-12 10:30:18,518067517  SIP/SDP  Request: INVITE sip:bob@192.168.56.1:6060 |
 712 2019-05-12 10:30:18,522310300  SIP      Status: 100 Trying |
 852 2019-05-12 10:30:18,740906655  SIP/SDP  Request: INVITE sip:bob@192.168.56.1:6060 |
 860 2019-05-12 10:30:19,226287468  SIP      Status: 100 Trying |
 861 2019-05-12 10:30:19,475645043  SIP/SDP  Request: INVITE sip:charlie@127.0.0.1:5060 |
 862 2019-05-12 10:30:19,480340962  SIP      Status: 100 Trying |
 863 2019-05-12 10:30:19,488682014  SIP/SDP  Request: INVITE sip:charlie@127.0.0.1:5060 |
 865 2019-05-12 10:30:19,497244810  SIP      Status: 100 Trying |
 869 2019-05-12 10:30:19,535499458  SIP      Status: 403 Forbidden |
 870 2019-05-12 10:30:19,543466555  SIP      Request: ACK sip:charlie@127.0.0.1:5060 |
 1…  2019-05-12 10:30:22,575099561  SIP      Request: CANCEL sip:bob@127.0.0.1:5060 |
 1…  2019-05-12 10:30:22,575900457  SIP      Status: 200 canceling |
 1…  2019-05-12 10:30:22,606894851  SIP      Status: 200 OK |
 1…  2019-05-12 10:30:22,711089286  SIP      Request: ACK sip:bob@192.168.56.1:6060 |
 1…  2019-05-12 10:30:22,716295589  SIP      Status: 487 Request Terminated |
```

Figure 5.1: Wireshark Capture of SIP Messages

The implementation of the feature interaction scenario and the attempt at a solution with the Rhino SIS made it possible to experience the capabilities of a SCIM as well as service orchestration. Figure 5.1 is a Wireshark trace that is made when a call is placed between Alice and Bob.

The blacked out traces are the interactions between the SIS (Port: 6060) and the ASes; AS1 - Call Barring (Port: 5062) and AS2- Call Forwarding (Port: 5064). The message contents blacked out trace frames (34, 705, 852, 861 and 869) are also represented with only focus on the information that the experiment is interested in, which reflects on how the header fields of User-Agent, Session and Diversion are manipulated with each transaction of the messages. The full content of one of the messages is included in Appendix C, Section C.2.

Frame 34: The first INVITE that reaches AS1 with 'method' = 'INVITE' and 'Session' = 'orig' which invokes composition 'S-Trigger Handling' for AS1.

```
User Datagram Protocol, Src Port: 6060, Dst Port: 5062
Session Initiation Protocol (INVITE)
Request-Line: INVITE sip:bob@192.168.56.1:6060 SIP/2.0 \par
Message Header
    From: <sip:alice@127.0.0.1:5060>;tag=1250587550
    To: <sip:bob@127.0.0.1:5060>
    User-Agent: Linphone/3.6.1 (eXosip2/4.1.0)
    Session: orig
```

Frame 705: The User-Agent value is changed by AS1 when the INVITE is sent back to the SIS and will reach OpenSIPS, this done to resend the INVITE back to the SIS by comparing the User-Agent value. The Session value is now 'term' indicating that the terminating service will be handled now. During the implementation, User-Agent values of 'Rhino' and 'NotRhino' were used which are just string values and have nothing to do with the tool that was used.

```
User Datagram Protocol, Src Port: 5062, Dst Port: 6060
Session Initiation Protocol (INVITE)
    Request-Line: INVITE sip:bob@127.0.0.1:5060 SIP/2.0
    Message Header
       From: <sip:alice@127.0.0.1:5060>;tag=1250587550
        To: <sip:bob@127.0.0.1:5060>
        User-Agent: Rhino
        Session: term
```

Frame 852: The INVITE is forwarded to AS2 according to the 'FS-Handling-Trigger' composition.

```
    User Datagram Protocol, Src Port: 6060, Dst Port: 5064
    Session Initiation Protocol (INVITE)
    Request-Line: INVITE sip:bob@192.168.56.1:6060 SIP/2.0
    Message Header
        From: <sip:alice@127.0.0.1:5060>;tag=1250587550
        To: <sip:bob@127.0.0.1:5060>
        User-Agent: Rhino
        Session: term
```

Frame 861: Session is changed as the call is being forwarded now and a ’Diversion’ header is added.

```
User Datagram Protocol, Src Port: 5064, Dst Port: 6060
    Session Initiation Protocol (INVITE)
    Request-Line: INVITE sip:charlie@127.0.0.1:5060 SIP/2.0
    Message Header
        From: <sip:alice@127.0.0.1:5060>;tag=A2ZryA
        To: <sip:charlie@127.0.0.1:5060>
        User-Agent: Rhino
        Session: orig-cdiv
        Diversion: sip:bob@127.0.0.1:5060;transport=UDP
```

Frame 863: The composition script checks and stores the ’Diversion’ value and removes it from this INVITE. Next forwards it to AS1.

```
    User Datagram Protocol, Src Port: 6060, Dst Port: 5062
    Session Initiation Protocol (INVITE)
    Request-Line: INVITE sip:charlie@127.0.0.1:5060 SIP/2.0
    Message Header
        From: <sip:alice@127.0.0.1:5060>;tag=A2ZryA
        To: <sip:charlie@127.0.0.1:5060>
        User-Agent: Rhino
        Session: orig-cdiv
        New-Diversion: sip:bob@127.0.0.1:5060;transport=UDP
```

Frame 869: As there is no ’Diversion’, the Request URI is compared between Charlie and Alice by AS1, sending a 403 Forbidden back to Alice.

```
    User Datagram Protocol, Src Port: 5062, Dst Port: 6060
    Session Initiation Protocol (403)
    Status-Line: SIP/2.0 403 Forbidden
    Message Header
        From: <sip:alice@127.0.0.1:5060>;tag=A2ZryA
        To: <sip:charlie@127.0.0.1:5060>;tag=IAn3cA
```

5.2 Findings from the Experiment

The findings from the implementation experiment is summarized below:

- ☐ The SIS worked with OpenSIPs without much configuration in either of them. Once the ports were assigned to them, the communication was easily established.
- ☐ It was possible to deploy the created services in external platforms. Even though the services were deployed inside the Rhino SIP Servlet, the SIS did not recognize them as local but as external services.
- ☐ Scripts in the SIS- triggers, compositions and their interceptors was utilized to manipulate the request headers to achieve the desired service composition.
- ☐ The SCIM has no internal database out-of-the-box, which means there is no possibility to store data that could potentially be utilized as part of the composition or trigger. Hence, every time the same call was made, the entire process had to happen again. This can be avoided if the processed information was available for later use as well.
- ☐ As the interaction scenario was predefined, the solution was tailored according to the need which makes it static. This might be true for any other feature interaction as the interaction management unit of the SIS does not mention anything regarding interaction detection, all solution might have to be static.
- ☐ The SIS log files were used to debug the errors during the deployment of the services as well as trigger and composition scripts. The web-based management console also allowed the overview of the deployed services and their states.
- ☐ This implementation was constructed on JAVA-based services and SIP protocol only which worked as it should. The SCIM was able to understand and use the protocol state model.
- ☐ The Wireshark trace gave insight on the session establishment time as well. But it is not enough to mention whether the SIS delayed the establishment time or not.

Chapter 6

Discussion

In the most standard form of the IMS architecture, the main communication is between the S-CSCF and the ASes to provide various combinations of services to the subscribers. In the previous chapters, the need for extension to this architecture with the incorporation of the SCIM has been illustrated. The SCIM lacks specifications till this date, but the 3GPP has published a technical report TR 23.810 [3GP09b] where some architectural requirements and alternatives for the inclusion of SCIM in the network has been discussed (Section 3.5). Suggestions from this report and research documents from the research community, this chapter proposes some functional and non-functional requirements that should considered for any SCIM implementation. In addition, this chapter also suggests one architectural choice for deployment of a SCIM.

6.1 Functional Requirements

The SCIM products existing in the market and the SCIM proposals from the research community has come up with various kinds of solution for the 'black box' that is suggested by the 3GPP. The authors of [HC09], [KRA06] and [HTMM13], for example, mention quite some requirements for the implementations but all of them do not mention the same things. After careful considerations of the existing work and experimenting with the feature interaction scenario as well as attempting to solve the issue with an existing SCIM (Chapter 4), this section suggests some of the most important functionalities that should be included in any SCIM product. It is worthy to mention that these suggestions are made in the light of IMS but can easily be extended towards any other technology that has provision for a similar type of service broker functions.

- **Resolution of conflict:** It should be possible for the entity to be able to solve service invocation conflicts between two or more services that are required to be active in the same dialogue based on static and dynamic information. Responses from services in a service composition may imply contradictory behavior that need to be resolved and handled in subsequent dialogue. When intercepting the response from one particular service and replacing the operation, that service still needs to get the impression in the subsequent dialogue that replacement was not done. These conflict resolution conditions should be available for both intra and inter domain services, i.e., the services that are hosted on local and external platforms.

- **Protocol awareness:** Complex protocols like SIP, TCAP and many others have strict rules for inter-message constraints for what forms valid message sequences, both in terms of allowed types and message parameters. State transition models are often used to capture some of these constraints. Making the SCIM protocol-aware will imply that the composition framework can maintain state models for all involved dialogues with the service applications, fundamentally simplifying the rule set which otherwise would become prohibitively complex.

- **Execution of Services:** The SCIM should be able to have some expressiveness in case of service execution. For example, serial execution of several services, parallel execution of services, mixed composition as well as the ability to execute same service twice in the same composition.

- **Promote flexibility in the core:** The SCIM should be flexible in terms of decoding composition pattern and dynamic decision making depending on received message types and contents, provisioned "products" for users, previous paths chosen when computing compositions as well as different composition patterns chosen for different messages e.g., the same SIP dialogue.
  High-level declarative language should be used for expressing composition and parameter rules; it will be easier to define, debug and maintain, have control over side effects and state that will hopefully result in better maintainable orchestration definitions.

- **Ability for parameter computation:** Powerful mechanisms for constructing invocation parameters based on messages received from the network and parameter responses from previous services in the same composition, provisioned properties at various levels and the ability to express any set of computation over the available variables and state - all of these should be used by the SCIM for parameter computation which would be used later when necessary.

- **Capability for parameter manipulation:** Manipulation of parameters can be a strong tool for conflict resolution. The SCIM should be capable of adjusting the parameters when invoking each individual service in the composition, use different parameters for each service, use a section of response from one service when building invocation parameters for services invoked afterwards, ability to build final response of total composition by taking responses for all "part" services into account while still maintaining dialogue states, so sub dialogues are handled correctly.

- **Compatibility with Protocols:** By protocols it means telco protocols such as SIP and SIP based protocols, SS7 TCAP based protocols: CAP, INAP, MAP, ISUP, Diameter and Diameter based protocols, REST, SOAP etc. The SCIM should understand protocol-specific state models in order to correctly handle requests and responses with appropriate default handling. It is also necessary for the SCIM to have multi-protocol compatibility as it allows it to be more flexible for integration with other technologies. It should be able to both invoke over other protocols as well as invoke sub-services using other protocols like the IP Multimedia Service Switching Function (IM-SSF / reverse IM-SSF) [3GP11]. SCIM should also be capable to invoke sub services using several other protocols (mix) in same composition.

- **Reuse of services:** Ability to reuse existing services by creative composition will allow launching innovative new service offerings. Instead of needing to build (as well as test, launch and maintain etc.) new service applications, new services and offerings can rather be launched by composing and tailoring behavior of existing services. This will allow faster delivery with lower cost and probably better quality.

- **Service life cycle support:** There are different changes that occur during the life cycle of a service. The services constituting the composition will evolve over time, business requirements or integration requirements may evolve over time that makes it necessary to modify the composition over time. Changes may involve changed invocation pattern, introduction of new sub services or removal of some services from composition, changes in rule for parameter computation and manipulation. The SCIM must be able to synchronize with such changes of version upgrades, commissioning and decommissioning of the other services.

- **Reactive and non-reactive invocation:** SCIM computation is usually triggered by some event from the surroundings, such as, from the network or from an application taking part in a service composition. In both cases, the SCIM can execute a composition and invoke sub services in defined order. SCIM actions can also be triggered "from-the-top", e.g. from web services, even invocation can also be timer based.

- **Clever and economic representation of state:** SCIM surely needs to maintain some minimum state information, e.g. identifiers for ongoing dialogues, selected composition patterns etc. Orchestration rules may require that dialogue state is maintained by SCIM. However, all unnecessary representation of state should be avoided, as it will pose problems for scalability and fault tolerance mechanisms.

- **Independence from vendor lock-in:** The SCIM should communicate with services irrespective of the vendor such that the it is not depended on any individual vendor's service logic.

- **Access to HSS:** The SCIM must implement a Diameter interface similar to the Sh interface [3GP06] to have native access to the HSS for User-Data-Request, which is the data transferred from the HSS to the AS. Access to subscriber data in the HSS can potentially be used for service orchestration or conflict resolution at the SCIM.

- **Storage provisioning:** A native data storage will open up possibilities for storing data that could be used for service orchestration as well as conflict resolution. This will also allow the SCIM to be more optimized as it will have local, already processed data for decision making.

6.2 Non-functional Requirements

The SCIM should have non-functional characteristics that enhance its integration into the IMS network. Non-functional requirements specify criteria that can be used to judge the operating capabilities of the system or entity, rather than specific behaviors (which are described as the functional requirements). Some of the most vital non-functional requirements are mentioned below. These requirements are general for any system that serve the service broker purpose and is not limited to only IMS.

- **Scalability:** The SCIM should be flexible to the size of the network. Scalability is an important feature for the SCIM. As the SCIM entity is placed between the S-CSCF and service layer, if it is not possible to be scaled, it might create a bottle neck in the network. The functional requirement of representation of state if not managed correctly can also pose threats to the scalability.

- **Fault tolerance:** The fault tolerance of the entity should be high, otherwise if the SCIM is not working as it should, it will have an extensive impact on the network. High fault tolerance and fail safes hence, are important features to be considered for any SCIM entities.

- **Session establishment time:** SCIM is adding another component between the S-CSCF and ASes. Hence, it is expected to change the previous session establishment time. But since, the inclusion of SCIM is not supposed to impact the existing network and its components, the impact of any additional processing at the SCIM should be minimal such that it does not adversely affect the call session.

- **Easy integration with existing core:** The SCIM should be able to integrate easily with the IMS network, which means that no significant changes should be required in the S-CSCF as well as the ASes in the Service Layer. The SCIM should also be compatible with the existing IMS protocols (e.g., SIP Protocol) in order to integrate as well as extend the existing protocols.
- **Security:** One the functional requirements mention native access to the HSS which holds all user related information as well as storage provisioning by the SCIM that will store other user information, it is necessary that the SCIM has secure communications with the HSS and the data it uses and stores cannot be accessed without correct authentication. In addition, supporting encryption, signatures, certificates etc. should be in place.
- **Management:** The SCIM should have a management console or dashboard that allows the checking available services and triggers as well as display alarms in case the SCIM experiences any kind of connectivity issues among other things.

6.3 Architectural Choice

In section 3.5 a technical report from the research community reflected some light on the architectural aspects of a SCIM. Even though it mentions three aspects including the hybrid approach, in this section two of the approaches are considered, namely the centralized approach and the distributed approach. The hybrid approach is considered as an extension of the distributed approach.
The authors of [GSD09] suggest the centralized approach as it decouples the service layer from call control. They also argue that the approach can facilitate SCIM to SCIM communication which can allow the MVNOs (Mobile Virtual Network Operator) to collaborate in an attempt to provide innovative services. On the other hand, an article by Gouya, Crespi and Bertin [GCB06] supports the distributed approach. They describe in detail that the distributed approach allow the interaction management task to be delegated to the S-CSCF and all the management decisions are not lying on the SCIM as in case of the centralized approach. They further elaborate that this approach provide a scalable solution in interaction management and enable service level inter-working.
Scalability is an important feature for the IMS network as there should always be provisions for extensions of the network. This is provided more by the distributed approach than the centralized approach. Another concern for the centralized approach, is the eventual bottlenecks that appear in this approach [GCB06]. Since the SCIM is placed between the S-CSCF and ASes and it is the the only entity of communication

between the two components, a failure in one of the SCIM will have lesser impact on the network than the centralized approach. Even though the centralized approach is more fitting for smaller networks like MVNOs as mentioned by [GSD09], distributed approach seems to be a better choice for networks due to its better fault tolerance giving this approach better reliability and scalability.

6.4 Evaluating Existing SCIM Products

Since the functional and non-functional requirements of a SCIM is discussed, it is essential to take a look into the SCIMs that were described in Section 3.6. Of the two SCIMs, the Rhino SIS has been used for experiments in this thesis, which makes the evaluation for this tool to be based on both the documentation available and also practical experience. The Lucent Service Broker™ , on the other hand, will only be evaluated based on the available documentation.

6.4.1 Lucent Service Broker™

Functional Fulfillments:

- The steplets of the service broker are able to select applications not only based on the information from the iFC but also uses user data from the HSS. This allows the steplets to detect conflicts among the services to be called. The steplets are able to solve them by dynamic selection of applications and even compensate for inappropriate actions taken by the applications. The steplets are also able to invoke multiple services both in a serial and parallel manner.
- The SB engine is protocol-aware of the common telecom protocols and can use the parameters for composition invocation and interaction management when required.
- The broker is able to reuse services deployed in the network as the steplets can select ASes flexibly and dynamically using the information from the ASes and the user data from the HSS.
- Services from any vendors can be accessed by the broker engine, allowing the network free from vendor lock-in.
- The Lucent Service Broker™ also has access to the HSS that allows the steplets to use user data for dynamic selection of the ASes.

Non-functional Fulfillments:

- The default steplet in the broker engine is able to invoke any number of steplets to complete the request in question. This allows the broker to be scalable according to the size of the network.
- The Lucent Service Broker™ also has a support system for fault tolerance. Even though it is not part of the broker engine, it is part of the whole network for operations and maintenance. There is no further elaboration on this in [KRA06].
- According to [KRA06], the Lucent Service Broker™ can be introduced into the existing IMS network without any modification to the core indicating easy integration with the core.
- Sessions can be initialized by the steplets while an INVITE request is handled. Hence, session establishment time is not impacted considerably.

Functional Constraints:

- The broker engine is only compatible with SIP protocol making it less flexible for integration with other protocols.
- The Lucent Service Broker™ is JAVA-based i.e., it is not declarative which limits flexibility in the core.
- There is no mention of parameter computation and manipulation in the documentation available for the SB.
- The engine is connected to many databases accessing information that is used by the steplets but there is no storage option for the SCIM to store processed user information.

Non-functional Constraints:

- There is no management information available for the SCIM, which will make keeping track of the entity difficult.
- The broker engine communicates with the HSS but does not mention any security measures taken to protect the user data.

6.4.2 Rhino Service Interaction SLEE(SIS)

Functional Fulfillments:

- Easily integrable with the existing core as it is compatible with the existing core. Instead of requiring separate modifications for every application a service interacts with, it is possible to write the service once and then add compositions for new applications [Met11b].
- Triggers and compositions are at the core of the SIS for interaction management. Scripting is a powerful tool that allows the manipulation of messages according to the need of the user. This makes the resolution of conflict a strong point for the SIS.
- The SIS is compatible with the existing protocols of the IMS. The documentation provides use of both SIP and IN protocols and mentions the that other protocols can be included as well with their JAVA API extensions.
- SIS allows high service reuse as the services are called using triggers and compositions so particular service interaction logics, if any, in the service are not taken into account.
- Each vendor's service logic does not influence a service from another vendor in the SIS allowing it to have independence from vendor lock-ins.
- The SIS is a declarative entity that allows integration of both SIP and non-SIP applications.

Non-functional Fulfillments:

- Configurable fault tolerance is one the features of the SIS that can be configured on a deployment-by-deployment basis for each service, to determine how Rhino should behave under various failure conditions [Opea]. In case of a failure, the Rhino platform will continue operate so that the SLEE and the services it is running are continuously available. This allows to avoid impact of the failure on the network and the users.
- The security model of the SIS is based on the standard Java security model; the Java Authentication and Authorization Service (JAAS) and the SLEE specification default permission sets for components [Opec], that prevents untrusted resource adaptors, services or human users from performing restricted functions.

- The Rhino SIS has command-line management consoles for all its components as well has a web-based console known as the Opencloud Rhino Element Manager [Opeb] that allows the managers to have an overview of the services and components at play.

Functional Constraints:

- Even though SIS has a good scripting tool for conflict resolution, it does not provide any method for conflict detection. This makes the detection a manual task for the operators deploying the services.
- The SIS has no access to the HSS. It is strongly depended on the S-CSCF to provide it any and all information for the services compositions which is only the iFC. This makes the SIS less susceptible to smart service conflict resolution.
- Only non-reactive invocation of services is possible with the SIS. As the SIS does not use any extra network or user information it seems that there is no information on reactive invocation.
- There is also no mention of storage provisioning for the SIS.

Non-functional Constraints:

- There can be multiple Rhino SIS deployed in the same network, but there is no mention about the communication between the multiple SIS which makes it difficult to be scalable according to the need of the network.
- The SIS has some impact on the session establishment time. If there are multiple triggers that need to be evaluated before reaching one that evaluates as true, it has an overall impact on the time.

Chapter 7 Conclusion

This thesis presented a proposal for the functional and non-functional requirements for a SCIM - an IMS node that has not yet been standardized. It also suggests the architectural choice for the SCIM. The proposals put forward in this thesis are done through extensive research into the existing IMS network and the research that has been done on the topic of SCIM or service broker. The thesis also experimented with one of the existing SCIM products in the market to solve an feature interaction among two services in order to test the expressiveness and limitations of the SCIM that may be present. A SCIM entity can be included in the existing IMS network without making any major changes in the network. Even though it is unknown if the SCIM will ever be standardized, the proposals of this thesis can be utilized to evaluate any SCIM. As the networks and service compositions become more complex with time, introduction of a SCIM in the network will prove to be an important step that will help the service providers to save cost while enabling greater user involvement in the execution of services.

References

[3GP00] 3GPP. IP-Multimedia Subsystem. http://www.3gpp.org/more/109-ims, 2000. Last accessed 24 October 2018.

[3GP06] 3GPP. 3GPP TS 29.329 version V10.03.0 Release 10. http://www.qtc.jp/3GPP/Specs/29329-a30.pdf, 2011-2006.

[3GP07] 3GPP. 3GPP TR 32.810 version V7.0.0 Release 7. http://www.qtc.jp/3GPP/Specs/32810-700.pdf, 2007.

[3GP09a] 3GPP. 3GPP TS 23.002 version V5.12.0 Release 05. http://www.qtc.jp/3GPP/Specs/23002-5c0.pdf, 2003-2009.

[3GP09b] 3GPP. 3GPP TR 23.810 version V8.0.0 Release 8. http://www.qtc.jp/3GPP/Specs/23810-800.pdf, 2009.

[3GP09c] 3GPP. 3GPP TS 23.198 version V8.0.0 Release 8. https://www.etsi.org/deliver/etsi_ts/123100_123199/123198/08.00.00_60/ts_123198v080000p.pdf, 2009.

[3GP10] 3GPP. 3GPP TS 24.604 version V11.04.0 Release 11. https://www.etsi.org/deliver/etsi_ts/124600_124699/124604/11.04.00_60/ts_124604v110400p.pdf, 2012-2010.

[3GP11] 3GPP. 3GPP TS 23.218 version 10.0.0 Release 10. https://www.etsi.org/deliver/etsi_ts/123200_123299/123218/10.00.00_60/ts_123218v100000p.pdf, 2011.

[3GP12a] 3GPP. 3GPP TS 29.228 version V5.22.0 Release 05. http://www.arib.or.jp/english/html/overview/doc/STD-T63v9_20/5_Appendix/Rel5/29/29228-5m0.pdf, 2010-2012.

[3GP12b] 3GPP. 3GPP TS 24.229 version V5.26.0 Release 05. https://arib.or.jp/english/html/overview/doc/STD-T63v9_30/5_Appendix/Rel5/24/24229-5q0.pdf, 2012.

[3GP13] 3GPP. 3GPP TS 23.228 version 11.10.0 Release 11. https://www.etsi.org/deliver/etsi_ts/123200_123299/123228/11.10.00_60/ts_123228v111000p.pdf, 2013.

[3GP17] 3GPP. 3GPP TS 29.292 version V14.23.0 Release 14. www.3gpp.org/ftp/tsg_ct/WG3_interworking_ex-CN3/DRAFT.../29292-e30.doc, 2017.

[3GP19] 3GPP. Service and System Aspects. https://www.3gpp.org/specifications-groups/25-sa, 2019. Last accessed 18 May 2019.

[Acc07] Accenture. Ims architecture overview. http://wpage.unina.it/rcanonic/didattica/at/lucidi_2007/AT_2006-07_IMS_Accenture.pdf, 2007. Last accessed 03 March 2019, all the subsection of the title are also used.

[Ber16] Karel Berkovec. IMS Service Routing: Service Profile. https://realtimecommunication.wordpress.com/2016/04/29/how-to-read-initial-filter-criteria/, 2016. Last accessed 14 March 2018.

[CGM09] Gonzalo Camarillo and Miguel-Angel Garcia-Martin. *The 3G IP Multimedia Subsystem (IMS).* A John Wiley and Sons, Ltd, Publication, third edition, 2009.

[CLP08] C Chrighton, D.T. Long, and D.C. Page. JAIN SLEE vs SIP Servlet Which is the best choice for an IMS application server?, 2008.

[Dia09] Dialogic. The architecture and benefits of ims. http://www.sintel.com/bibli/telechargement/194/document_Multi.pdf, 2008-2009. Last accessed 03 March 2019, all the subsection of the title are also used.

[EP06] Oracle Emmanuel Proulx. An Introduction to SIP, Part 2: SIP Servlets. https://www.oracle.com/technetwork/articles/entarch/sip-servlet-101751.html, 2006. Last accessed 17 May 2019, all the subsection of the title are also used.

[20] European Telecommunications Standard Institute (ETSI). Digital cellular telecommunications system (Phase 2+); Signalling interworking for supplementary services. https://www.etsi.org/deliver/etsi_gts/09/0911/05.01.00_60/gsmts_0911v050100p.pdf, 1996.

[21] European Telecommunications Standard Institute (ETSI). GSM Technical Specification: Call Forwarding (CF) supplementary services, Stage 2. https://www.etsi.org/deliver/etsi_gts/03/0382/05.00.00_60/gsmts_0382v050000p.pdf, 1996.

[GC08] Anahita Gouya and Noël Crespi. Detection and resolution of feature interactions in IP multimedia subsystem . *International Journal of Network Management*, pages 315–337, 2008.

[GCB06] Anahita Gouya, Noël Crespi, and Emmanuel Bertin. SCIM (Service Capability Interaction Manager) Implementation Issues in IMS Service Architecture. *IEEE ICC 2006*, 2006.

[Gou07] Christophe Gourraud. IMS Service Routing: Service Profile. http://theimslantern.blogspot.com/2007/07/ims-service-routing-service-profile.html, 2007. Last accessed 14 March 2018.

[Gro] SIPCORE Working Group. A p-served-user header field parameter for originating cdiv session case in session initiation protocol (sip).

[GSD09] R Goveas, R Sunku, and D Das. Centralized service capability interaction manager (scim) architecutre to support dynamic-blended services in ims network, 2009.

[HC09] Cuiting Huang and Noël Crespi. Enriched SCIM for Service Composition within IMS Environment. *IEEE*, 2009.

[HTMM13] Nguyen Hung, Nguyen Thanh, Thomas Magedanz, and Julius Mueller. Toward a full implementation of scim functional block in ims framework, 2013.

[IET02] IETF. Rfc 3261 sip: Session initiation protocol. https://tools.ietf.org/html/rfc3261, 2002. Last accessed 14 March 2019, all the subsection of the title are also used.

[IET03a] IETF. Rfc 3550 rtp: A transport protocol for real-time. https://tools.ietf.org/html/rfc3550, 2003. Last accessed 14 March 2019, all the subsection of the title are also used.

[IET03b] IETF. Rfc 3588 diameter base protocol. https://tools.ietf.org/html/rfc3588, 2003. Last accessed 14 March 2019, all the subsection of the title are also used.

[Iva05] Ivelin Ivanov. JAIN SLEE, SIP Servlets, and Parlay/OSA (2nd Ed). https://ivelinivanov.blogspot.com/2005/08/jain-slee-sip-servlets-and-parlayosa.html, 2005.

[KG08] Hechmi Khilfi and Jean-Charles Grégoire. IMS Application Servers- Roles, Requirements and Implementation Technologies. *IEEE Computer Society*, pages 41–42, 2008.

[KRA06] Kristin F. Kocan, William D. Roome, and Vinod Anupam. Service Capability Interaction Management in IMS Using the Lucent Service Broker Product. *Bell Labs Technical Journal*, page 217–232, 2006.

[KT07] Srinivasan Krishnamoorthy and J. M. Torres. Ims enhanced filter and action criteria. *2007 International Conference on IP Multimedia Subsystem Architecture and Applications*, pages 1–3, 2007.

[Lin19] Linphone. Linphone products. https://www.linphone.org/products, 2019. Last accessed 09 May 2019, all the subsection of the title are also used.

[Met11a] Metaswitch. Rhino tas - telecom application server. https://docs.opencloud.com/ocdoc/books/rhino-documentation/2.6.0/rhino-home/, 2011. Last accessed 09 May 2019, all the subsection of the title are also used.

[Met11b] Metaswitch. Sis overview and concepts. https://docs.opencloud.com/ocdoc/books/sis-documentation/2.5.4/sis-overview-and-concepts/about-the-sis/index.html, 2011. Last accessed 06 November 2018, all the subsection of the title are also used.

[Opea] Opencloud. Configurable Fault Tolerance. https://developer.rhino.metaswitch.com/devportal/display/RD2v0/3.5+Configurable+Fault+Tolerance. Last accessed 18 May 2019, all the subsection of the title are also used.

[Opeb] Opencloud. Rhino Element Manager. https://docs.opencloud.com/ocdoc/books/rem/1.5.0/rem-home/. Last accessed 18 May 2019, all the subsection of the title are also used.

[Opec] Opencloud. Security. https://developer.rhino.metaswitch.com/devportal/display/RD2v3/6+Security. Last accessed 18 May 2019, all the subsection of the title are also used.

[Oped] Opencloud. SIP Resource Adaptor. https://docs.opencloud.com/ocdoc/books/sip/2.5.0/sip-resource-adaptor-home/. Last accessed 17 May 2019, all the subsection of the title are also used.

[Ope16] OpenSIPS. Opensips about. https://www.opensips.org/About/About, 2016. Last accessed 09 May 2019, all the subsection of the title are also used.

[SB07] Rene Gabner Julia Gross Sandford Bessler, Joachim Zeiss. An orchestrated execution environment for hybrid services. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.396.5408&rep=rep1&type=pdf, 2007.

[SM08] Inc. Sun Microsystems. JAIN SLEE (JSLEE) 1.1 Specification, Final Release. https://download.oracle.com/otn-pub/jcp/jain_slee-1_1-final-oth-JSpec/jslee-1_1-fr-spec.pdf?AuthParam=1558097205_f5c97a252a02193b2146fa80566d79d2, 2003-2008. Last accessed 17 May 2019, all the subsection of the title are also used.

[Sys] Cisco Systems. Rfc 5806-diversion indication in sip.

[Zoi19] Zoiper. Zoiper desktop user guide. https://www.zoiper.com/en/products, 2019. Last accessed 09 May 2019, all the subsection of the title are also used.

Appendix A

A.1 Algorithm for Call Forwarding (Unconditional) Service

Algorithm A.1 CFU Service Algorithm

```
Result: Call is Forwarded
initialization
 Get Request Header information - toUri
 if $toUri = bob$ then
    set To = URI of Charlie
    set User Agent = 'differentString'
    add Diversion = URI of Bob
    set Session = orig-cdiv
    $SC\_CALL\_BEING\_FORWARDED$
else
    set User Agent = 'differentString'
end
request.send()
```

A.2 CallForwarding.java

```java

package com.opencloud.sipservlet.proxy;

import java.io.IOException;
import java.text.MessageFormat;
import java.util.*;
import java.util.concurrent.ConcurrentHashMap;

import javax.annotation.Resource;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.sip.*;
import javax.servlet.sip.ar.SipApplicationRoutingDirective;

import org.apache.log4j.Logger;


public class CallForwarding extends SipServlet {

    private static Logger log = Logger.getLogger(CallForwarding.class);


    @Resource
    private SipFactory sipFactory;

    @SuppressWarnings("unchecked")
    @Override
    public void init(ServletConfig servletConfig) throws ServletException {
        log.info("Call Forwarding has started");
        super.init(servletConfig);

    }

    @Override
    protected void doRequest(SipServletRequest req) throws ServletException, IOException {
        if (log.isDebugEnabled()) {
            log.debug(MessageFormat.format("Received {0}{1} request for {2} call-id {3}\n{4}", req.isInitial() ? "initial " : "", req.getMethod(),
                    req.getRequestURI(), req.getCallId(), req));
        }

        if ("INVITE".equals(req.getMethod())){
            doInvite(req);
        }else if ("ACK".equals(req.getMethod())){
            doAck(req);
        }else if ("BYE".equals(req.getMethod())){
            doBye(req);
        }else if ("CANCEL".equals(req.getMethod())){
            doCancel(req);
        }

    }

    @Override
    protected void doInvite(SipServletRequest request)
        throws ServletException, IOException{
        if (log.isInfoEnabled()){
            log.info("Received : " + request.toString());
            log.info(request.getFrom().getURI().toString());
        }
        String toUri = request.getTo().toString();
        if (toUri.contains("bob")){

            B2buaHelper helper = request.getB2buaHelper();

            SipFactory sipFactory = (SipFactory)
                getServletContext().getAttribute(SIP_FACTORY)
                ;
           Map<String, List<String>> headers = new HashMap<
               String, List<String>>();
           List<String> toHeaderSet = new ArrayList<String
               >();

            toHeaderSet.add("sip:charlie@127.0.0.1:5060");
          headers.put("To", toHeaderSet);


            SipServletRequest forkedRequest = helper.
                createRequest(request,true, headers);
            SipURI sipURI = (SipURI) sipFactory.createURI("
                sip:charlie@127.0.0.1:5060");
            forkedRequest.setRequestURI(sipURI);
            if (log.isInfoEnabled()){
                log.info("forkedRequest = " + forkedRequest)
                    ;
            }
```

```
                forkedRequest.getSession().setAttribute("originalRequest", request);
                forkedRequest.addHeader("Diversion" , "sip:bob@127.0.0.1:5060;transport=UDP");
                forkedRequest.addHeader("Forward", "True");
                forkedRequest.setHeader("Session", "origcdiv");
                forkedRequest.setHeader("User-Agent", "differentString");
                forkedRequest.setRoutingDirective(SipApplicationRoutingDirective.CONTINUE, request);
                forkedRequest.send();

            }else {
                if (log.isInfoEnabled()){
                    log.info("INVITE not forwarded");
                }
                SipURI uri = (SipURI) request.getRequestURI();
                uri.setHost("127.0.0.1");
                uri.setPort(5060);
                request.setHeader("User-Agent", "differentString");
                request.addHeader("Forward", "False");
                request.setRequestURI(uri);
                request.getProxy().proxyTo(request.getRequestURI());
            }
        }

        @Override
        protected void doAck (SipServletRequest request) throws ServletException, IOException{
            if (log.isInfoEnabled()){
                log.info("Received : " + request.toString());
            }
        }

        @Override
        protected void doBye(SipServletRequest request) throws ServletException, IOException{
            if (log.isInfoEnabled()){
```

```java
            log.info("Received BYE : " + request.toString());
        }

        //sending OK directly to the first call
        SipServletResponse sipServletResponse = request.createResponse(SipServletResponse.SC_OK);
        sipServletResponse.send();

        //forwarding the BYE
        SipSession session = request.getSession();
        B2buaHelper helper = request.getB2buaHelper();
        SipSession linkedSession = helper.getLinkedSession(session);
        SipServletRequest forkedRequest = linkedSession.createRequest("BYE");
        if (log.isInfoEnabled()){
            log.info("forkedRequest" + forkedRequest);
        }
        forkedRequest.send();
        if (session != null && session.isValid()){
            session.invalidate();
        }
        return;
    }

    @Override
    protected void doCancel (SipServletRequest request) throws ServletException, IOException{
        if (log.isInfoEnabled()){
            log.info("Received Cancel : " + request.toString());
        }
        SipSession session = request.getSession();
        B2buaHelper helper = request.getB2buaHelper();
        SipSession linkedSession = helper.getLinkedSession(session);
        SipServletRequest originalRequest = (SipServletRequest) linkedSession.getAttribute("originalRequest");
```

```
            SipServletRequest cancelRequest = helper.
                getLinkedSipServletRequest(originalRequest).
                createCancel();
            if (log.isInfoEnabled()){
                 log.info("forkedRequest" + cancelRequest);
            }
            cancelRequest.send();
        }

        @Override
        protected void doSuccessResponse(SipServletResponse
            sipServletResponse) throws ServletException,
            IOException{
            if (log.isInfoEnabled()){
                 log.info(" Received: " + sipServletResponse.
                     toString());
            }
            if (sipServletResponse.getMethod().indexOf("BYE") !=
                 -1) {
                 SipSession sipSession = sipServletResponse.
                     getSession(false);
                 if (sipSession != null && sipSession.isValid()){
                      sipSession.invalidate();
                 }
                 SipApplicationSession sipApplicationSession =
                     sipServletResponse.getApplicationSession(
                     false);
                 if (sipApplicationSession != null &&
                     sipApplicationSession.isValid()){
                      sipApplicationSession.invalidate();
                 }
                 return;
            }

            if (sipServletResponse.getMethod().indexOf("INVITE")
                 != -1) {
                 // if this is a response to an INVITE, ACK and
                     forward OK
                 SipServletRequest ackRequest =
                     sipServletResponse.createAck();
                 if (log.isInfoEnabled()){
```

```
                log.info("Sending " + ackRequest);
            }
            ackRequest.send();
            //create and send OK to first call leg
            SipServletRequest originalRequest = (
                SipServletRequest) sipServletResponse.
                getSession().getAttribute("originalRequest");
            SipServletResponse responseToOriginalRequest =
                originalRequest.createResponse(
                sipServletResponse.getStatus());
            if (log.isInfoEnabled()){
                log.info("Sending OK on 1st call leg " +
                    responseToOriginalRequest);
            }
            responseToOriginalRequest.setContent(
                sipServletResponse.getContent(),
                sipServletResponse.getContentType());
            responseToOriginalRequest.send();

        }

    }
    @Override
    protected void doErrorResponse(SipServletResponse
        sipServletResponse)
            throws ServletException, IOException {
        if(log.isInfoEnabled()) {
            log.info("Got : " + sipServletResponse.getStatus
                () + " "
                    + sipServletResponse.getReasonPhrase());
        }
        // don't forward the timeout nor the Request
            Terminated due to CANCEL
        if(sipServletResponse.getStatus() != 408 &&
            sipServletResponse.getStatus() != 487) {
            //create and send the error response for the
                first call leg
            SipServletRequest originalRequest = (
                SipServletRequest) sipServletResponse.
                getSession().getAttribute("originalRequest");
                SipServletResponse responseToOriginalRequest =
                    originalRequest.createResponse(
                    sipServletResponse.getStatus());
                if(log.isInfoEnabled()) {
                     log.info("Sending on the first call leg " +
                         responseToOriginalRequest.toString());
                }
                responseToOriginalRequest.send();
           }
      }

}
```

A.3 Algorithms for Call Barring Service

Algorithm A.2 CB Service Algorithm A

```
Result: Call is Forbidden
initialization
 Get Request Header information - toUri, fromUri and session
 if $fromUri$ = $alice$ and $toUri$ = $charlie$ then
   $SipServletResponse.SC\_FORBIDDEN$
else

end
```

Algorithm A.3 CB Service Algorithm B

```
Result: Call is not Forbidden 1
initialization
 Get Request Header information - toUri, fromUri, diversion and session
 if $diversion$ != $null$ then
     if $session$ = $orig$ and $toUri$ = $bob$ then
         set Session = term
          set User Agent = 'someStringvalue'
     else
         set User Agent = 'differentString'
     end
else
   r
end
equest.send()
```

Algorithm A.4 CB Service Algorithm C

```
Result: Call is not Forbidden 2
initialization
 Get Request Header information - toUri, fromUri and session
 if $session$ = $orig$ and $toUri$ = $bob$ then
     set Session = term
      set User Agent = someStringvalue
else
   set User Agent = differentString
end
request.send()
```

A.4 CallBarring.java

```java

package com.opencloud.sipservlet.proxy;

import org.apache.log4j.Logger;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.sip.*;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class CallScreening extends SipServlet implements
    SipServletListener {

    private static Logger logger = Logger.getLogger(
        CallScreening.class);

    public CallScreening(){}

    @Override
    public void init (ServletConfig servletConfig) throws
        ServletException{
        logger.info("Call Screening service has started");
        super.init(servletConfig);

    }

    @Override
    protected void doInvite(SipServletRequest request)
        throws ServletException, IOException {
        logger.info("Request received: " + request.toString
            ());
        String fromUri = request.getFrom().getURI().toString
            ();
        String toUri = request.getTo().getURI().toString();
        String sessionValue = request.getHeader("Session");
        logger.info(fromUri);
        if (request.getHeader("Diversion") != null) {
            logger.info(toUri + " will not be screened");
            SipURI uri = (SipURI) request.getRequestURI();
            uri.setHost("127.0.0.1");
            uri.setPort(5060);
            if (sessionValue.contains("orig") && toUri.contains("bob")) {
                request.setHeader("Session", "term");
                request.setHeader("User-Agent", "someStringvalue");
            }else{
                request.setHeader("User-Agent", "NotsomeStringvalue2");
            }
            request.setRequestURI(uri);
            request.getProxy().proxyTo(request.getRequestURI());

        } else {
            if (fromUri.contains("alice") && toUri.contains("charlie")) {
                    logger.info("The call is to "+ toUri);
                    logger.info("The call is from " +fromUri);
                    logger.info(toUri + " is screened ");
                    SipServletResponse sipServletResponse = request.createResponse(SipServletResponse.SC_FORBIDDEN);
                    sipServletResponse.send();
                } else {
                    logger.info(toUri + " has not been screened");
                    SipURI uri = (SipURI) request.getRequestURI();
                    uri.setHost("127.0.0.1");
                    uri.setPort(5060);

                if (sessionValue.contains("orig") && toUri.contains("bob")) {
                    request.setHeader("Session", "term");
                    request.setHeader("User-Agent", "someStringvalue");
                }else{
                    request.setHeader("User-Agent", "
                        differentString");
                     request.setHeader("Session", "term");
                }
                    request.setRequestURI(uri);
                    request.getProxy().proxyTo(request.
                        getRequestURI());

                }
            }
        }

    @Override
    protected void doBye (SipServletRequest req) throws
        ServletException, IOException{
         logger.info("No BYE for call screening");
    }


    @Override
    public void servletInitialized(SipServletContextEvent
        sipServletContextEvent) {
         logger.info("Call Screening has been initialized");
    }
}
```

Appendix B

B.1 Trigger1.xml

```

<trigger xmlns="http://www.opencloud.com/SIS/Trigger"
    xmlns:sip="http://www.opencloud.com/SIS/Trigger/SIP"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opencloud.com/SIS/Trigger/SIP sip-sis-trigger-1.5.xsd
                        http://www.opencloud.com/SIS/Trigger sis-trigger-1.5.xsd">

  <description>
    Runs the originating composition if an originating INVITE request is received.
  </description>

  <trigger-name>Screen</trigger-name>
  <trigger-vendor>OpenCloud</trigger-vendor>
  <trigger-version>1.0</trigger-version>

  <trigger-priority>15</trigger-priority>

  <composition-ref>
        <composition-name>S-Trigger Handling</composition-name>
        <composition-vendor>OpenCloud</composition-vendor>
        <composition-version>1.0</composition-version>
        <composition-alias>S-Trigger</composition-alias>
  </composition-ref>

  <!-- Trigger on INVITE request with "orig" Route header
      parameter. -->
  <on-condition>
    <and>
        <equal a="${method}" b="INVITE"/>
        <string-contains source="${Session}" substring="orig
            "/>
    </and>
  </on-condition>

  <select>
    <!-- SIS will lookup the subscriber address in its
        subscription profiles,
          and select the originating composition, if
              available -->
        <composition-alias-ref>S-Trigger</composition-alias-
            ref>
  </select>


</trigger>
```

B.2 Trigger2.xml

```xml

<trigger xmlns="http://www.opencloud.com/SIS/Trigger"
    xmlns:sip="http://www.opencloud.com/SIS/Trigger/SIP"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opencloud.com/SIS/Trigger/SIP sip-sis-trigger-1.5.xsd
                        http://www.opencloud.com/SIS/Trigger sis-trigger-1.5.xsd">

  <description>
    Runs the originating composition if an originating INVITE request is received.
  </description>

  <trigger-name>ForwardScreen</trigger-name>
  <trigger-vendor>OpenCloud</trigger-vendor>
  <trigger-version>1.0</trigger-version>

  <trigger-priority>10</trigger-priority>

  <composition-ref>
        <composition-name>FS-Trigger Handling</composition-name>
        <composition-vendor>OpenCloud</composition-vendor>
        <composition-version>1.0</composition-version>
        <composition-alias>FS-Trigger</composition-alias>
  </composition-ref>

  <!-- Trigger on INVITE request with "orig" Route header parameter. -->
  <on-condition>
    <and>
        <equal a="${method}" b="INVITE"/>
       <string-contains source="${Session}" substring="term"/>
    </and>
  </on-condition>

  <select>
```

```
      <!-- SIS will lookup the subscriber address in its
           subscription profiles,
             and select the originating composition, if
                  available -->
            <composition-alias-ref>FS-Trigger</composition-alias
                -ref>
   </select>

</trigger>
```

B.3 compositionScreen.xml

```xml
<composition xmlns="http://www.opencloud.com/SIS/Composition"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://www.opencloud.com/SIS/Composition sis-composition-1.6.xsd">

  <composition-name>S-Trigger Handling</composition-name>
    <composition-vendor>OpenCloud</composition-vendor>
    <composition-version>1.0</composition-version>

  <script>
    <invoke service="Screening"/>
  </script>

  <debug-level>0</debug-level>
  <audit>false</audit>
</composition>
```

B.4 compositionForwardScreen.xml

```
<composition xmlns="http://www.opencloud.com/SIS/Composition"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.opencloud.com/SIS/Composition sis-composition-1.6.xsd">

  <composition-name>FS-Trigger Handling</composition-name>
    <composition-vendor>OpenCloud</composition-vendor>
    <composition-version>1.0</composition-version>

  <script>
    <invoke service="Forwarding"/>
    <if><present variable="${diversion}" />
        <then>
            <assign toVariable="${NewDiversion}" value="${diversion}"/>
            <delete variable="${diversion}"/>
            <invoke service="Screening"/>
            <assign toVariable="${diversion}" value="${NewDiversioniversion}"/>
      </then>
    </if>
  </script>

  <debug-level>0</debug-level>
  <audit>false</audit>
</composition>
```

Appendix

Appendix C

C.1 OpenSIPs Configuration File (opensips.cfg)

```
if (is_method("INVITE")) {
    xlog("------------------------$ua--------------------------------------");
    xlog("----------$hdr(Session)----------------------");
    if(!is_present_hf("Session") && $ua !="someStringValue") {
        xlog("------------block 1 -------- first rewrite");
        rewritehostport("192.168.56.1:6060");
        append_hf("Session: orig\r\n");
    }else{
        if($hdr(Session) == "term" && $ua == "someStringValue"){
        xlog("----------block 2-------- second rewrite");
        rewritehostport("192.168.56.1:6060");
        }
    }
    do_accounting("log");
        }
```

C.2 Wireshark Trace (Frame 34 Full)

```
Frame 34: 1298 bytes on wire (10384 bits),
    1298 bytes captured (10384 bits) on interface 0
Linux cooked capture
Internet Protocol Version 4, Src: 192.168.56.1,
                            Dst: 192.168.56.1
User Datagram Protocol, Src Port: 6060, Dst Port: 5062
Session Initiation Protocol (INVITE)
  Request-Line: INVITE sip:bob@192.168.56.1:6060 SIP/2.0
  Message Header
    Record-Route: <sip:127.0.0.1;lr>
    Via: SIP/2.0/UDP 192.168.56.1:6060;oc-node=101;rport;
      branch=z9hG4bKU94Bd-EoaRzjixzCL-D3uA;ext,
      SIP/2.0/UDP 127.0.0.1:5060;
      branch=z9hG4bK068a.c6df694.0
    Via: SIP/2.0/UDP
      192.168.0.215:6050;received=127.0.0.1;rport=6050;
      branch=z9hG4bK489798323
    From: <sip:alice@127.0.0.1:5060>;tag=1250587550
    To: <sip:bob@127.0.0.1:5060>
    Call-ID: 1486688370
    CSeq: 20 INVITE
    Contact: <sip:alice@127.0.0.1:6050>
    Content-Type: application/sdp
    Allow: INVITE,ACK,CANCEL,OPTIONS,BYE,REFER,NOTIFY,
      MESSAGE,SUBSC    RIBE,INFO
    Max-Forwards: 69
    User-Agent: Linphone/3.6.1 (eXosip2/4.1.0)
    Subject: Phone call
    Content-Length: 439
    Session: orig
    P-hint: outbound
    Route: <sip:192.168.56.1:5062;lr>,
      <sip:aqg9Lw.0@192.168.56.1:6060;oc-node=101;lr;
      transport=udp>
```