NTNU
Norwegian University of
Science and Technology

Identity Management in a Fixed Mobile convergent IMS environment

Trung Kien Nguyen

Master of Science in Communication Technology

Submission date:	June 2008
Supervisor:	Van Thanh Do, ITEM
Co-supervisor:	Markus Hidell, Royal Institute of Technology (KTH), Sweden

Norwegian University of Science and Technology
Department of Telematics

Problem Description

For the time being there are different technologies in the fixed networks e.g. PSTN, ISDN, ADSL, etc. and the mobile networks e.g. GSM, UMTS, etc. As a technology based on IP, IMS is a promising candidate for a fixed-mobile environment. However, there are still several challenges as follows:
First, the central protocol of IMS is SIP (Session Initiation Protocol) but the 3GPP SIP specifications for IMS are different with the native IETF SIP ones. This will lead to interoperability problem between different SIP specifications. The devices in different SIP environments may not be able to communicate with each other.
Secondly, IMS for a fixed mobile environment should also allow the users to subscribe to any type of service (fixed or mobile), to get access to these services on any arbitrary number of registered mobile and fixed devices interchangeably, to dynamically add or remove the number of registered fixed devices. In fact, the general IMS infrastructure is only a high-level scheme and does not address these requirements. A mobile user moving from a mobile domain to a fixed domain will lose a conversion or even worse will not be able to receive the calls addressed to him at the mobile domain.due to the problems with identities.
Therefore, in order to satisfy all the mentioned requirements it is crucial to have a sound identity management solution.
The goal of this thesis work is to propose a sound identity management solution for a fixed mobile convergent IMS environment. More specifically, the thesis work will be aiming at the following objectives:
- To provide a concise but clear introduction to Identity Management
- To provide a comprehensive description on how identities are organized and managed in both fixed and mobile networks
- To propose an identity management solution for a fixed mobile convergent IMS environment
- To demonstrate the soundness and feasibility of the proposed solution via the implementation of a prototype.

Assignment given: 15. January 2008
Supervisor: Van Thanh Do, ITEM

Identity Management in a Fixed Mobile convergent IMS environment

Nguyen Trung Kien

June 20th, 2008

Master of Science Thesis

Supervisor: Professor Do Van Thanh
Co-Supervisor: Professor Markus Hidell

Trondheim, Norway

Norwegian University of Science and Technology

Royal Institute of Technology, Sweden

Abstract

Today, there are still different technologies used in fixed and mobile networks environment such as cellular technologies (GSM, UMTS, 3G, etc.), wireless network technology like WLAN, wired network technology like ADSL, etc. With the usage of IMS (IP Multimedia Subsystem), all those technologies can be combined together in a fixed mobile convergence environment based on an IP-based infrastructure. Although IMS applies well for a Fixed Mobile convergent environment, there are still issues that need to be solved.

First, the IMS central protocol is SIP but there are differences between the 3GPP SIP specifications and the IETF one. This will lead to interoperability problem with different SIP specifications, and the devices between different SIP environments cannot communicate with each other.

Second, IMS for a fixed mobile environment should also allow the users to subscribe to any type of services (fixed or mobile), to get access to services on an arbitrary number of registered mobile and fixed devices interchangeably, to dynamically add or remove the number of registered fixed devices. In fact, the general IMS infrastructure is only a high-level scheme and does not support the mentioned requirements.

Therefore, in order to satisfy all the mentioned requirements it is crucial to have a sound identity management solution. The goal of this thesis work is to propose a sound identity management solution for a fixed mobile convergent IMS environment. More specifically, the thesis work will be aiming at the following objectives:

- To provide a concise but clear introduction to Identity Management
- To provide a comprehensive description on how identities are organized and managed in both fixed and mobile networks
- To propose an identity management solution for a fixed mobile convergent IMS environment
- To demonstrate the soundness and feasibility of the proposed solution via the implementation of a prototype.

Two Identity Management solutions have been proposed and analyzed as follows:

1. Modified IMS and SSO-enabled SIP system
2. Modified SIP-enabled Client

Due to time limitation, only the second solution is selected and a proof-of concept has been successfully designed and implemented. The proof-of-concept has demonstrates the following features:

- Single-Sign-On between the Mobile and Fixed domain enabling a mobile phone moving from the IMS mobile network to a SIP WLAN network without re-authentication.
- Identity Federation between the Mobile and Fixed domain enabling the delivery of calls addressed to one domain at the other domain.

Acknowledgements

I would like to express my honest thanks to my supervisor, **Professor Do Van Thanh**, for his strong direction and guidance throughout this work. I could not have completed the thesis without his constant advices that helped me to find new directions whenever I got stuck. I would also like to thank my secondary supervisor, **Professor Markus Hidell**. He helped me in reviewing, analyzing the thesis structure and gave me valuable recommendations.

Most of all, I would like to thank my parents and my friends for their encouragement and strong support.

Table of Contents

List of Figures 6
Abbreviations 7
I. Introduction 8
  1. Problem Statement 9
  2. Organization of the thesis 10
II. Background 10
  1. Identity Management 10
    1.1. Defining Identity Management 10
    1.2. The need for Identity Management 11
    1.3. Identity Management Standards 12
    1.4. Liberty Alliance 14
      1.4.1. What is Liberty Alliance 14
      1.4.2. Liberty Alliance Architecture Overview 15
  2. Identity Management in the Fixed and Mobile Networks 16
    2.1. Identity Management in the fixed network 16
      2.1.1. ENUM 17
    2.2. Identity Management in the mobile network 20
      2.2.1. The GSM Authentication Mechanism 20
      2.2.2. The 3G Network Authentication Protocol 22
  3. IMS for Fixed Mobile Convergent Environment 23
    3.1. Introduction to IMS 23
      3.1.1. IMS Architecture 23
      3.1.2. IMS Access Principles 26
    3.2. IMS for Fixed Mobile Environment 28
    3.3. Limitations 29
  4. Related Works 29
    4.1. Federated Identity Management 29
    4.2. Identity Management for a Fixed Mobile Convergent Environment 30
III. Analysis and Design of an Identity Management Solution for a Fixed Mobile Convergent IMS environment 31
  1. Analysis 31
    1.1. Identify the Actors 31
    1.2. Identify the High-level Use Cases 32
    1.3. Analyzing the Use Cases 33
      1.3.1. Use Case: Registering/Subscribing 33
      1.3.2. Use Cases: Making and Receiving Calls 33
      1.3.3. Use Cases: Federating and De-federating Identities 34
      1.3.4. Use Case: User Profile Administrating 35
      1.3.5. Use Case: Authentication 35
    1.4. Identify the problems 36
  2. Design 38
    2.1. Solution 1: Modified IMS and SSO-enabled SIP system 38
      2.1.1. Enable HTTP-Digest Authentication in IMS 38
      2.1.2. Enable Single Sign-On in SIP 40
      2.1.3. Discussion 42
    2.2. Solution 2: A modified SIP-enabled Client 43

2.2.1. Enable transparent authentication between IMS and native SIP 43
2.2.2. Enable transparent call-redirecting 48
2.2.3. Discussion 48
IV. Modified SIP-enabled Client Solution Implementation 49
1. Implementation Cases 49
1.1. SIP Server 49
1.1.1. BEA Weblogic$^{®}$ SIP Server Overview 49
1.1.2. BEA Weblogic$^{®}$ SIP Server Communication Platform 50
1.1.3. Implementation Results 51
1.2. Registrar Server 52
1.2.1. Registrar Server Components 52
1.2.2. Building the Registrar Server 53
1.2.3. Implementation Results 55
1.3. SIP Client based on Proposed Solution 55
1.3.1. Prerequisites 55
1.3.2. SIP Client-Designing Use Cases 56
1.3.3. SIP Client Designing Processes 59
2. Demonstrations 63
2.1. Preparation 63
2.1.1. Starting the Servers 63
2.1.2. Starting and Binding the Clients 63
2.2. Scenarios 65
2.2.1. Sending and Receiving Messages 65
2.2.2. Transparent Call-Redirecting 69
3. Discussion 72
V. Conclusion 74
1. Achievements 74
2. Review and Future Work 75
References 76

List of Figures

Figure 1. Basics for Identity Management 11
Figure 2. Liberty Alliance Organization Structure 14
Figure 3. Liberty Alliance Architecture 15
Figure 4. GSM Authentication Protocol 21
Figure 5. Authentication Protocol in 3G Network 22
Figure 6. IMS Architecture 25
Figure 7. IMS Registration and Authentication 27
Figure 8. IMS Registration Diagram 28
Figure 9. General use case diagram 33
Figure 10. Making and Receiving calls use case diagram 34
Figure 11. User profile administrating use case diagram 35
Figure 12. Authentication use case diagram 36
Figure 13. User moving between IMS and native SIP domains 37
Figure 14. HTTP-Digest flow messages 39
Figure 15. New SIP registration flow messages 40
Figure 16. Liberty Alliance SSO messages flow [22] 41
Figure 17. IMS Registration & Authentication 43
Figure 18. Moving from IMS to native SIP 46
Figure 19. HTTP-Digest Authentication in SIP 47
Figure 20. Functional Overview of BEA Weblogic SIP Server [29] 50
Figure 21. Communications Platform 51
Figure 22. Domain Setting 52
Figure 23. Registrar Server 55
Figure 24. Registering Setting Interface 56
Figure 25. Sending/Receiving Interface 57
Figure 26. Send command Interface 57
Figure 27. Profile Administrating Interface 58
Figure 28. Sending Request Phases [34] 60
Figure 29. Receiving Request Phases [34] 61
Figure 30. Sending REGISTER Request Phases (at the first time) 62
Figure 31. Sending REGISTER Request Phases (after the first time) 62
Figure 32. Running SIP servers 63
Figure 33. First Device Settings 64
Figure 34. Successful Register 64
Figure 35. Second Device Settings 65
Figure 36. Sending and Receiving messages 68
Figure 37. Redirecting Messages 71
Figure 38. Transparent reply 72

Abbreviations

AuC	Authentication Centre
AP	Access Point
FMC	Fixed Mobile Convergence
GPRS	General Packet Radio Service
GSM	Global System for Mobile communication
HLR	Home Location Register
IDM	Identity Management
IDP	Identity Provider
IMS	IP Multimedia Subsystem
IMSI	International Mobile Subscriber Identity
ISDN	Integrated Services Digital Network
LA	Liberty Alliance
PSTN	Public Switched Telephone Network
SAML	Security Assertion Mark-up Language
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
SLO	Single-Logout
SMS	Short Message Service
SOAP	Simple Object Access Protocol
SS7	Signaling System no. 7
SSO	Single-Sign-On
TMSI	Temporary Mobile Subscriber Identity
VLR	Visitor Location Register
WLAN	Wireless Local Area Network

I. Introduction

The idea of having a common network infrastructure capable of providing and delivering all the services originated for several decades ago with ISDN (Integrated Services Digital Networks). The advantages of such a convergent network system are the ease of management and the cost efficiency. Unfortunately, ISDN did not succeed and neither did its successor B-ISDN (Broadband Integrated Services Digital Networks) proposed a decade later. Regarding fixed mobile convergence there was an attempt with DECT (Digital Enhanced Cordless Telecommunications) which allows a portable handset to communicate with a PSTN (Public Switched Telephone Network). However, DECT has only very limited usage inside households as cordless phone. The dream of fixed mobile convergence is so far unrealized.

Lately, the IP Multimedia Subsystem (IMS) has arisen as a promising technology that facilitates the delivery of multimedia services to both the mobile and fixed network. The focus is until now on the interworking and interoperability issues that enables the extension of IMS, originally designed for the mobile network, to the fixed network. These issues include certainly a lot of technology challenges that must be solved. But, there are also issues which are not less important but have been so far neglected. One of them is the identity management, which is really challenging. The difficulty relies on the profound difference between the fixed network system and the mobile one in the way the systems view and manage the end-users. More specifically, the user identity management schemes are different in the fixed and mobile networks. In this project, we will study thoroughly these differences and propose interoperability solutions that will pave the way for using IMS in a Fixed Mobile convergent system.

When one subscribes to a convergent fixed mobile phone service, he should receive a unique ID that the fixed mobile convergence system is using to identify and deliver the services to him. He should be able to make calls and receive calls both from his mobile phone and his plain old telephone. With mobile phone with WLAN, the calls should be routed via the fixed network when he is in a WLAN area. This is a problem because the fixed network will not be able to recognize the user. In fact, the fixed network subscription, e.g. PSTN, ADSL is associated to the physical copper wire pair that goes to his house and no other identity is assigned to the subscriber. The system is not able to recognize whether it is his mobile phone that is connected via WLAN to the network or any other foreign phone. Delivering a phone call to the user’s mobile phone is hence not possible.

In the mobile network system like GSM and UMTS, the situation is completely different. Each subscriber has a unique identity called IMSI (International Mobile Subscriber Identity) that enables the network system to identify and authenticate the subscriber. The IMSI is stored securely in a tampered resistant device called Subscriber Identity Module or simply SIM. The SIM card also contains encryption functions that are used in the authentication of the subscriber when the mobile phone is switched on. The IMSI is again 1-1 mapped will a number called Mobile Subscriber Integrated Services Digital Network (MSISDN), or normally known as the telephone number of a mobile subscriber. Whenever the mobile is turned on, the IMSI number in the SIM will be used to identify the subscriber and to initiate the authentication procedure towards the mobile network

[36]. If the authentication procedure is successful, the mobile phone (or the subscriber) is granted access the network.

1. Problem Statement

For the time being there are different technologies in the fixed networks e.g. PSTN, ISDN, ADSL, etc. and the mobile networks e.g. GSM, UMTS, etc. As a technology based on IP, IMS is a promising candidate for a fixed-mobile environment. However, there are still several challenges as follows:

First, the central protocol of IMS is SIP (Session Initiation Protocol) but the 3GPP SIP specifications for IMS are different with the native IETF SIP ones. This will lead to interoperability problem between different SIP specifications. The devices in different SIP environments may not be able to communicate with each other.

Secondly, IMS for a fixed mobile environment should also allow the users to subscribe to any type of service (fixed or mobile), to get access to these services on any arbitrary number of registered mobile and fixed devices interchangeably, to dynamically add or remove the number of registered fixed devices. In fact, the general IMS infrastructure is only a high-level scheme and does not address these requirements. A mobile user moving from a mobile domain to a fixed domain will lose a conversion or even worse will not be able to receive the calls addressed to him at the mobile domain.due to the problems with identities.

Therefore, in order to satisfy all the mentioned requirements it is crucial to have a sound identity management solution.

The goal of this thesis work is to propose a sound identity management solution for a fixed mobile convergent IMS environment. More specifically, the thesis work will be aiming at the following objectives:

- To provide a concise but clear introduction to Identity Management
- To provide a comprehensive description on how identities are organized and managed in both fixed and mobile networks
- To propose an identity management solution for a fixed mobile convergent IMS environment
- To demonstrate the soundness and feasibility of the proposed solution via the implementation of a prototype.

2. Organization of the thesis

This thesis is organized as follows:

Chapter I introduces the problem statement and the objectives of the thesis.

The background theory and information about Identity Management in both fixed and mobile networks, Liberty Alliance's specifications, IMS for fixed mobile convergence environment and related works will be studied in Chapter II.

In Chapter III, we start analyzing and designing two identity management solutions for fixed mobile convergent IMS environment. One solution is then selected for implementation and demonstration in Chapter IV. Finally, Chapter V presents the conclusion and further works.

II. Background

1. Identity Management

1.1. Defining Identity Management

The essence of Identity Management as a solution is to provide a combination of processes and technologies to manage and secure access to the information and resources of an organization while also protecting users' profiles. Identity Management can provide the capabilities to effectively manage such processes both internal and external to an organization - for employees, customers, partners, and even applications, and, correspondingly, anyone or anything that needs to interact with an organization.

Identity Management (IdM) takes care of business processes, policies and technologies that enable organizations to control their users' access to IT applications and resources while protecting confidential personal and business information from unauthorized users. There is a well-defined mapping of physical world subjects to digital identities. A digital identity can be defined as follows: "A digital identity contains data that uniquely describes a person or thing (called the subject or entity in the language of digital identity) but also contains information about the subject's relationships to other entities." [1]

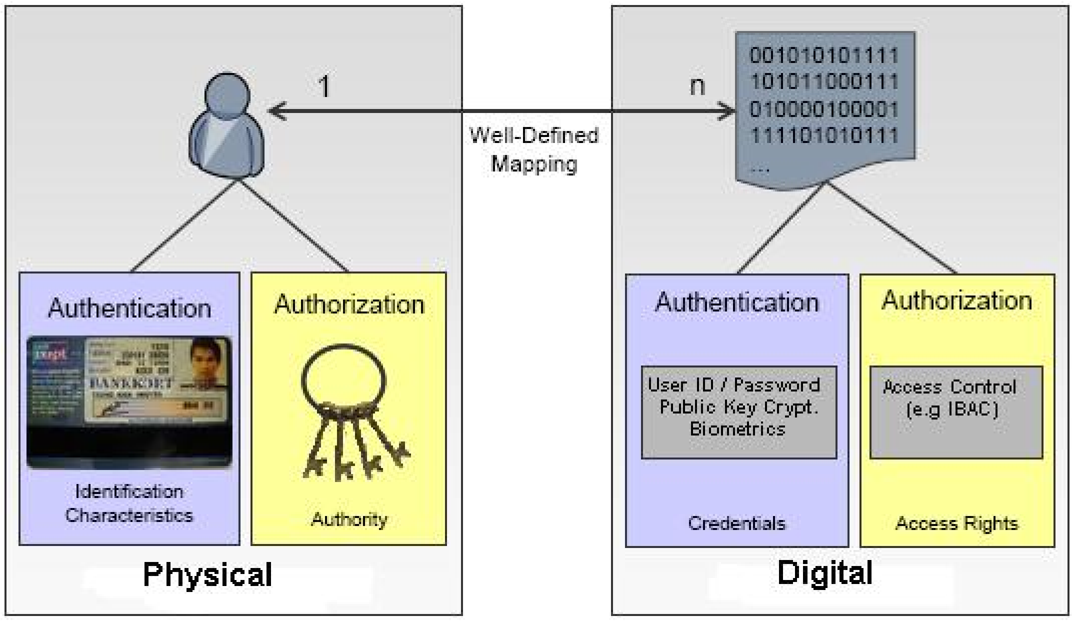


Figure 1. Basics for Identity Management

Usually a person has more than one digital identity which is then called partial digital identity. An identity comprises a set of criteria allowing a distinct characterization of an individual subject. The core of a digital identity is its universal identifier. This identifier can be a name, a number or another element which has to be unique within the treated scope. This may enable concepts like Single Sign-On (SSO) as described later on and in the part about Liberty Alliance [part 1.4].

1.2. The need for Identity Management

The need for identity management is reflected in two roles:

First, organizations have employees and own resources. They want to be able to get hold of both. Employees are mapped to user accounts and these accounts are assigned with rights to specified resources. There is the need to keep track of the identity information of their employees especially to ensure this data to be updated. As there are usually many software applications involved in a company which use their own internal user management, there is the need to keep them synchronized to ensure an overall consistency. Robust mechanisms have to be set up to prevent forging of identities in order to escalate assigned access rights.

Second, people have a different view on identity management: they want to use resources or services and they accept that they have to give some sort of authentication. Especially in large companies they often have a personal interest to keep their core data up-to-date, for instance in the way that they can be contacted. What people hate is the need to remember a huge bunch of usernames and passwords. As the technical possibility for user tracking increases, people get a little bit nervous on this issue and would like to decrease the amount of digital footprints they leave behind.

Identity management takes care of diverse functional aspects as follows:

- **The account lifecycle management**

    - Digital identities of the human beings are created and associated with access rights.
    - If any changes occur, this information has to be updated.
    - Especially if there are many different applications whose identity management has not yet been integrated, there is the need to perform a provisioning process. Provisioning means the propagation of a digital identity to all relevant destination systems. There has to be a clean-up process as well, starting if an employee leaves the company. This is called de-provisioning and takes care that all identity management related data is removed from all related systems.
    - Authorization information should not only be managed by a central instance. A delegation of this work into the corresponding enterprise parts should be possible.
    - Some aspects of data maintenance might be done by the users themselves, like the update of personal core data or a password change. Therefore self service operations should be offered.
    - A central aspect of identity management takes care of passwords and their synchronization in an often very heterogeneous IT environment.
- **Authentication and access control**

    - The aspect of Single Sign-On (SSO) takes care that a user just has to sign in once and then can use every application he has been authorized to in this environment without the need for re-authentication.
    - There are different concepts on how an effective access control can be enforced. Usually this is implemented by identity, role or attribute based concepts.

- **Integration capabilities**

    - Identity management has to take care of integration. Federation aspects exist both inside and outside of a company. Federation means that there are digital identities around that have to be aggregated on-the-fly to allow collaboration over organizational units' boundaries [2].

- **Auditing and Reporting**

1.3. Identity Management Standards

Identity Management standards have been around since the early 80's. X.500 has provided a mechanism for representing identity around the world, in a replicated and secure system since 1984 and through several revisions. In the 1990's, the rise of LDAP heralded a requirement for and resurgence in identity solutions, however LDAP gained only a modest acceptance in application developments and did not solve all the problems of Identity Management. As a result, numerous new efforts have been initiated to support Identity Management. Under the auspices of Organization for the Advancement of Structured Information Standards (OASIS) [3], several efforts have found a home.

OASIS is a nonprofit, global consortium that drives the development, convergence, and adoption of e-business standards, including:

- **SAML:** The Security Access Markup Language is intended to provide a session-based security solution for authentication and authorization across disparate systems and organizations through the use of XML expressions.

- **SPML:** The Service Provisioning Markup Language is a proposed standard for managing the process of provisioning of accounts across disparate systems.

- **XACML:** The eXtensible Access Control Markup Language is an XML specification for expressing policies for information access over the Internet. XACML is intended to define the representation for rules that specify the Who, What, When, and How of information access. Access control, which is often called rights management or entitlement management, determines who can look at something, what they can do with it, and the type of device they can look at it on.

- **WS-Security (Web Services Security):** In June 2002, the original owners of WSSecurity (IBM, Microsoft, and VeriSign) passed the WS-Security to OASIS. The intention of WS-Security is to provide support, integrate and unify multiple security models, mechanisms, and technologies, allowing a variety of systems to interoperate in a platform- and language-neutral manner. The WS-Security specification defines a set of standard Simple Object Access Protocol (SOAP) extensions (message headers) to allow the implementation of integrity and confidentiality in Web services applications. WSSecurity provides a foundation for secure Web services, laying the groundwork for higher-level facilities such as federation, policy, and trust.

In terms of Identity Management in a large-scale effort, especially focused on Internet solutions, several efforts exist from major organizations or groups in the industry to supply various degrees of identity data and related capabilities across distributed networks, some of them are:

- **Microsoft Passport:** This is one of the largest existing identity infrastructures with a claim of more than 200 million account entries. This is an example of a monolithic and centrally controlled Identity Management solution. With the reactions from the European commission and the rest of the world Microsoft has abandoned Passport and replaced it with **CardSpace.**

- **Liberty Alliance Project:** The intention of the Liberty Alliance Project is to allow distributed or federated identity services for authentication and authorization and beyond, to allow for cross-system interaction through a single logon. Released based largely on the SAML work from OASIS, the second phase is intended to provide a more extensive solution for expressing more complex security policies between organizations, focused on levels of trust. We will study about this project in the following part.

Like many single-sign on solutions, the goal of these solutions is to eliminate the need to remember multiple names and passwords, specifically while browsing the Web. To realize that it is required that data is stored and managed securely as well as being able to be securely passed between sites or businesses

1.4. Liberty Alliance

As mentioned in the previous part, Liberty Alliance proposes one of the federated identity infrastructures that allow users to connect all of their identities between accounts without centrally storing all of their personal information. The official information, statistic numbers in this part are based on [4].

1.4.1. What is Liberty Alliance

Liberty Alliance is a business alliance, formed in Sept 2001 with the goal of establishing an open standard for federated identity management. It has many members, including consumer-facing companies and technology vendors as well as policy and governmental organizations, to reach the following goals:

- Provide open standard and business guidelines for federated identity management spanning all network devices.
- Provide open and secure standard for SSO with decentralized authentication and open authorization.
- Allow consumers or businesses to maintain personal information more securely and on their terms.

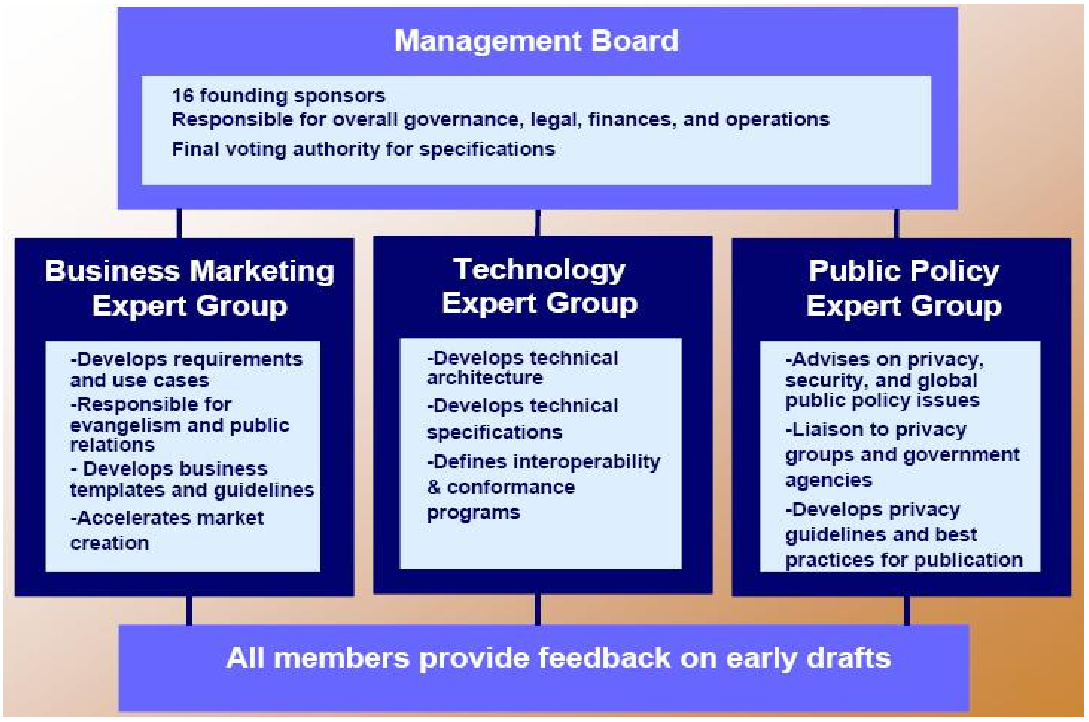

Management Board
16 founding sponsors
Responsible for overall governance, legal, finances, and operations
Final voting authority for specifications
Business Marketing Expert Group
-Develops requirements and use cases
-Responsible for evangelism and public relations
- Develops business templates and guidelines
-Accelerates market creation
Technology Expert Group
-Develops technical architecture
-Develops technical specifications
-Defines interoperability & conformance programs
Public Policy Expert Group
-Advises on privacy, security, and global public policy issues
-Liaison to privacy groups and government agencies
-Develops privacy guidelines and best practices for publication
All members provide feedback on early drafts

Figure 2. Liberty Alliance Organization Structure

The Alliance has grown from under 20 companies in 2001 to more than 160 for-profit, not-for-profit and governmental organizations today. These companies represent a billion customers and a worldwide cross-section of organizations, ranging from educational institutions and government organizations, to service providers and financial institutions, to technology firms and wireless providers.

1.4.2. Liberty Alliance Architecture Overview

Figure 3 shows the complete Liberty architecture

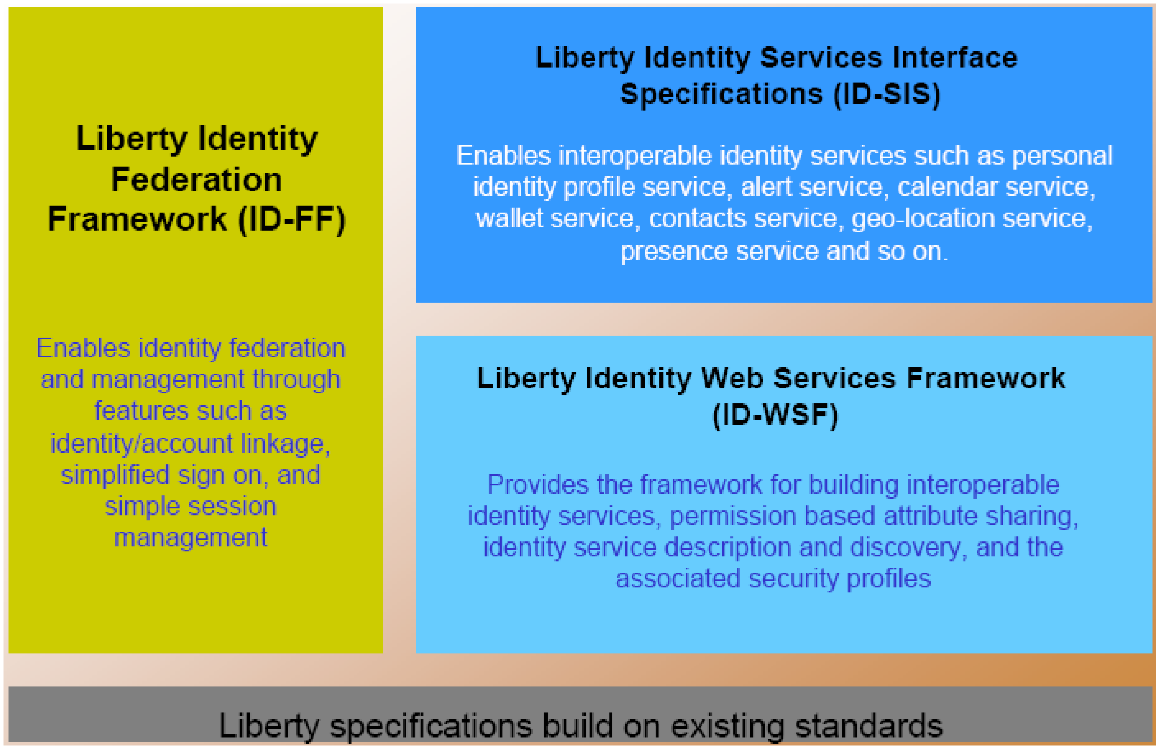

Liberty Identity Federation Framework (ID-FF)
Enables identity federation and management through features such as identity/account linkage, simplified sign on, and simple session management
Liberty Identity Services Interface Specifications (ID-SIS)
Enables interoperable identity services such as personal identity profile service, alert service, calendar service, wallet service, contacts service, geo-location service, presence service and so on.
Liberty Identity Web Services Framework (ID-WSF)
Provides the framework for building interoperable identity services, permission based attribute sharing, identity service description and discovery, and the associated security profiles
Liberty specifications build on existing standards

Figure 3. Liberty Alliance Architecture

The Liberty Identity Federation Framework (ID-FF) enables identity federation and management. This framework is suitable with almost all types of network devices and has the following features:

- **Federation**
    - Allows users with multiple accounts at different Liberty enabled sites to link them for authentication and sign in at those sites

- **Simplified Sign On (SSO)**
    - Allows users to sign on once at a Liberty ID-FF enabled site and to be seamlessly signed on when navigating to other Liberty-enabled sites without re-authenticating.

- **Global Sign Out**
    - Allows the property that once users sign out of a Liberty-enabled site, they can be automatically signed out on all the sites they have linked to (signed on) in that session.

- **Affiliations**
    - Allows users to choose to federate with a group of affiliated sites.

- **Anonymity**
    - Allows a service to request certain attributes without needing to know the user's identity.

The Liberty Identity Services Interfaces Specifications (ID-SIS) consists of specifications for interoperable services. Those independent services will be made interoperable through implementing Liberty for each service. One of the first services available in ID-SIS is the Personal Profile Identity Service, which defines mechanism for basic profile information of users.

The Liberty Identity Web Services Framework (ID-WSF) will utilize the Identity Federation Framework for creating, discovering, and consuming identity services.

The existing standards that Liberty specifications built on are SAML, WS-Security, HTTP, WSDL, XML, SOAP, XML-ENC, XML-SIG, SSL/TLS and WAP.

2. Identity Management in the Fixed and Mobile Networks

2.1. Identity Management in the fixed network

To have an overview of identity management in the fixed network environment, we will first look at the PSTN/ISDN characteristics.

The Public Switched Telephone Network (PSTN) is a common term for the traditional telephone system as it has developed through the twentieth century. A key word in PSTN is "switched" - the traditional telephone network is called a "circuit switched" network. When a telephone call is placed in a circuit switched network, a single dedicated electrical connection (a "circuit") is created (using "switches") between the calling party and the called party. That dedicated channel is maintained until the call ends.

Meanwhile, Integrated Service Digital Network (ISDN) was developed by ITU-T and ANSI to provide a digital interface between the customer equipment and the network for the transport of a wide range of digitized services such as voice, data, images and their control messages. The ISDN version used in North America was developed by ANSI, and has minor differences from the ITU-T [5]

In both PSTN and ISDN, the subscription management is based on the physical lines, either analogue (PSTN) or digital (ISDN) going to the subscriber's home. In other words, the identity of the subscriber is fully depends on the identity of the physical line which is unique and fixed to a subscriber. In this case, the identity of the line is the phone number that is given to the subscriber. As we will see in the next part, this phone number can be translated to be internet-wide using the ENUM and E164 recommendations. So, in PSTN/ISDN networks, the subscriber has no (private) identity. Indeed, the subscriber has to use the physical line identity as his identity and of course that identity can be shared between different subscribers having the same physical line. From the operators' perspective, the PSTN/ISDN network cannot distinguish which subscriber is using the line for making and receiving phone calls.

Nowadays we also have xDSL (Digital Subscriber Line) technology, or the most popular one like ADSL (Asymmetric DSL) that uses the same telephone line for data transmission. With this technology, we now have subscriber identity comparing with the PSTN/ISDN case mentioned above. The subscriber identity is defined and will be asked to be configured in the DSL modem. There are also authentication methods in xDSL technology but the subscriber identities are still based on the physical telephone line, which means different people can share the same xDSL line and the operator cannot distinguish them.

To sum up, in the PSTN/ISDN case, the only thing that can be considered as the subscriber's "identity" is the physical line itself, or the telephone number that uniquely mapped with the line. Indeed, the subscriber has no specific identity. In the xDSL case, there is a small difference: The subscriber has a specific identity, but that identity is again based on the telephone line in the same way that the telephone number is based on the physical line. Therefore, we can see that in the fixed environment, the physical telephone line is the only component used to identify the subscriber. That physical line, in a more user-friendly way, can be known as the telephone number (PSTN case) or subscriber identity (xDSL case).
.
In the next part, we will study on ENUM protocol to see how it extends the identities in PSTN/ISDN networks to be recognized and connected to IP-based networks

2.1.1. ENUM

2.1.1.1. Definition

With the need of mapping telephone numbers to IP-based networks addresses as discussed early in this part, we will need a "protocol" which is so-called ENUM[1].
At its broadest, ENUM is a technology protocol created by the Internet Engineering Task Force (IETF) that will be used to tie the traditional telephone network with the Internet [6]. As such, ENUM may prove to be an important component of the communications industry's transition away from the traditional telephone network to the Internet as the primary carrier of communications, both voice and data. Although the dedicated telephone system will likely continue to exist for years to come, voice communications will increasingly be carried over the Internet. ENUM has a variety of potential uses, but its first significant use will likely be to facilitate the carrying of voice phone calls over the

[1] The word "ENUM" had its origins in the phrases "Electronic NUMbering" and "tElephone NUmber Mapping" [21]

Internet. In supporting this and other services, it will be important to implement ENUM so that - at a minimum - it does not reduce the level of privacy and personal control available in the regular phone system.

To be simple, ENUM is a protocol that defines a method to convert an ordinary telephone number (such as my home country number, +84-4-9842443) into a format that can be used on the Internet to look up Internet addressing information (such as, for example, VoIP or email addresses). In a regular telephone number, the most significant numbers appear first - for Vietnam, the country code "84" is followed by an area code (4). In Internet domain names, however, the most significant information appears last – for example, to locate my website with the name "www.myowngate.com", a computer would first determine the location of the top level domain (.com), then the domain itself (myowngate), and then the particular server in that domain (www). Therefore, to accommodate the different convention used in Internet domain names the ENUM protocol reverses the sequence of the digits in an ordinary telephone number (and also assigns a special domain name, "e164.arpa"). The "e164" refers to the global telephone numbering system established by the International Telecommunications Union (ITU). The ".arpa" represents to a top-level domain on the Internet. Thus, the ENUM format of my telephone number +84-4-9842443 would be "3.4.4.2.4.8.9.4.4.8.e164.arpa".

The second key element of ENUM is that the Internet addressing information associated with an ENUM number is stored within the "domain name system" (DNS) providing instructions on how to reach the device associated with a particular ENUM number (which in turn is associated with a particular telephone number).
For example, if I used ENUM and was set up to receive "VoIP" calls, then my ENUM DNS record would be stored at "3.4.4.2.4.8.9.4.4.8.e164.arpa" in the Domain Name System and would point to my VoIP address (e.g. sip:trungkie@stud.ntnu.no). Thus, if someone has my telephone number and the appropriate VoIP equipment, they could place a voice call over the Internet by:

- Dialing my ordinary phone number, +84-4-9842443, using a regular phone
- Having their computer (or phone system) convert the telephone number into my ENUM number
- Looking up the ENUM record in the Domain Name System and then placing a VoIP phone call to sip:trungkie@stud.ntnu.no

A third key element of the ENUM protocol is that more than one piece of contact information can be stored in the DNS record that is associated with a particular ENUM number. Thus, an ENUM record associated with me might contain instructions for a VoIP voice call (e.g., sip:trungkie@stud.ntnu.no) or an e-mail communication (mailto: trungkien.nguyen@gmail.com). Under some conceptions of ENUM, this ability to include multiple contact methods in an ENUM record would allow an ENUM number to be a single contact number that could be given out to support, in theory, any type of communication (voice, fax, e-mail, cellular, SMS, etc.).

To sum up, we can see that the ENUM protocol provides, first, a method to use a regular phone number to look up a special type of record in the public Domain Name System, or DNS, and then second, a format to store one or more pieces of Internet contact information associated with that phone number in the DNS. A factor that is crucial to the privacy concerns discussed below is that any information stored in the DNS is completely public and accessible worldwide.

2.1.1.2. Usage of ENUM

ENUM is capable of being used in a number of different ways. One of its usages allows a single ENUM number to provide access to voice, fax, cellular, and other channels of communication - is related somehow to a identity management system's characteristic. More important things might be the ability to route telephone calls over the Internet instead of over the regular telephone system.

2.1.2.2.1. Single Number Point of Contact

As noted above, one element of ENUM permits an ENUM DNS record to include multiple methods of contact. For example, nowadays we have numerous different contact telephone numbers (such as office, home, mobile, and fax). ENUM would allow a person to provide VoIP-capable clients with one ENUM number that could reach all of those different points of contact. Similarly, some companies may choose to set up ENUM numbers – with multiple contact methods – for the companies' sales forces. Using ENUM, the companies can maximize the possibility that a customer will be able reach a sales person.

2.1.2.2.2. Transitioning the Traditional Telephone System

The first broad use of ENUM will more likely be to facilitate the transition away from the PSTN and to the Internet as the primary carrier of voice communications.

- **Corporate Bypass of the PSTN:** ENUM may initially see widespread adoption with large- and medium-sized corporations (and other institutions) seeking to reduce telephone costs by using VoIP for voice calls (and thereby bypassing local and long distance telephone companies). Corporations could implement ENUM entirely behind the scenes, and route some (but not all) phone calls over the Internet instead of the normal telephone network (the PSTN). A typical corporate implementation of ENUM would allow seamless routing of phone calls.

- **Individual Bypass of the PSTN:** Just as corporations may use internal telephone systems to bypass the local telephone service, individuals will likely be able to install Internet-to-telephone equipment in their homes and, in conjunction with VoIP and ENUM-based technology, may be able to reduce their usage of traditional telephone services. As with corporations, the more of an individual's friends, family, and colleagues who use similar Internet-based voice services, the more potential that the individual can save money.

- **Dialed Access to New Internet Services:** ENUM may also be used as a convenient way to access new Internet services from cellular telephones and other devices that lack a full computer keyboard. For cellular telephones that also have access to Internet services, users may have to use cumbersome methods to enter an Internet URL (e.g. www.myowngate.com). Service providers that are targeting wireless uses may offer an alternative method of access, by allowing users simply to dial a telephone number that can be translated - using ENUM - into the desired Internet address.

ENUM offers a range of potential benefits, and if properly implemented could increase end users' control over their privacy and their communications services more generally. It gives us the overview of how we can manage identities in circuit-switched network such as PSTN in the trend to be converged with IP-based networks.

2.2. Identity Management in the mobile network

In the PSTN/ISDN case studied above, normally there are one or more users sharing the same physical line. For example, all the members in a family can share the same fixed telephone number, or all the computers in an Internet café can share the same ADSL line. On the contrary, in the mobile network, each user can have more than one subscription. The cases of sharing the same subscription as in the PSTN/ISDN case of course exist, but are rare exceptions.

Subscriber access in mobile networks is based on a hardware token called Subscriber Identity Module (SIM) that can be placed into any mobile phone. It is a smart card with much smaller plastic substrate. It is the heart of the mobile phone identity. It can identify the phone number, the home network and the traffic to the world. The SIM card architecture represents one of the smallest computing platforms in use today. It not only can store some information in the card, but also can process that information on card and can communicate with the mobile phone terminal via application protocol data units (APDUs). In each SIM card, there is special information called International Mobile Subscriber Identity (IMSI) which makes the subscriber unique. When a user subscribe to a mobile service provider, he will get a world-wide identity known as Mobile Subscriber Integrated Services Digital Network Number (MSISDN), or can be simply understood as his mobile phone number. In a nutshell, the identity of a subscriber (or his mobile phone number) is totally based on the SIM card. This identity does not depend on the network or the terminal equipment of the subscriber. With the SIM card, a subscriber can be uniquely identified and authenticate with the serving network.

Today, when talking about mobile networks, we should mention GSM network since it is still a global dominance for mobile communication. In the next part, we will study how subscriber are authenticated and granted access in GSM networks through SIM card to see how identities are managed in GSM networks. In additions, for a broader view of mobile networks, we also take a look at authentication mechanism in 3G networks.

2.2.1. The GSM Authentication Mechanism

There are three major components in a GSM network [10]: The Mobile Station (MS) that has a subscriber identity module (SIM), which provides the user’s unique identity and stores the secret authentication key Ki; The Base Station Subsystem (BSS), which connects one user to other mobile users; The Network Subsystem (NSS), which contains elements used to authenticate MS, such as the Authentication Center (AUc), the Home Location Register (HLR) and the Visiting Location Register (VLR).

Figure 4 depicts the authentication protocol used in GSM network. When a mobile station wants to authenticate to the network, the mobile station sets up a link to the base station and relays an IMSI (international mobile subscriber identity) or a TMSI (temporary mobile subscriber identity) from the SIM to the VLR. The VLR forwards the IMSI/TMSI and its own identity to the HLR.
In GSM networks, the Authentication Center (AuC) shares a key (Ki) with the SIM: this is the shared secret between the user and the network used in the authentication challenge. When authentication is required, for example when the mobile phone performs a location update in a new Visitor Location Register (VLR), the AuC generates a random number (RAND) and sends it to the mobile. The SIM in the mobile applies the A3 algorithm on RAND and Ki. The result (Signed Response, SRES) is then sent back to the AuC that may authenticate the mobile by comparing the received response with the output of its own computation. The key used to encrypt the data on the radio link, Kc, is generated from the Ki and the RAND number by running the A8 algorithm. This key is sent by the AuC along with SRES and RAND to the VLR. These three numbers are known as Authentication Vector Response, or "the triplet".
The HLR asks its own AUc for a set of Triplets containing: a RAND (a random number), a Kc (a corresponding session key) and a SRES (a signed response). The Kc and the SRES are calculated with A3 and A8 algorithms that implement a one-way function:
Kc= A8 (Ki, RAND)
SRES=A3 (Ki, RAND)

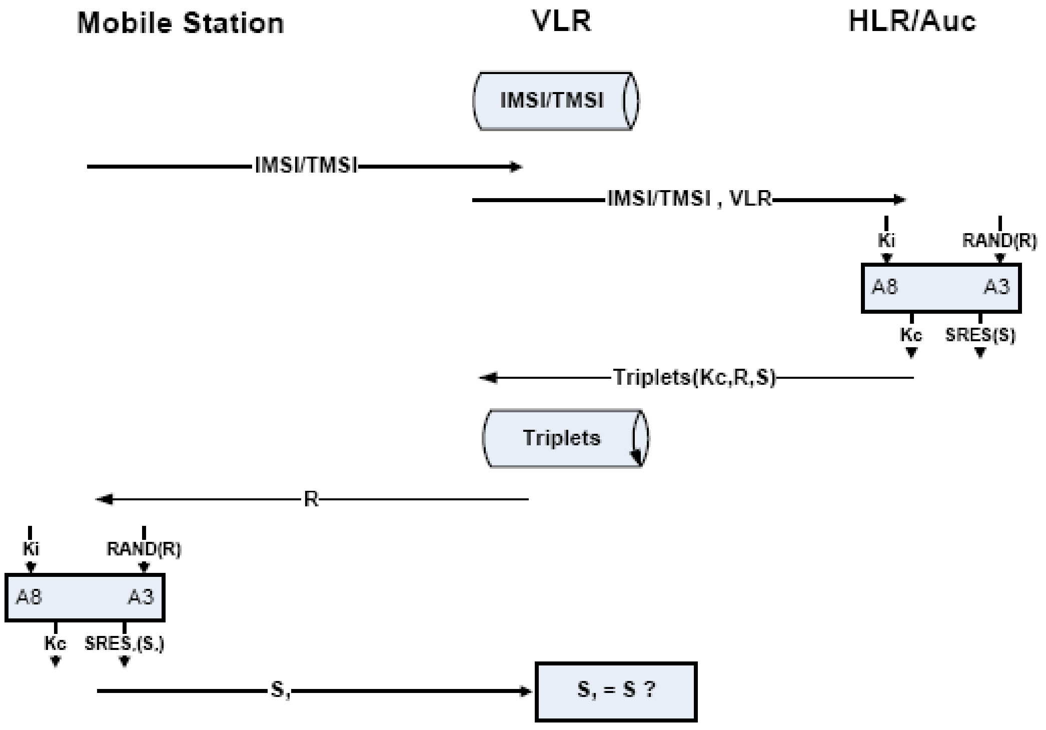


Figure 4. GSM Authentication Protocol

When the set of Triplets is sent back to the VLR, the VLR sends the RAND to the mobile station. Then, the RAND is passed to the SIM card, which is sent through the A3 algorithm together with a key (Ki). The output of the A3 algorithm is the SRES'.

Afterwards, the mobile station passes the SRES' to the VLR. Finally, the VLR compares the SRES' with SRES. If they are equal, the mobile station is authenticated. Otherwise, the mobile station will be blocked.

2.2.2. The 3G Network Authentication Protocol

The Authentication & Key Agreement protocol (AKA) [11] used in 3G network makes some improvements and overcomes some disadvantages existed in the GSM authentication protocol. It uses two-way authentication, which can protect from the fake base station attack. It adds a sequence number into a set of quintets, which can prevent the replay attack.

Figure 5 shows the data flows of the AKA protocol.

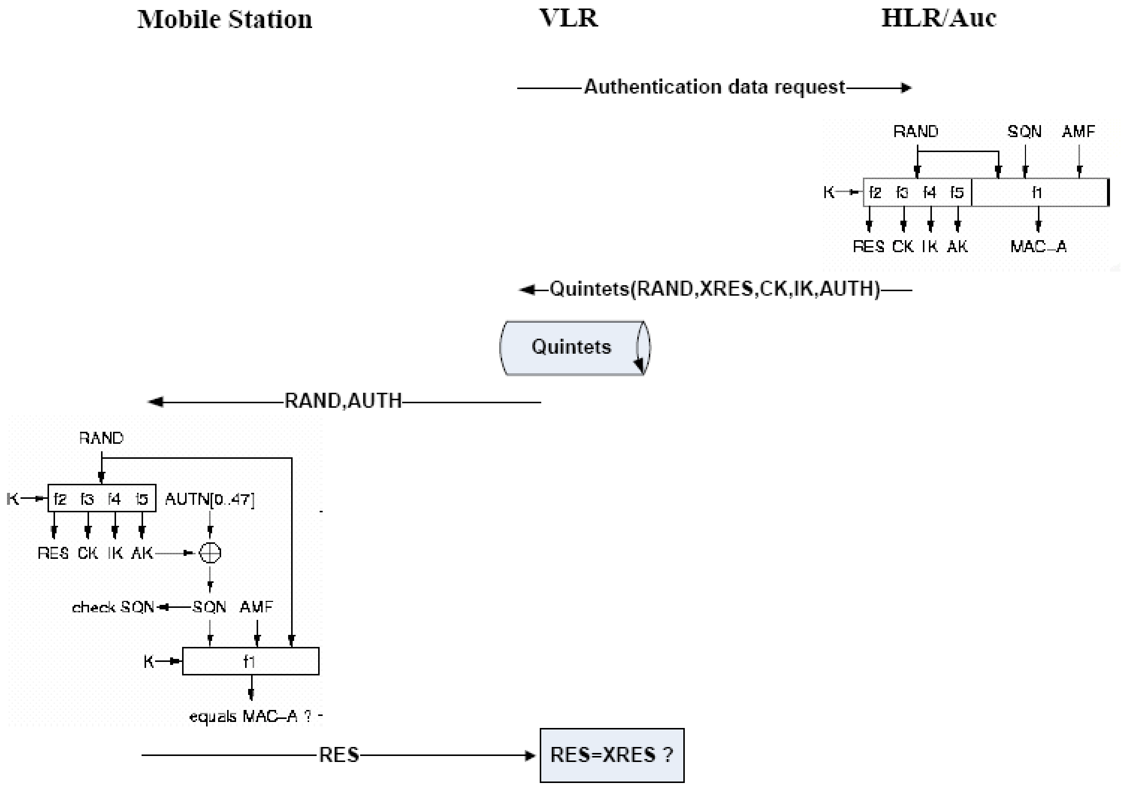


Figure 5. Authentication Protocol in 3G Network

When a mobile station wants to authenticate to the network, the VLR sends an authentication data request to the HLR/Auc. The HLR/Auc generates a RAND (a 128 bits random challenge), a SQN (a 48 bits sequence number) and an AMF (a 16 bits authentication management field). Then, the HLR/Auc computes a XRES (an expected user response), a CK (a cipher key), an IK (integrity key), an AK (anonymity key) and a MAC (a message authentication code) by using the following functions:

MAC = $f1_K$ (SQN || RAND || AMF) where f1 is a message authentication function

XRES = $f2_K$ (RAND) where f2 is a message authentication function
CK = $f3_K$ (RAND) where f3 is a key generating function
IK = $f4_K$ (RAND) where f4 is a key generating function
AK = $f5_K$ (RAND) where f5 is a key generating function or $f5 \equiv 0$.

Finally the authentication token AUTN = SQN XOR AK || AMF || MAC is constructed. The HLR/AuC sends a set of quintets (consisting of a RAND, a XRES, a CK, an IK and an AUTH) to the VLR. The VLR passes the RAND and the AUTH to the mobile station. Upon receipt of the RAND and the AUTN, the SIM first computes the anonymity key AK = f5K (RAND) and retrieves the sequence number SQN = (SQN XOR AK) XOR AK.
Next, the SIM computes a XMAC = f1K (SQN || RAND || AMF) and compares this with the MAC which is included in the AUTN. If the SQN is in the correct range and the XMAC matches the MAC, the mobile station computes a RES = f2K (RAND) and sends it back to the VLR. The VLR compares the RES with the XRES in the set of quintets. If they are equal, the mobile station will be authenticated.

3. IMS for Fixed Mobile Convergent Environment

3.1. Introduction to IMS

In a fixed mobile convergence environment, we need to integrate services provided on different technologies (cellular, IP-based) to a homogeneous network infrastructure. One of those infrastructures is IP Multimedia Subsystem (IMS) which is defined by 3GPP. IMS is a complete application layer system that is built on top of the UMTS PS (Packet Switched) domain, but is designed in such a way that it is independent of the underlying access technology. In addition to UTRAN (UMTS Terrestrial Radio Access Network) and GERAN (GSM/EDGE Radio Access Network) access, it is planned for future releases to support other kinds of access technologies such as WLANs. The crucial protocol in IMS is the Session Initiation Protocol (SIP), which is used for session management operation with multimedia services. SIP manages IP-based sessions by setting up, modifying and terminating multimedia sessions. User plane traffic is separated from control plane traffic in IMS. A major part of 3GPP Release 5 is devoted to the specification of the IMS. In this release, the aim is just to provide basic functionality (i.e., a platform on top of which various IP-based services can be developed). In Release 6, some IMS services are going to be standardized (e.g., presence, push, instant messaging and chat). Later, conversational real-time services like video-conferencing will be the target. In addition to SIP, IMS utilizes the Session Description Protocol (SDP) to negotiate the parameters used in sessions [12].

3.1.1. IMS Architecture

The information here is based on [13], [14] and [15]. The central elements of IMS are a number of SIP servers and proxies, called Call Session Control Function (CSCF). The main purposes of these are listed in the following:

- **The proxy CSCF (P-CSCF)** is the first contact point in IMS. It carries out the following functions:

- Forwards SIP registration requests received from the UE to an I-CSCF;
- Forwards SIP messages received from the UE to the SIP server (e.g., S-CSCFs) whose name the P-CSCF has received as a result of the registration procedure;
- Forwards SIP requests or responses to the UE;
- Performs SIP message compression/decompression.

- **The interrogating CSCF (I-CSCF)** is the contact point within a subscriber's home network. Its functions are:

  - As part of the registration procedure, to assign an S-CSCF to a user;
  - To route an SIP request received from another network toward the S-CSCF;
  - To forward SIP requests or responses to the S-CSCF.

- **The serving CSCF (S-CSCF)** undertakes session control services for the UE by maintaining a session state. The functions it carries out during a session are listed as follows:

  - Accepts registration requests and informs the Home Subscriber Server (HSS);
  - Session control for the registered user's sessions;
  - Interaction with service platforms for the support of various services;
  - It provides end points with service event-related information (e.g., notification of tones);
  - On behalf of both originating and terminating subscribers, it forwards SIP requests and responses to I-CSCF or P-CSCF (or to other elements in case one of the communicating parties is outside the IMS).

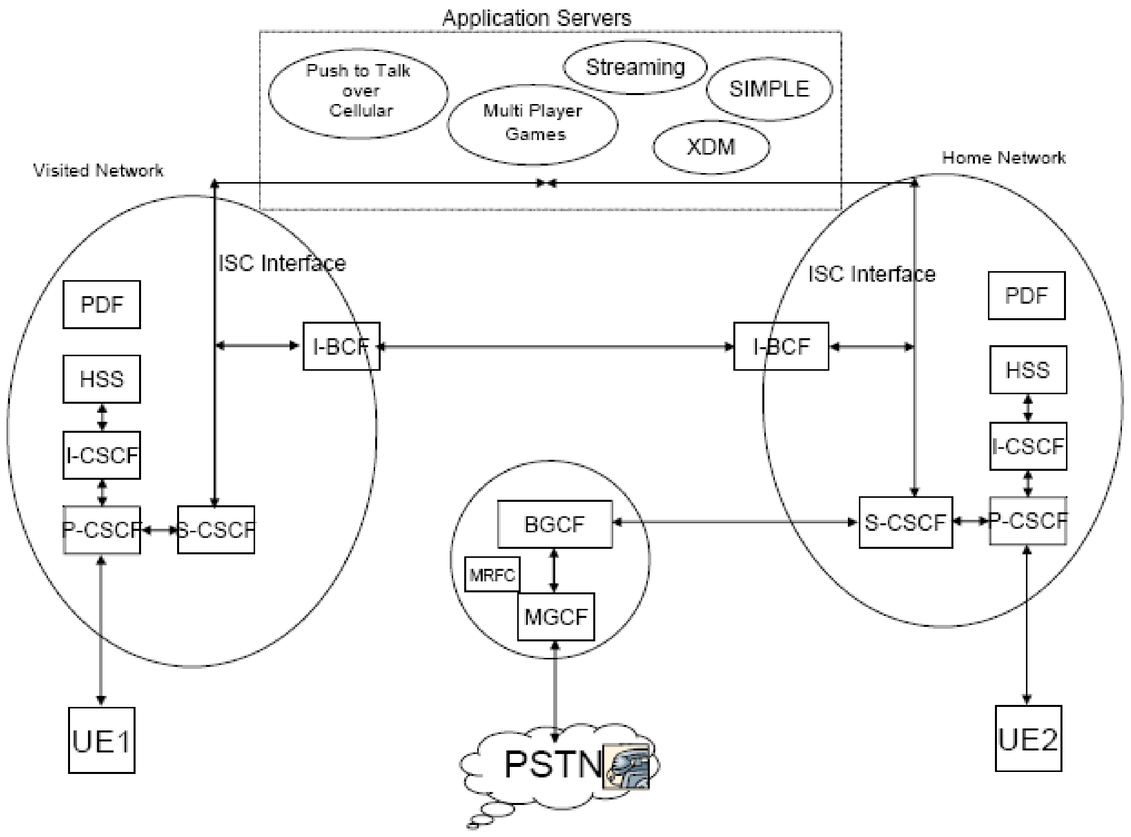


Figure 6. IMS Architecture

Although IMS was designed for mobile networks, it can also be used to provide services for fixed networks at the same time, providing unique mixtures of services with transparency for the end-user. Figure 6 gives a diagrammatic view of the IMS architecture in the context of the radio network and its importance for application server. It also shows the inter-working possibility with PSTN networks. There are some other elements that are discussed below:

- **Media Gateway Control Function (MGCF)**: IMS is designed to integrate with PSTNs and traditional telephony services such as caller ID, and local number portability. MGCF is used to transfer non–SIP packetized voice to SIP user agent (UA) and vice versa. If the calls are originating or terminating on the PSTN, the MGCF converts the PSTN time division multiplex (TDM) voice bit stream to an IP real-time transport protocol (RTP) stream and direct it to the IP address of the corresponding IP phone.

- **Media Resource Function (MRF)**: MRF is divided into two parts: Multimedia Resource Function Controller (MRFC) and Multimedia Resource Function Processor (MRFP). Tasks of MRFC are as follows:
  - Control the media stream resources in the MRFP
  - Interpret information coming from an application server (AS) and S-CSCF (e.g., session identifier) and control MRFP accordingly
  - Generate charging data records (CDR)

Tasks of MRFP are as follows:

- Control the bearer on the Mb reference point
- Provide resources to be controlled by the MRFC
- Mix incoming media streams (e.g., for multiple parties)
- Determine media stream source (e.g., for multimedia announcements)
- Handle media stream processing (e.g., audio transcoding, media analysis)
- Floor control (i.e., manage access rights to shared resources in a conferencing environment). Generate charging data records (CDR)

- **Interconnect Border Control Function (IBCF):** IBCF functionality includes the provision of network address port translation (NAPT) and firewalls. It also provides conversion between IPv4 and IPv6. IBCF controls media exchange, signaling, and policies across the operator boundary.

- **Home Subscriber Server (HSS):** HSS is a master database for users' subscription-related information. It supports the IMS entity to establish the registration/call control. HSS is the superset of the home location register (HLR) and provides users' physical location. HSS is operated in IP core networks and can be contacted through diameter protocol. It is connected to IMS network via Sh (AS) and Cx (S–CSCF) interfaces.

3.1.2. IMS Access Principles

There are two SIP procedures that play a central role both in SIP itself and in IMS: REGISTER for registration and INVITE for session establishment. Use of IMS is based on a subscription. The user makes an agreement with the IMS operator and has an IMS Private Identity (IMPI) stored in both the ISIM and the HSS. There is also a cryptographic, 128-bit master key stored in association of the IMPI. It is not intended that IMPI be used to address the user; instead, there exists at least one IMS Public Identity (IMPU) that is tied to IMPI. There may be different service profiles inside a single subscription that result in different IMPUs being tied to the same IMPI. Before a subscriber can begin to use services provided by the IMS, he must have an active registration, which can be obtained by sending a REGISTER request message to a P-CSCF. This message contains both the private address IMPI to be authenticated and at least one public identity IMPU to be registered. The P-CSCF forwards the REGISTER request to the I-CSCF, which in turn contacts the HSS to allocate a suitable S-CSCF to the user. After the S-CSCF is selected, the REGISTER message is forwarded to it. Next,
S-CSCF fetches Authentication Vectors (AVs) from the HSS. At the same time as fetching the AVs, the HSS stores the address of the selected S-CSCF. Now, the S-CSCF picks up the first AV and sends three or four parameters (excluding XRES and possibly CK) to P-CSCF via the I-CSCF. The P-CSCF extracts IK (Integrity Key) but forwards RAND (random number) and AUTN (authentication token) to the UE. The SIP message used to carry all this information is 401 Unauthorized. So, from a pure SIP perspective the first registration attempt has failed. Nevertheless, the ISIM in the UE is now able to

check the validity of AUTN, and (if the result of the check is positive) RES and IK are also computed. A parameter derived from RES is included in a new REGISTER request that is already integrity-protected by IK. The new REGISTER goes first to the P-CSCF and then is forwarded to the ICSCF, which checks the validity of the S-CSCF address with the HSS. The REGISTER is then forwarded to the S-CSCF, which then compares the parameter derived from RES with a respective parameter derived from XRES. If these two match, the OK message is sent all the way back to the UE.

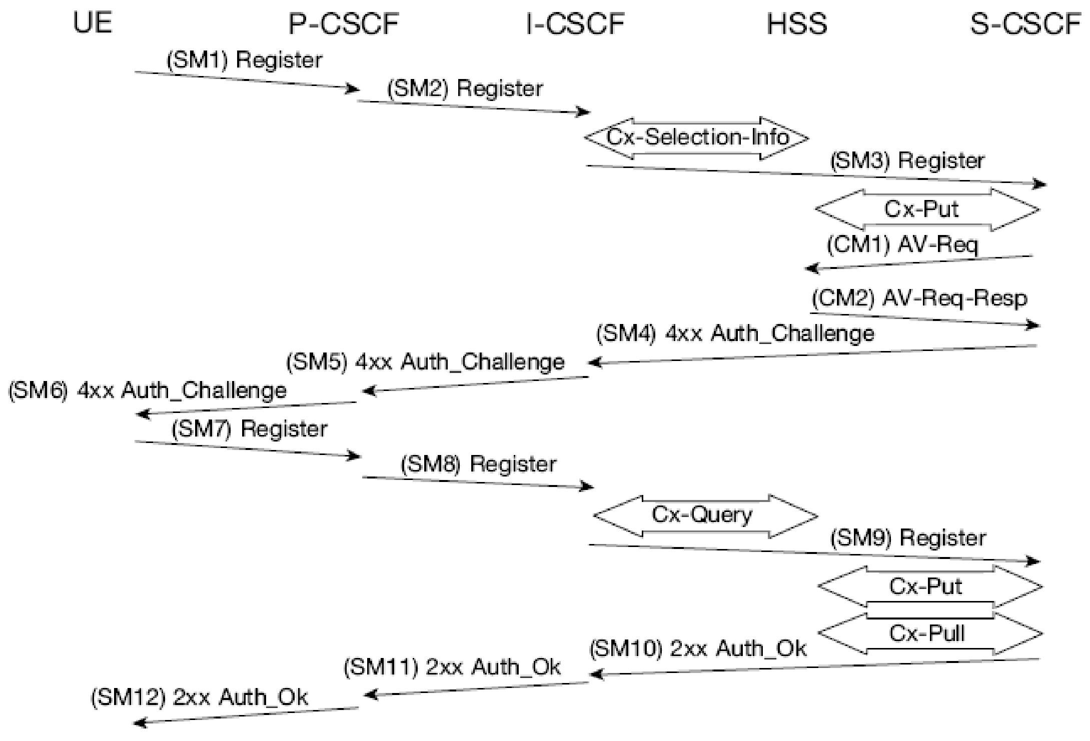


Figure 7. IMS Registration and Authentication

At this moment, the procedure is finished and the S-CSCF and HSS have both changed the status of the subscriber from ‘‘unregistered’’ to ‘‘registered’’. The registration and authentication procedure is represented in Figure 7.

For the sake of clarity, Figure 8 illustrates the IMS Registration sequence diagram respectively.

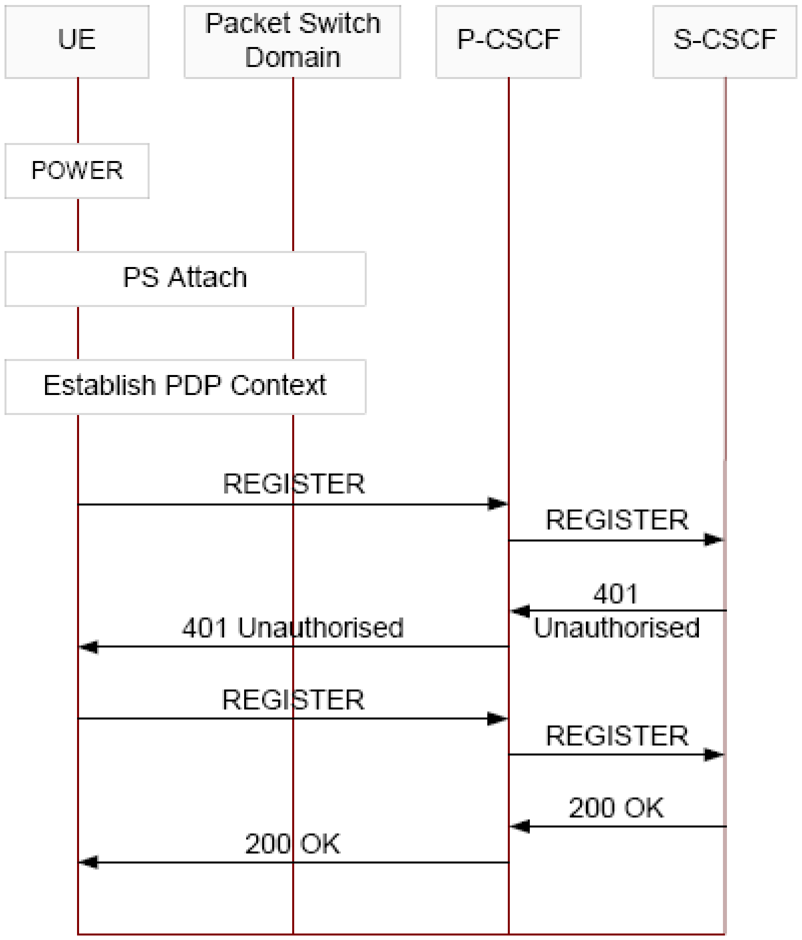


Figure 8. IMS Registration Diagram

3.2. IMS for Fixed Mobile Environment

For the time being, we have different technologies in fixed and mobile networks environment such as cellular technologies (GSM, UMTS, 3G, etc.), wireless network technology like WLAN, wired network technology like ADSL, etc. With the extension of IMS, all those technologies can work together in a fixed mobile convergence environment based on an IP-based infrastructure.

As studied in the part 3.1, each IMS subscriber has one IMPI and maybe more than one IMPU. The IMPI is stored in an IP Multimedia Service Identity Module (ISIM) which is available in any IMS-enabled 3G mobile phone. This IMPI can be digits-based form such as +47 917 34 542 or alphabet-based one like trungkie@stud.ntnu.no. Both of those forms are called Uniform Resource Identifier (URI).
For a mobile phone with WLAN enabled, IMS can support it to get access to a fixed or mobile network. In that IMS infrastructure, the WLAN access can connect to a fixed network (ADSL), go through a gateway to the Internet to reach the IMS core network,

which in turn uses IP-based technology to reach a specific network no matter it is a mobile or fixed one.
For a fixed device that works in IMS environment, with appropriate registering and federating operations, that device will be come a registered flexible device that suits the need of the users. He can have his fixed phone number forwarded to his mobile phone while he stay away from home, or has his fixed IPTV to receive calls while his mobile phone is out of order.
In fact, there should be a unified subscription for both mobile and fixed networks, regardless of the type of the devices (fixed or mobile).

3.3. Limitations

As specified earlier in the Section I, although IMS can be applied well for a fixed mobile convergence environment, there are still things need to be mentioned:

- IMS's crucial protocol is SIP but the SIP specifications between 3GPP and IETF are different. That will lead to interoperability problem with different SIP specifications, and the devices between different SIP environments cannot communicate with each other.

- IMS for a fixed mobile environment should also allow the users to subscribe to any type of service (fixed or mobile), to get access to services on an arbitrary number of registered mobile and fixed devices interchangeably, to dynamically add or remove the number of registered fixed devices. In fact, the general IMS infrastructure is only a high-level scheme, hence to satisfy all the requirements above we need to develop more specifications/implementations.

However, with the idea of identity federation studied in the part II.1, we can overcome those limitations. This issue will be studied in more details in later parts.

4. Related Works

In this part, we will try to have a look at existing Identity Management projects/solutions of different organizations/bodies, especially those projects aimed for fixed mobile convergence environment. The goal is to find the general background knowledge, the results (if any) to take advantage of in order to better serve our work.

4.1. Federated Identity Management

In fact, in terms of Federated Identity Management projects, there are numerous and most of them are being deployed at a certain level. We can list here some examples such as ***Feide*** project which is an identity management system on a national level for the educational sector in Norway [16]. This project has established a working federation for the majority of users in higher education in Norway. At this time writing this thesis (February 2008), Feide project is aiming to work on cross-federation including establishing operational cross-federations for services demanded by international users.
In June 2006, Novell unveiled an open source Identity Management project called ***Bandit***, aimed at interoperability among disparate identity systems and consistency in

securing and managing identities, which can number in the thousands for enterprises. The companies participating in this project even include Novell's top Linux competitor Red Hat. Other companies and groups that have signed onto the Bandit identity management approach, which incorporates previous standards and work including WS*, Liberty Federation, and Eclipse Higgins, are Microsoft, Sun Microsystems and IBM [17].
In April 2004, the ***Ufinity's RightAccess suite*** commercial solution of StarHub and Ufinity companies in Singapore was deployed. This solution enables StarHub to integrate its various service portals with just one single user identity. With a single-sign-on platform, StarHub's mobile, cable TV and online customers can now enjoy the convenience and ease-of-use of one single identity called Hub ID [18]

Of course there are many more similar federated Identity Management solutions. However, those solutions are more or less very locally used or peculiar to a specific number of users. In other words, they are not world-wide standardized solutions, just for the need of a small number of users.

4.2. Identity Management for a Fixed Mobile Convergent Environment

So far, the biggest project involved with this matter is the Mobicome project [19]. The main goal of that project is to provide unified user subscription management and service continuity for an IMS system deployed on a fixed mobile convergence multi access environment [19]. The second goal is to enable the development of rich communication services which make use of all the communication channels such as voice, text, video, document, etc. in various ways depending on the context.

In fact, the project has been lasted for about one year (from January 2007 till February 2008) and it is scheduled to be finished in December 2009. So far, the project has not published any Identity Management solution for a fixed mobile convergent IMS environment.

III. Analysis and Design of an Identity Management Solution for a Fixed Mobile Convergent IMS environment

In this part of the thesis, we are going to design a proof-of-concept an Identity Management Solution for a Fixed Mobile Convergent IMS environment which was studied in the part II.3. In the solution, we will also use the idea of *identity federation* to build bridges interconnecting identities across different domains (i.e. fixed and mobile environments). All the processes of the design phase will be analyzed and represented through Unified Modeling Language (UML), a language which is used to visualize, specify, construct and document a system.

This part consists of two main phases: Analysis and Design.

In the Analysis part, we will identify and construct all the high-level use cases. After that, the necessary UML diagrams like Collaboration, Sequence and Class will be introduced.

In the Design part, the desired solution will be presented will all the main components as well as methods enabling those components to work together.

1. Analysis

1.1. Identify the Actors

In a fixed mobile convergent IMS environment, a user should receive a unique ID that the fixed mobile convergent system is using to identify and deliver the services to him. He should also be able to make calls and receive calls both from his mobile phone and his fixed telephone or any registered stationary device.

Besides, in a fixed mobile convergent environment, there are, of course, two main roles that we need to specify: The Mobile Operator and Fixed Operator. As studied in the Liberty Alliance specifications, each Operator along with its appropriate Identity Provider (IdP) and Service Providers (or Contents Provider) will create a Circle of Trust (CoT).

So, we can now start to identify different actors in a fixed mobile convergent IMS environment. An actor, as defined in UML, is not any part of the being designed system. It represents a user or another system that interact with our system. In our case, there are different actors below:

- The Mobile User (user that uses services provided by mobile operators)
- The Fixed User (user that uses services provided by fixed operators)
- The Mobile Operator
- The Fixed Operator
- The Mobile Service Provider
- The Fixed Service Provider

1.2. Identify the High-level Use Cases

The user in a Fixed Mobile Convergent environment can be a mobile or fixed user interchangeably, depending on which service he is using. He will have a certain number of both mobile and fixed communication devices. With those devices, he can receive and make calls from both mobile and fixed networks.

From that point, we can see the most basic use cases as following:

- Making calls
- Receiving calls
- Registering/Subscribing

Besides, with the idea of identity federation, the user can *federate* his registered identities to another one. For example, a user has a fixed telephone at home and a mobile phone. When he goes out, he still wants to receive calls to his fixed telephone at home. The only solution is federating his fixed telephone identity to his mobile phone identity and all the calls will be forwarded as he desires. In addition to federating, when he comes back home and wants to stop receiving calls addressing to his fixed phone at his mobile one, he just simply de-federate the identity of his fixed phone at his mobile one.

So, two other use cases we can identify here are:

- Federating identities
- De-federating identities

The last issue but not less important one is how to administrate the user profiles. Therefore we have another use case of the system:

- User Profiles Administrating

So we have a general use case diagram that can be initiated by a user like below:

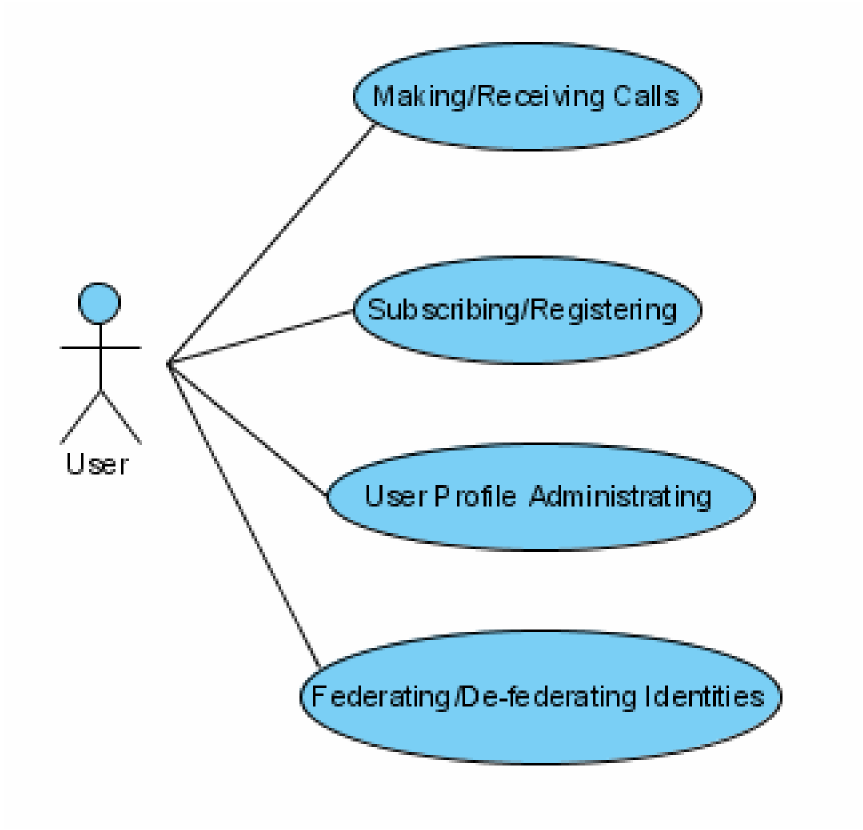

Making/Receiving Calls
Subscribing/Registering
User
User Profile Administrating
Federating/De-federating Identities

Figure 9. General use case diagram

However, as stated earlier, those six defined use cases here are all at high-level. We will continue analyzing each of them in more details in the next section.

1.3. Analyzing the Use Cases

1.3.1. Use Case: Registering/Subscribing

This is the first action that a user has to do before taking any further step to communicate with other users. A subscription may consist of both fixed and mobile services and may include more than one user. When registering, the user can specify mobile devices or stationary ones.

1.3.2. Use Cases: Making and Receiving Calls

These are normal actions of a user. For the Making and Receiving Calls use cases, there are some extended use cases that can be defined in our fixed mobile convergence environment:

- ***Calling/Receiving Calls in the same domain:*** The user makes/receives call from/at his device with a specific IMPU to/from another device (subscriber) in an IMS-enabled domain, or he makes/receives call from a pure IETF IP telephone with a specific SIP ID to/from another one in fixed IETF SIP domain.
- ***Calling/Receiving Calls across different domains:*** The user makes/receives call from/at his device in an IMS-based domain to/from an IP-Telephone in a SIP domain or vice-versa.

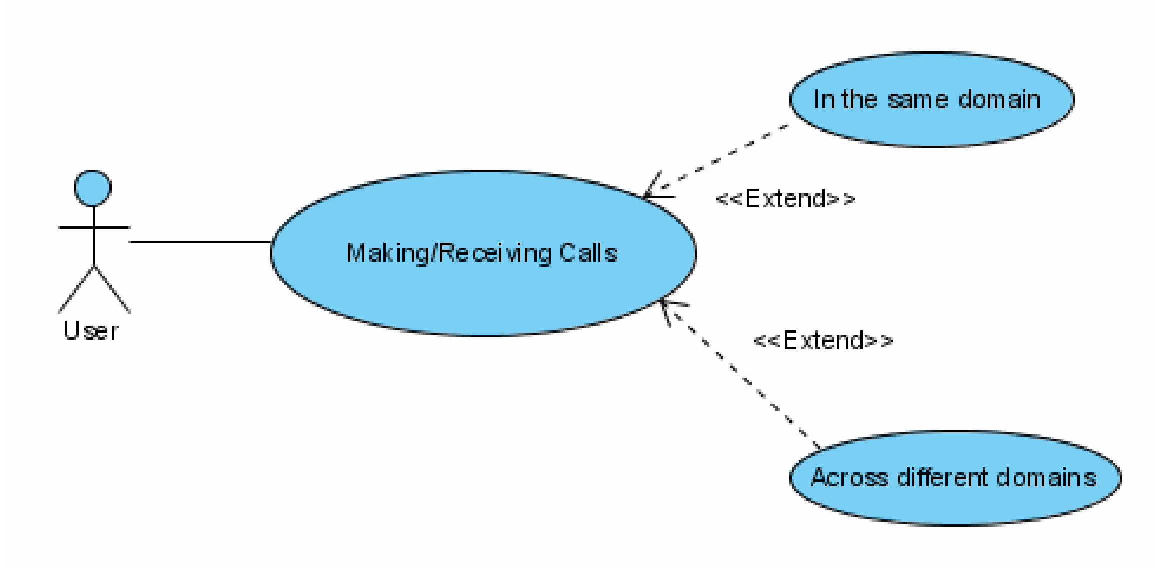


Figure 10. Making and Receiving calls use case diagram

1.3.3. Use Cases: Federating and De-federating Identities

In fact, IMS uses a SIP version which is a refinement of the SIP version developed by IETF. A standard SIP-phone cannot claim IMS compatibility; several add-ons are required (support for AKA, IPSEC, etc.). One of the biggest differences is that a standard SIP-phone implements HTTP-Digest authentication (based on password), while an IMS phone should support AKA-Digest (based on USIM or ISIM). Therefore this leads to interoperability issues between IMS and native IETF SIP domains. This problem can be solved by identity federation, so these use cases here are defined to federate/de-federate the identities between identities in different domains (IMS and native SIP).

From here, we can have the following extended use cases:

- Federating/De-federating an IMS identity to a native SIP identity
- Federating/De-federating a native SIP identity to an IMS identity

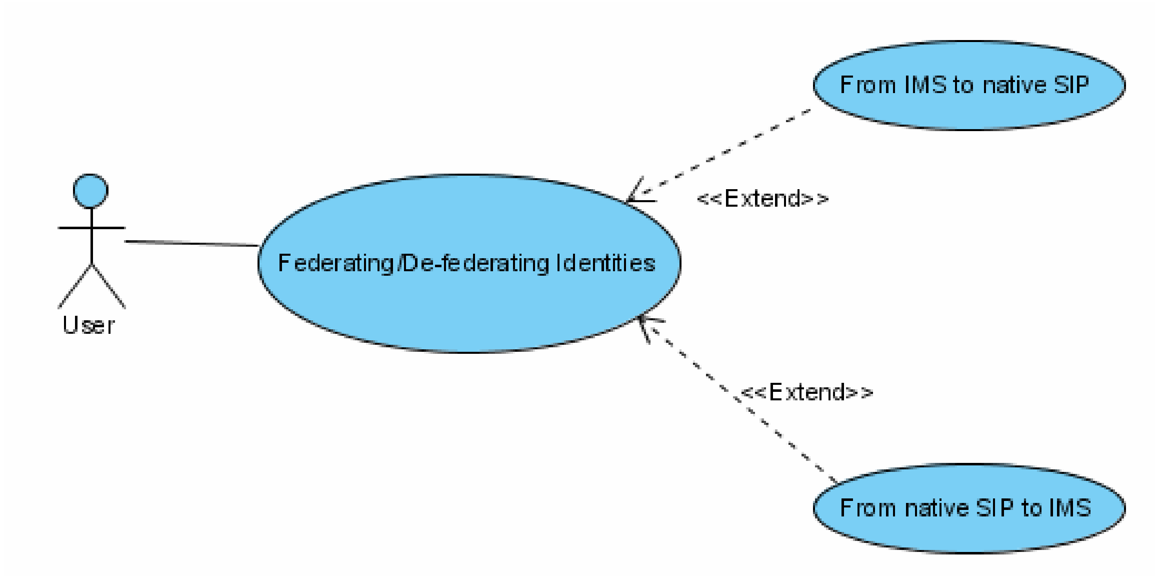


Figure 11. User profile administrating use case diagram

1.3.4. Use Case: User Profile Administrating

Each user can access his profile to update his personal details such as preferences, physical address, etc. In fact, some services are initiated based on users' profiles, for example the IDP Introduction service can be triggered based on user's physical location.

1.3.5. Use Case: Authentication

In fact, the federating/de-federating actions to be able to make/receive calls across different domains (IMS and native SIP) as well as User profile administrating can be initiated at any time depending on the user's needs. Before getting the permission to start these actions, the user must be authenticated to prevent fraudulent abuses on his identities.

In other words, we should define and include here a use case called Authentication that can be used in those use cases described above.

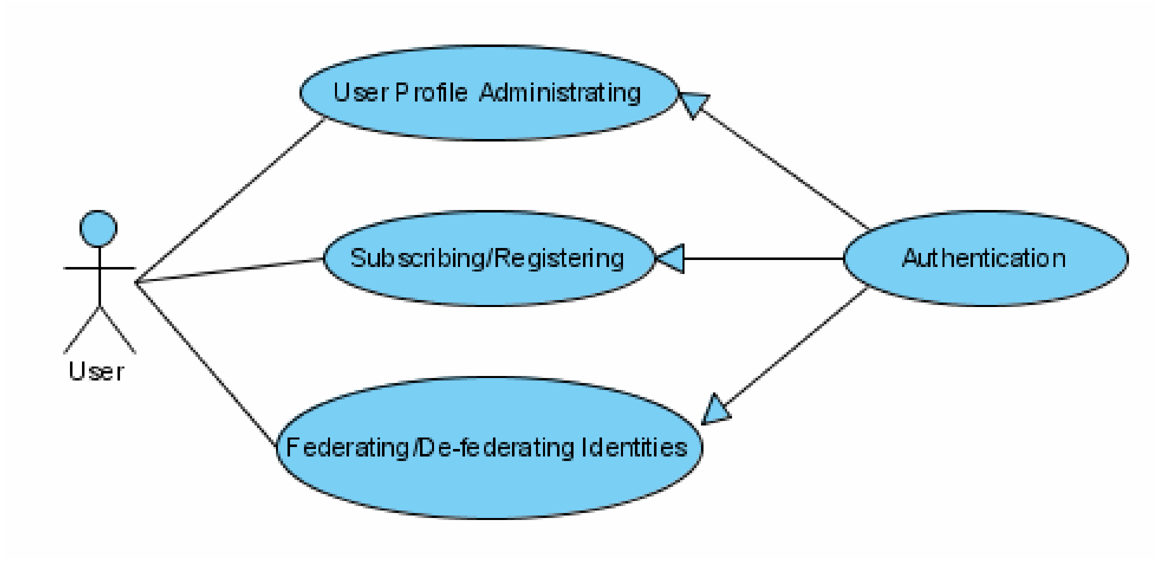

User Profile Administrating
Subscribing/Registering
Federating/De-federating Identities
Authentication
User

Figure 12. Authentication use case diagram

1.4. Identify the problems

From the previous sections, we can see that the main problem is how the new system can be able to allow users to call and receive calls transparently across IMS and native SIP domains. In other words, when a user talking with his friend on his mobile phone in an IMS-enabled domain moves to a native SIP domain, he should continue his call without knowing about the switch. The switch (from IMS to native SIP domain) should be done transparently with minimal disruption to get the high quality of the mobile calls. For example, a user has an IMPU in an IMS-enabled environment (let's say it user@ims.com) tied to his mobile phone with a fixed IMPI. He also has a SIP address in a native SIP environment (let's say it user@sip.com). When he receives or makes a call, for example in IMS domain, he will first have to authenticate with the network using his IMPI to start using his IMPU user@ims.com for his calls. This is normal behavior of all users. But when he moves to a native SIP domain while still talking on his IMPU address, the call should be redirected to his native SIP address user@sip.com instead of his IMPU user@ims.com due to the incompatibility between the two systems.

Figure 13 shows the described situation.

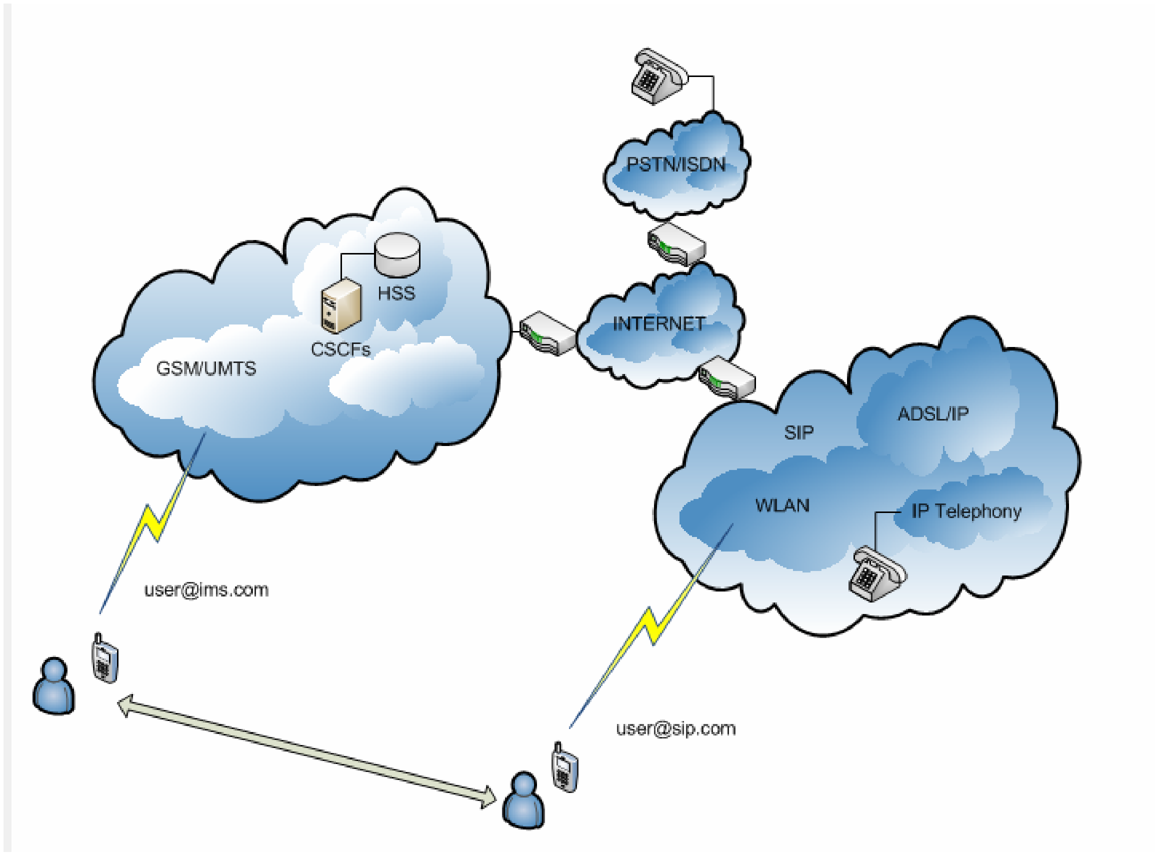


Figure 13. User moving between IMS and native SIP domains

In fact, the system should not ask the user to authenticate his native SIP address user@sip.com again since the call will be disrupted by such authentication. Therefore this problem leads to two issues:

- **Issue 1:** How can the user claim his native SIP address user@sip.com and have his call redirected to that address instead of his IMPU address?

- **Issue 2:** How can the system allow the user to use his SIP address without asking him to authenticate again? (because he is already authenticated in IMS domain before, and we do not want the user to authenticate every times he moves to new domains)

The same problems apply for the inverse situation (i.e. when the user moves from native SIP domain to an IMS-enabled one).

2. Design

The issues analyzed in the previous part can be solved separately without much effort. However, to build a system that solves all those issues while giving to end-users high quality of services in a friendly way is a challenging task.

We have come up with two possible solutions; each has both pros and cons that will be analyzed in later parts.

2.1. Solution 1: Modified IMS and SSO-enabled SIP system

In this solution, the idea is that we consider both native SIP and IMS domains are in a “big” SIP environment. This is reasonable, since IMS uses SIP as its crucial protocol. We will then modify IMS system to support the HTTP-Digest authentication as used in native SIP environment. After that, with the foundation of Liberty Alliance project studied in Section II.1.4, we will modify SIP protocol in order to enable Single Sign-On (SSO) in SIP framework.

What we achieved so far is both modified IMS and native SIP can support authentication method as in native SIP environment (i.e. HTTP Digest), and when a user logs in to a SIP environment, he can have the capabilities of a SSO-enabled environment. The answer for our problems analyzed above will be given at the end of this section.

2.1.1. Enable HTTP-Digest Authentication in IMS

In the part 1.3 of this Section, we have pointed out that one of the biggest differences between IMS and native SIP environments is that a native SIP-phone implements HTTP-Digest authentication (based on password), while an IMS phone supports AKA-Digest (based on USIM or ISIM).

In IMS, the only authentication and key agreement (AKA) system which was specified by the 3GPP was IMS-AKA. This is an authentication system based on USIM/ISIM approach. The UMTS Subscriber Identity Module (USIM) and IMS Subscriber Identity Module (ISIM) are found on the Universal Integrated Circuit Card (UICC) of 3G devices. USIM-based IMS-AKA and ISIM-based IMS-AKA authentication require the phone to be equipped with the operator UICC card (i.e. a 3G SIM card) with USIM or ISIM capability. In the absence of UICC (because of hardware design limitation), ISIM can be provided as a software module on end-devices (PC for example).

In SIP, HTTP-Digest is a challenge/response approach defined for authentication of messages and access to services. In this approach, unlike that of IMS-AKA, the challenges are not pre-computed (as in UICC) but are computed in real time by S-CSCF.

To have the minimum impact on IMS architecture, we will maintain the existing headers of SIP messages (i.e. Authentication-Info header) used in the register phase.

Figure 14 shows the HTTP-Digest flow messages.

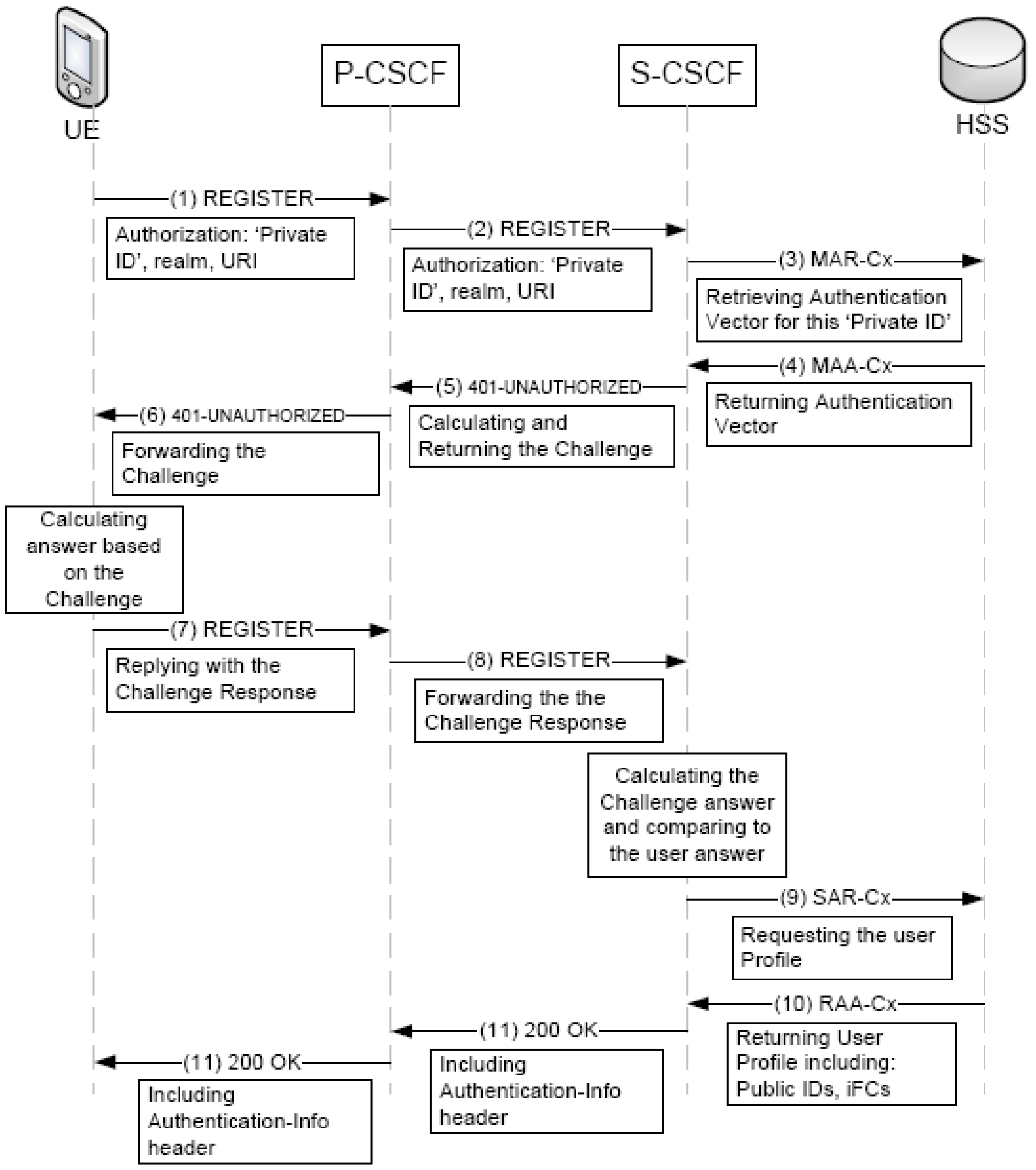

P-CSCF
S-CSCF
UE
HSS
(1) REGISTER
Authorization: 'Private ID', realm, URI
(2) REGISTER
Authorization: 'Private ID', realm, URI
(3) MAR-Cx
Retrieving Authentication Vector for this 'Private ID'
(4) MAA-Cx
Returning Authentication Vector
(5) 401-UNAUTHORIZED
Calculating and Returning the Challenge
(6) 401-UNAUTHORIZED
Forwarding the Challenge
Calculating answer based on the Challenge
(7) REGISTER
Replying with the Challenge Response
(8) REGISTER
Forwarding the the Challenge Response
Calculating the Challenge answer and comparing to the user answer
(9) SAR-Cx
Requesting the user Profile
(10) RAA-Cx
Returning User Profile including: Public IDs, iFCs
(11) 200 OK
Including Authentication-Info header
(11) 200 OK
Including Authentication-Info header

Figure 14. HTTP-Digest flow messages

To add the HTTP-Digest supporting capability in IMS, we need to modify the following of its components [25]:

- **Modifying the S-CSCF:** Any S-CSCF should be able to compute the challenges/response required for authentication of the user. Besides, the S-CSCF has to include an Authenticate-Info header in 2xx responses (for example: the 200 OK message) after a successful authentication of a user's terminal.

- **Modifying the HSS:** In IMS, the HSS communicates with other CSCFs through Cx interface (specified in Diameter protocol). Therefore, the Cx interface should be modified. Indeed, digest authentication adds new parameters to be transmitted via Cx interface to the S-CSCF. These parameters allow S-CSCF to compute the required challenge responses in the registration phase. In addition, HSS itself should store all the required parameters for challenge computation in each user profile.

2.1.2. Enable Single Sign-On in SIP

As studied earlier, Single Sign-On (SSO) enables users to access multiple services and resources with one-time login. In this part, we will build SSO capabilities to the SIP framework on the basis of the Liberty Alliance specifications. The reason why we chose Liberty Alliance as the basis of our SIP SSO design is that, as studied earlier, Liberty Alliance is one of the most popular industrial open standards specifying identity management issues. To summarize, SSO process is defined in Liberty Identity Federation Framework, known as ID-FF and ID-FF takes the Security Assertion Markup Language (SAML) as the protocol reference, which is an XML standard for exchanging authentication and authorization data between security domains.

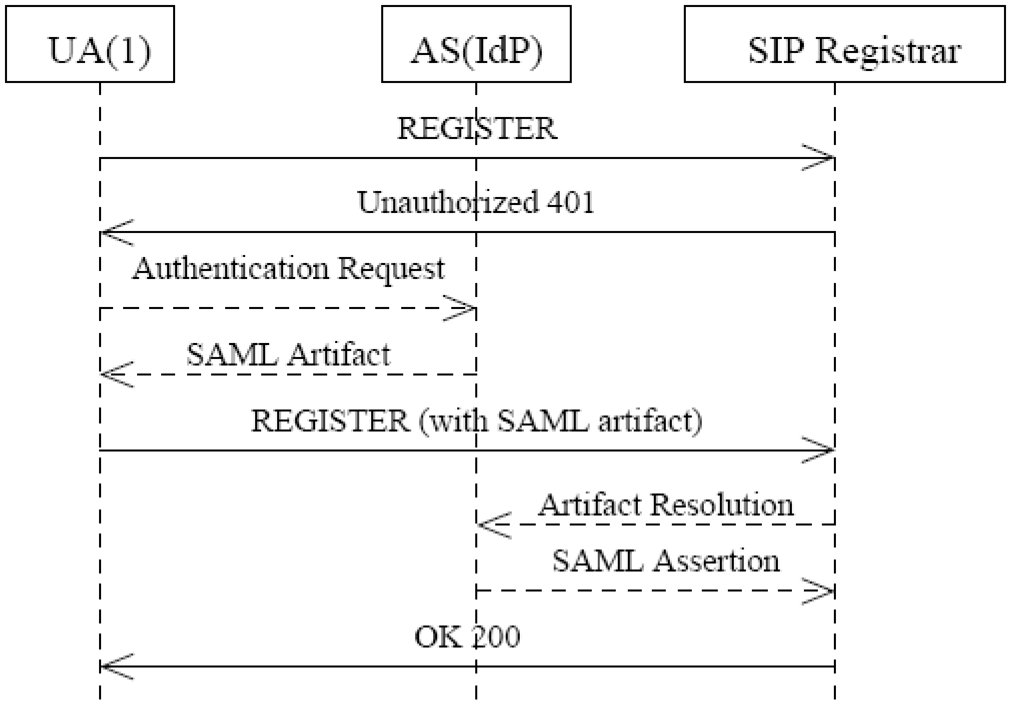


Figure 15. New SIP registration flow messages

Figure 15 illustrates the message flow during SIP registration in the new SIP system. A SIP proxy generates response UNAUTHORIZED 401 with its authentication request (Authn-Request) enclosed when receiving an unauthenticated REGISTER. The user agent (UA) extracts the authentication request from the response and forwards it to the Identity Provider (IDP). Based on a pre-defined agreement with the service provider (SP), IDP will issue a corresponding assertion for the authentication request and return the SAML artifact of the assertion to the user agent. By encapsulating the artifact, user agent resends REGISTER to the proxy, which verifies the artifact with IDP for the actual assertion. If the verification passes, the proxy returns the response OK 200 and a shared secret will be exchanged between the caller and SIP proxy for subsequent sessions. The verification is defined in the Liberty Single Sign-On and Federation Protocol, with respect to the ID-FF Liberty Artifact Profile. The REGISTER can be replaced by any other SIP request such as INVITE and SUBSCRIBE. The authentication requiring response could be either UNAUTHORIZED 401 or PROXY AUTHENTICATION REQUIRED 407, depending on SIP elements (i.e. registrar or proxy).

The Liberty Alliance's SSO messages flow is illustrated in Figure 16.

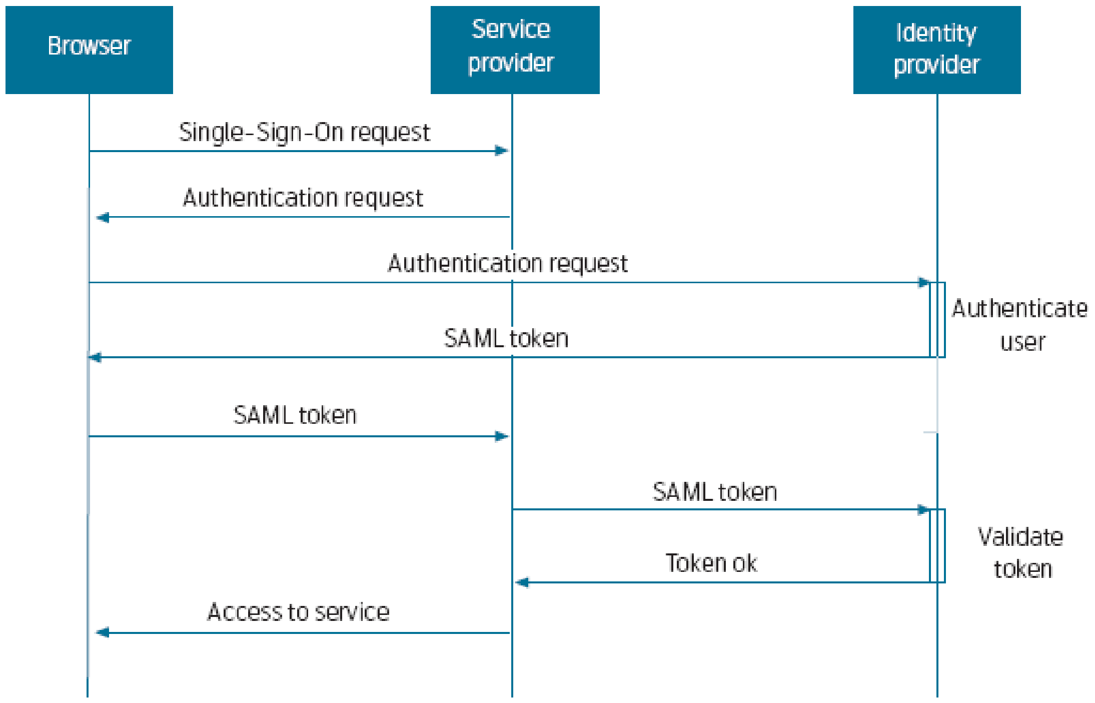


Figure 16. Liberty Alliance SSO messages flow [22]

So, in order to indicate SSO capability and convey SAML contents such as Authn-Request and artifact, we need to design a new protocol feature following the extension service instruction in SIP specification [26]. In other words, the followings must be modified:

- **A new option tag should be added:** It will be added in the Proxy-Require header to designate Liberty Alliance's SSO authentication. This tag is used to expect artifact header in the SIP request and Authn-Request header in the SIP response.

- **SIP headers should be added:** They are Authn-Request header and Artifact header. Authn-Request header contains the SP's authentication request and is added in the SIP response 401 or 407. Artifact header contains the artifact and is added in the SIP requests. It is removed after the artifact resolution once the verification is done.

Since we are considering adding SSO capability to SIP environment only, that means both SAML assertion request and artifact resolution stages are executed in SIP protocol. Therefore, one primary task is to carry SAML protocols with SIP framework. There is an IETF draft on SIP SAML profile and binding [20] has been proposed. However, it relies on HTTP URI in the SAML HTTP-URI-based SIP Binding and hence cannot be applied in this case. Instead, we should use SAML SIP-URI-based Binding (SSUB) and SAML SOAP Binding over SIP (SSSB) to solve the problem. SSUB binding extends the SAML URI Binding specified in the SAML binding standard [21]. The general form of SIP URI is as follows:

sip:user:password@host:port;uri-parameters?headers

Assume that an IDP's SIP address is "sip:idp@ntnu.no", SAML assertion URI uses the parameter "assertion", the assertion is issued for the caller "trungkie", and a request for the assertion can be sent to following SIP address:

sip:idp@ntnu.no;transport=tcp&assertion=id?from=trungkie%40stud.ntnu.no

SSSB binding keeps SAML SOAP binding [21] and uses SIP to transport SOAP messages. Since SAML negotiation may contain a lot of data that often results in large messages to transport, connection-oriented protocol like TCP should be used to transmit. That parameter is specified in the field "transport".

However, in native SIP environment, IDP must provide authentication service on the top of SIP framework. In fact, Identity Provider, also known as Authentication Server, is a special SIP proxy in this case. It provides authentication service to users and is trusted by other proxies. The underlying SSO function and ID-FF profile remains the same, but with a different protocol binding.

2.1.3. Discussion

With the HTTP-Digest authentication supporting capability added in IMS, a user can now login to an IMS of a native SIP domain using that method. In other words, that means a user can login to his IMPU user@ims.com or his native SIP address user@sip.com using HTTP-Digest. Once the user logs in to the system, authentication is completed for SSO-enabled services. In mobile device, it is considered to bind with the credential in SIM card. As a result, SSO will be a bootstrapping service on the user end. Generic Bootstrapping Architecture and Generic Authentication Architecture (GBA/GAA) [23] that was accepted in 3GPP project is also an example. It also provides an interface [24] to cooperate with Liberty Alliance project.
Furthermore, it can be expanded by sharing the account on a group of devices. By taking advantage of the Contact header list in SIP, the initiator could append all trusted devices' SIP addresses into the list. In fact, by using the User Profile Administrating function, each user can add or remove other address/identity that will be federated with one another to achieve SSO capability. Combination of all those will lead to an identity federation framework for a user whether he is in IMS or native SIP domain, and our issues are solved.

2.2. Solution 2: A modified SIP-enabled Client

In this solution, we consider IMS and native SIP domains separately, meaning that we have to face with the same problems as pointed out in the Section III.1.4. These problems will now be consequently considered.

2.2.1. Enable transparent authentication between IMS and native SIP

This part deals with one of the two issues identified in the Section III.1.4: How can a user move between IMS and native SIP domains without having to re-authenticate?

As studied earlier, IMS and native SIP have different authentication methods, namely IMS-AKA based on USIM/ISIM and HTTP-Digest based on challenge/response respectively.

That leads to two cases: When a user already authenticated in IMS domain moves to native SIP domain and vice-versa. Each case has different properties and need to be analyzed separately.

2.2.1.1. From IMS to native SIP

In this case, the user is authenticated in IMS domain using normal method IMS-AKA. We have studied the authentication process of IMS in Section II.3.1.2 but this part will give a more details of what those messages flow contain [27]. Figure 17 refers to the same contents as Figure 7.

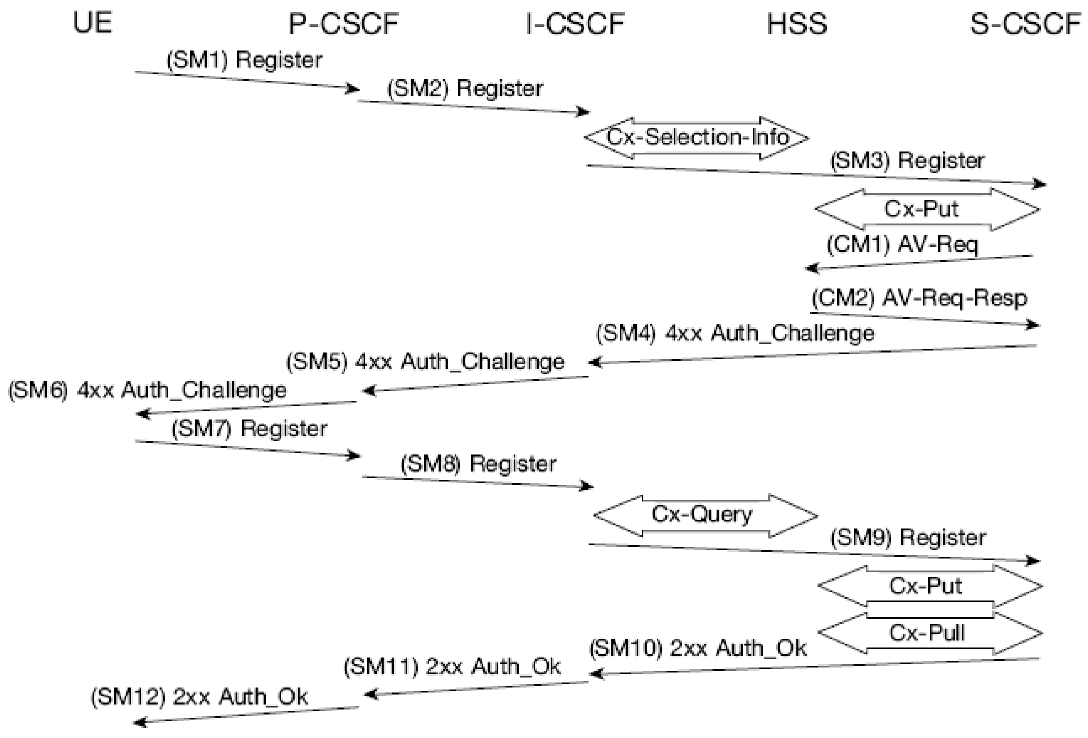


Figure 17. IMS Registration & Authentication

The message flows are the same regardless of whether the user has an IMPU already registered or not.

- In the first message: **SM1: REGISTER(IMPI, IMPU)**
- In SM2 and SM3 the P-CSCF and the I-CSCF respectively forwards the SIP REGISTER towards the SCSCF.
- Upon receiving the SIP REGISTER the S-CSCF shall use an Authentication Vector (AV) for authenticating and agreeing on a key with the user. If the S-CSCF has no valid AV then it shall send a request for AV(s) to the HSS in CM1 together with the number *m* of AVs wanted where m is at least one [27]:
  CM1: Cx-AV-Req(IMPI, m)
- Upon receipt of a request from the S-CSCF, the HSS sends an ordered array of *n* authentication vectors to the S-CSCF using CM2. The authentication vectors are ordered based on sequence number. Each authentication vector consists of the following components: a random number RAND, an expected response XRES, a cipher key CK, an integrity key IK and an authentication token AUTN:
  CM2: Cx-AV-Req-Resp{IMPI,RANDi||AUTNi||XRESi||CKi||IKi}
  (where i runs from 1 to *n*)
- The S-CSCF sends a SIP 4xx Auth_Challenge i.e. an authentication challenge towards the UE including the challenge RAND and the authentication token AUTN in **SM4: 4xx Auth_Challenge(IMPI, RAND, AUTN, IK, CK)**
- When the P-CSCF receives SM5 it shall store the keys and remove that information and forward the rest of the message to the UE:
  SM6: 4xx Auth_Challenge(IMPI, RAND, AUTN)
- The UE receives AUTN, which includes a MAC and the SQN. The UE calculates the XMAC and checks that XMAC=MAC, and then checks if the SQN is in the correct range. If both these checks are successful the UE calculates the response, RES, puts it into the Authorization header and sends it back to the registrar in SM7. The UE at this stage also computes the session keys CK and IK.
  SM7: REGISTER(IMPI, RES)
- Upon receiving SM9 containing the response, the S-CSCF retrieves the active XRES for that user and uses this to check the response sent by the UE. If the check is successful then the user has been authenticated and the IMPU is registered in the S-CSCF.

So, from the authentication procedure above, we can temporarily store the authentication “proofs” somewhere in some databases that can be accessed by both the UE and the servers to use later. When the authenticated user moves to a native SIP domain, those “proofs” will be showed to prove that the user has been authenticated and therefore does not need to re-authenticate in native SIP domain.

To achieve this, we need to deploy the following components:

- **IMS Client:** This client will be installed in the user's mobile phone and directly participate in the IMS authentication process. It will store the authentication parameters every time after the user is authenticated. In our design, it will locally store the RAND, RES and IK parameters but in a hashed way. In other words, it will store the hashed value (by any hash function such as SHA or MD5, etc.) of RES concatenated with IK, i.e. the value to be stored is *the hashed value of (RES|IK)*. The RAND value is stored visibly to everyone. Those parameters will be expired after the call is terminated.

- **Authentication Database:** There will be a database, accessed by both HSS and the S-CSCF. This database will be indexed by the user's IMPI (i.e. the *master key of this database is IMPI*, since each device has only one IMPI). After each user's successful authentication, this database will store the RAND, XRES and IK but also in a hashed way, i.e. the value to be stored *is the hashed value of (XRES|IK)*. The RAND value is stored visibly to everyone.

As we can see from the message flows, after each successful authentication, the RES value stored in IMS Client and the XRES value stored in the database should be the same, and so does the IK value. When a user moves to native SIP domain, his mobile phone will detect new environment and trigger the IMS client to send to the SIP server the stored hashed value, i.e. the hashed value of (RES|IK), along with the RAND parameter. After receiving that, instead of asking the user to authenticate as normal, the SIP server should access and compare that value with the hashed value of (XRES|IK) and RAND stored in the Authentication Database. If they match, then it means that the user has been authenticated and no need to ask him for authenticate again. Figure 18 illustrates the idea.

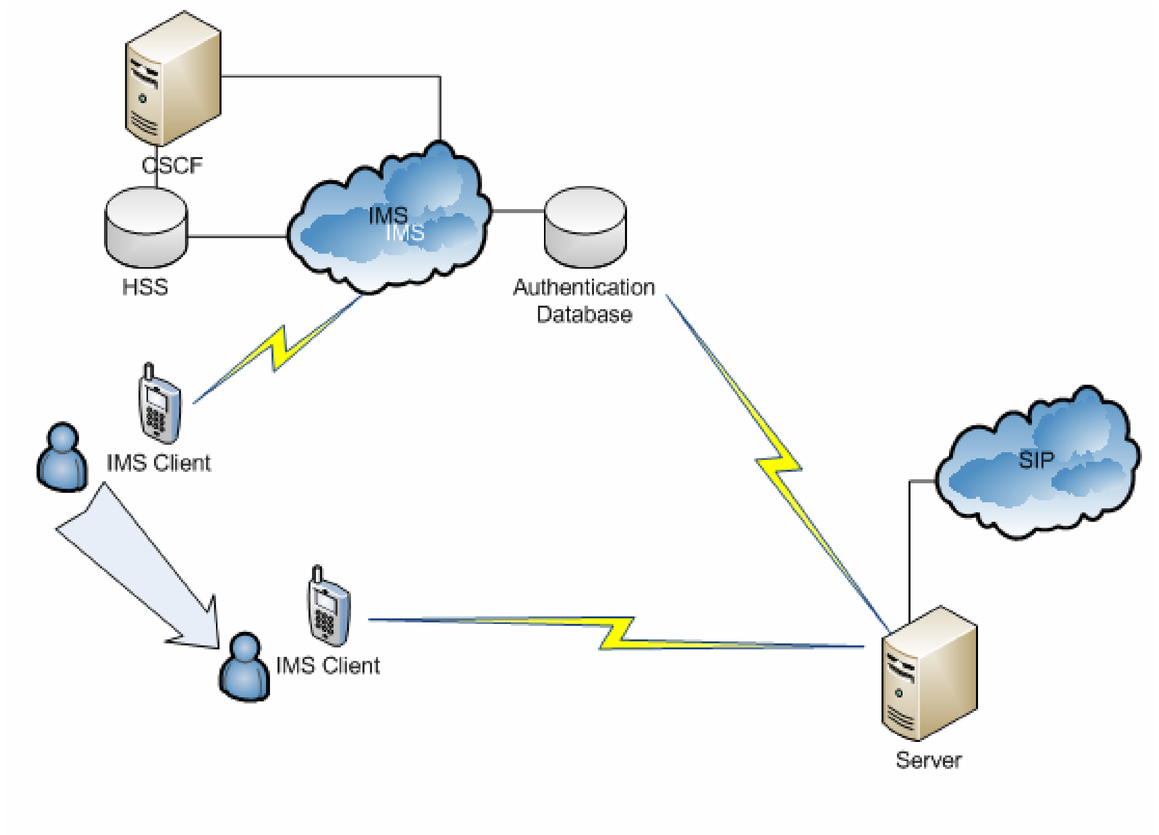


Figure 18. Moving from IMS to native SIP

It is worth noting that by storing the hashed value of RES (XRES) concatenated with the integrity key IK, we prevent the lost of sensitive data. This information is useless for intruders and will be deleted after the call is terminated. There maybe left one primary concern, which is how the SIP server know and access the Authentication Database. This can be done by using another server with the role of Identity Provider (IDP) as specified in Liberty Alliance scheme.

2.2.1.2. From native SIP to IMS

As in the previous part, we will first look at the SIP authentication mechanism to see what information the messages are exchanging. The authentication process takes place between the User Agent (UA) and the server (proxy, registrar, user agent server). The server requires the UA to authenticate itself before processing the request. A user agent can also request authentication of a server.

SIP uses headers for authentication. The WWW-Authenticate header is used in 401 response message sent by the server and the Proxy-Authenticate header is used in 407 response message sent by the proxy. In response to those headers, a user agent should send an Authorization and Proxy-Authorization header respectively containing the credentials in the next request.

Generally, the SIP authentication mechanism is illustrated in Figure 19 below [28].

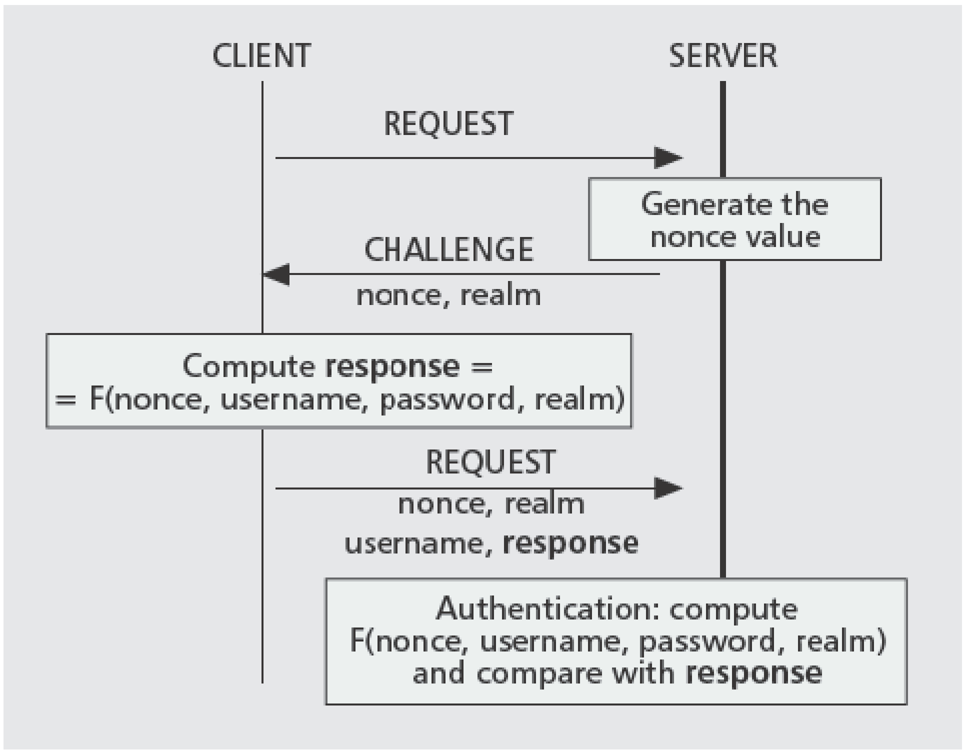

CLIENT
SERVER
REQUEST
Generate the nonce value
CHALLENGE
nonce, realm
Compute **response** = = F(nonce, username, password, realm)
REQUEST
nonce, realm
username, **response**
Authentication: compute F(nonce, username, password, realm) and compare with **response**

Figure 19. HTTP-Digest Authentication in SIP

Comparing with the IMS case above, we can see that the *nonce* value in SIP is similar to the RAND value in IMS, the password in SIP is similar to IK in IMS and the username in SIP is similar to the IMPI in IMS. That leads to the same idea as in the previous case, i.e. we will have:

- SIP Client storing the hashed value of (username|realm|response) and the nonce value.
- Authentication Database storing the hashed value of the username|realm concatenated with its expected response (different from the response sent by the user agent) and the nonce value.

When moving to IMS domain, the CSCF servers will receive the stored information from the SIP client and compare with those stored in the Authentication Database. If they are match, the user does not have to re-authenticate.

It is worth noting that we include the *realm* value in the hash function since the same username with the same nonce but different realm (i.e. different SIP proxies) can lead to different response values.

2.2.2. Enable transparent call-redirecting

The movement between different domains (i.e. IMS and native SIP) means the user has to switch between his IMS and native SIP addresses (i.e. user@ims.com and user@sip.com as mentioned earlier). The first part of this solution (Part 2.2.1) allows the user to use his IMS and native SIP address alternatively in appropriate domains without re-authentication. However, there is also another problem as in the following example. When a user with a SIP address user1@domain.com wants to call to another user having two addresses user2@ims.com and user2@sip.com, everything seems to be normal if the second user stays in the same domain in which he started the conversation (i.e. IMS or native SIP). In this example, we assume that user1@domain.com is talking with the second user in IMS domain using his user2@ims.com address. But in fact, the second user can move from IMS domain to native SIP domain during the conversation and he will have to use his native SIP user2@sip.com address. The transparent authentication triggers, and the switch proceeds smoothly. But the first user will then notify that he is talking with a user having the address user2@sip.com instead of user2@ims.com (the one he intended to talk to). There will not be problem if the first user knows that both the addresses are of the same person but in fact, one can have an IMS address something like trungkie@ims.com and a totally different native SIP address like NTNU_student_220181@sip.com. The issue is how the first user can be certain that he is talking to the same person or not.

The same problem arises in the case that a user (user1@domain.com) calls to another one through the address user2@sip.com. But the second user, at that time, is in an IMS domain and using his user2@ims.com address. He has all the calls to his native SIP user2@sip.com redirected to his IMS address. When the first user receives reply from user2@ims.com instead of user2@sip.com, he could not be sure that he speaks to the right person.

Therefore, this section will propose a solution that can be implemented in a SIP client application. When a user changes his address, the SIP client of the caller side will be triggered and replaces the new address by the called address. In other words, when user1@domain.com calls user2@sip.com, he will then receive a reply from user2@ims.com. The SIP client will replace the address so that the caller will see a reply from user2@sip.com instead of user2@sip.com.

It is also worth mentioning that the redirection can be done by a Redirect Server using 3xx messages with interaction with Proxy Servers.

2.2.3. Discussion

With a SIP-enabled client application installed in his mobile, a user can now have incoming calls redirected between his different addresses and still let the callers know he is the one that they want to call. Of course that SIP-enabled client has to support both the features as discussed above. This solution can be implemented as a Java-based SIP client, since most of the modern mobile devices now support Java.

IV. Modified SIP-enabled Client Solution Implementation

Due to the huge amount of work required by the Solution 1, we choose the second solution for implementation (i.e. the modified SIP-enabled Client solution). We will develop a Java-based SIP client that can be run in both PC and mobile (of course, since the main purpose is to run that SIP client on mobile devices). To make it works in real conditions, a real SIP server (including Proxy Server) will also be deployed. To have our SIP client connected to the SIP server, there should be a Registrar to let the devices carry out the registration. All the implemented cases and steps will be discussed in details in the following parts.

1. Implementation Cases

1.1. SIP Server

There are a lot of SIP server applications on the Internet, and selecting one to be implemented along with our SIP client is not an easy task. After carefully considering, we decided to choose BEA Weblogic® SIP Server. It is the industry's premier open, SIP-based, converged Java EE-IMS-SOA application server designed specifically for large-scale IMS-NGN deployments. BEA WebLogic SIP Server delivers the most powerful highly available service convergence platform for network operators, network equipment providers (NEPs), systems integrators that seek to develop, integrate, and operate revenue-generating, real-time multimedia communication services [29].

For more understanding about this SIP Server and how it can interact with our SIP client, we will take a quick look at its technical components.

1.1.1. BEA Weblogic® SIP Server Overview

BEA WebLogic SIP Server is the industry's first converged Java EE-SIP-IMS-SOA application server, delivering unparalleled high availability, reliability, scalability, and performance. It is the IMS-SIP application server component of the BEA WebLogic Communications Platform family of network middleware. A robust global ecosystem of partners have standardized on BEA WebLogic SIP Server as the IMS-SIP foundation for next-generation communications. BEA WebLogic SIP Server delivers a converged application container, offering integrated support for Java, Web services, and IMS standards. It is compliant with the SIP Servlet API (JSR 116) specification, the Java community standard and programming model for developing and executing SIP-based applications. BEA WebLogic SIP Server also supports 3GPP IMS standards such as *Sh* for real-time access to subscriber data, *Ro* for online charging event, and *Rf* for offline charging events. And through tight integration with BEA WebLogic Server, BEA WebLogic SIP Server provides native access to Java EE and Java SE services including JDBC, JMS, JNDI, RMI/IIOP, EJB, Java Web Services, and IPv6. As network operators worldwide deploy SIP-based NGN architectures such as IMS and fixed-mobile convergence (FMC) networks, new revenue-enhancing services must be created and deployed rapidly and cost-effectively [29].

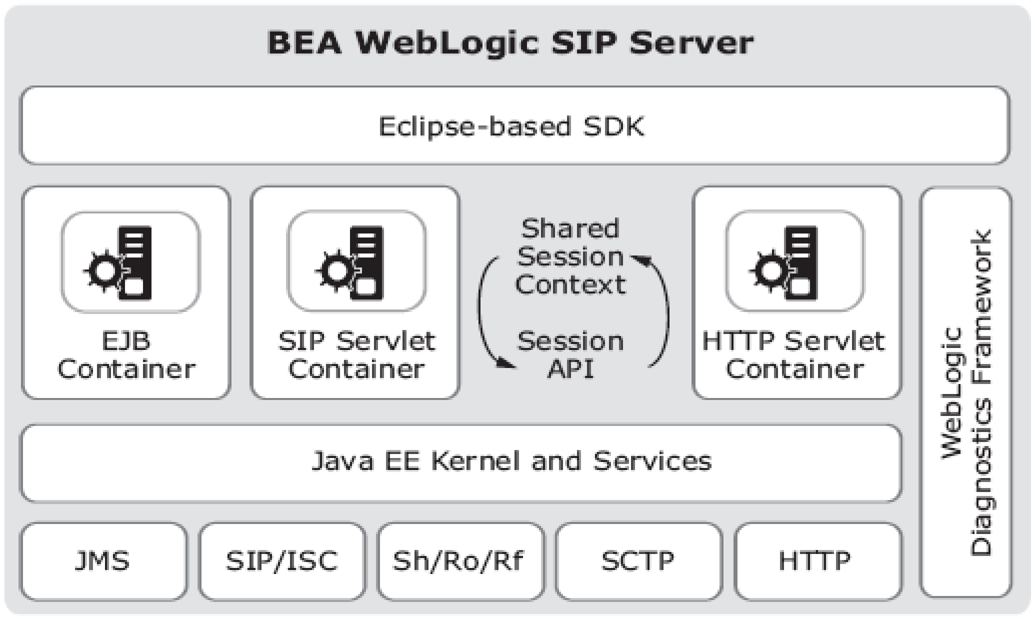


Figure 20. Functional Overview of BEA Weblogic SIP Server [29]

Services for IMS and FMC networks are deployed at a new layer above the core network, commonly referred to as the Services Layer. At the core foundation of the NGN Services Layer is the SIP-IMS application server. BEA WebLogic SIP Server is a Java EE-SIP-IMS application server for the IMSNGN Services Layer. As a fully IMS-compliant Java EE-SIP application server, BEA WebLogic SIP Server supports all key interfaces for an IMS application server, including ISC, Ro, Rf, and Sh, as well as the general SIP Profile mandated by the 3GPP Release 6 specifications. This allows services deployed on BEA WebLogic SIP Server to function as Charging Trigger Functions (CTF), issuing charging events to Online Charging Function (OCF) and Charging Data Function (CDF). It also allows for seamless integration with Home Subscriber Server (HSS), Call Session Control Function (CSCF), and Media Resource Function (MRF) components of IMS-based networks. BEA WebLogic SIP Server has successfully completed interoperability testing with IMS core network components from many of the world’s leading NEPs.

1.1.2. BEA Weblogic® SIP Server Communication Platform

As mentioned above, this SIP Server strongly support Java community, hence our Java-based SIP client will also be compatible. BEA WebLogic SIP Server tightly integrates its operations, administration, and management framework with BEA WebLogic Server®, the underlying Java EE platform.

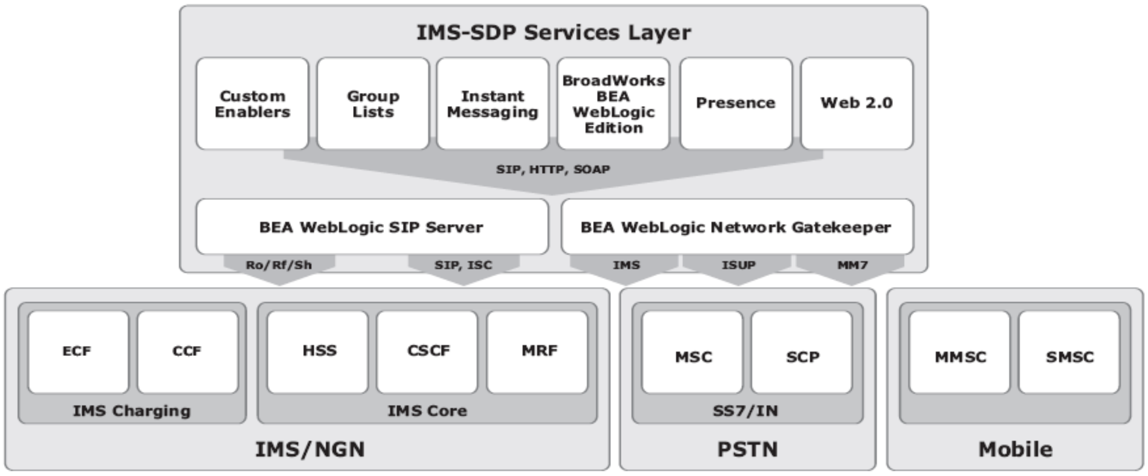
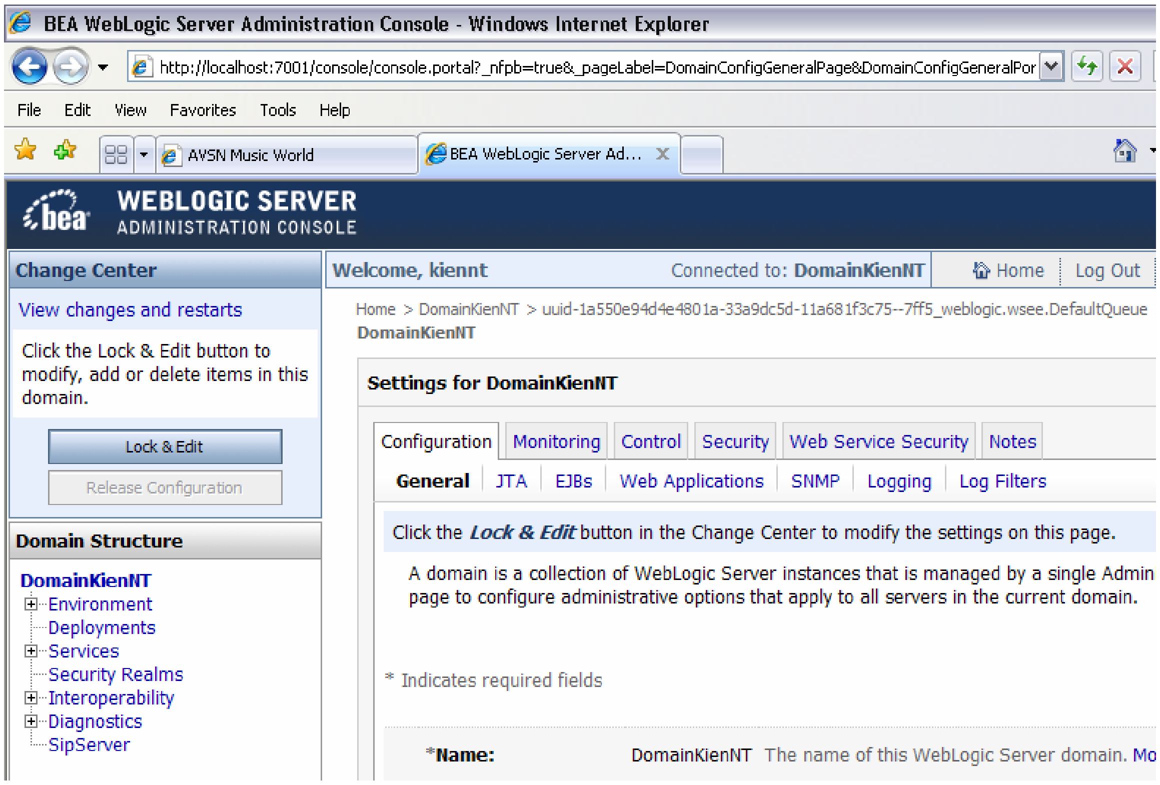


Figure 21. Communications Platform

The core architecture of BEA WebLogic SIP Server is the implementation of a single converged application container, based on a comprehensive collection of key Java, Internet, Web services, and IMS industry standards. The breadth and depth of supported standards provides a feature-rich environment for creating innovative converged Internet-IMS services. As the specification lead for SIP Servlet API 1.1 (JSR 289) within the Java Community Process (JCP), BEA is also committed to driving the Java community standard for converged Java EE-SIP application servers. BEA WebLogic SIP Server is the SIP-IMS application server component of the BEA WebLogic Communications Platform product family. With the BEA WebLogic® Communications Platform product family, BEA delivers the industry's most comprehensive portfolio of telecom middleware products for customers implementing a SIP-, Java-, and telecom Web services-based service layers for service delivery platforms and IMS/NGN networks.

1.1.3. Implementation Results

After installing and configuring, we created a domain named *DomainKienNT* which will be used to host the Registrar Server later on. Figure 22 shows the overview of the administration console of that domain, which enable us to configure the whole server-side things during the implementation phase.



Click the Lock & Edit button to modify, add or delete items in this domain.

Click the ***Lock & Edit*** button in the Change Center to modify the settings on this page.

A domain is a collection of WebLogic Server instances that is managed by a single Admin page to configure administrative options that apply to all servers in the current domain.

* Indicates required fields

***Name:** DomainKienNT The name of this WebLogic Server domain. Mo

Figure 22. Domain Setting

1.2. Registrar Server

This implementation demonstrates registrar and location service functionality. It also demonstrates the Globally Routable User Agent URIs (GRUU) specifications as described in draft-ietf-sip-gruu-11 and Session-Timer handling by proxies as described in RFC 4028.

1.2.1. Registrar Server Components

The implemented Registrar Sever has the following components:

- **Registrar:** Handles REGISTER requests and communicates the bindings to and from the LocationService.

- **LocationService:** A LocationService is exposed as an interface for application-scoped classes, and as a java.util.Map to components outside of the application. The LocationService has two different implementations to store bindings in memory and in a database.

- **DomainManager**: Keeps track of domains managed by the Registrar. The DomainManager also has methods to resolve incoming GRUU to contact addresses.

1.2.2. Building the Registrar Server

The Registrar Server is an open-source application which handles REGISTER requests and interacts with Location Service. However, it does not handle SIP MESSAGE messages. This will be a problem since we need all SIP messages to be handled in order to allow our SIP client to run comfortably. Hence there are some more tasks to be done here:

1.2.2.1. Modifying the XML-based source code

As we want this registrar server to handle SIP messages, we need to add the following XML tags into the source file.

```
<servlet-mapping>
  <servlet-name>registrar</servlet-name>
  <pattern>
    <equal>
   <var>request.method</var>
   <value>REGISTER</value>
    </equal>
  </pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>proxy</servlet-name>
  <pattern>
    <equal>
      <var>request.method</var>
      <value>INVITE</value>
    </equal>
  </pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>proxy</servlet-name>
  <pattern>
    <equal>
      <var>request.method</var>
      <value>UPDATE</value>
    </equal>
  </pattern>
</servlet-mapping>

<servlet-mapping>
  <servlet-name>proxy</servlet-name>
  <pattern>
    <equal>
      <var>request.method</var>
      <value>SUBSCRIBE</value>
    </equal>
    </pattern>
  </servlet-mapping>

  <servlet-mapping>
    <servlet-name>proxy</servlet-name>
    <pattern>
      <equal>
        <var>request.method</var>
        <value>NOTIFY</value>
      </equal>
    </pattern>
  </servlet-mapping>

 <servlet-mapping>
    <servlet-name>proxy</servlet-name>
    <pattern>
      <equal>
     <var>request.method</var>
     <value>MESSAGE</value>
      </equal>
    </pattern>
 </servlet-mapping>
```

As you can see, all basic SIP messages like REGISTER, MESSAGE, INVITE, etc. are now added.

1.2.2.2. Rebuilding the Application

After some minor tasks like setting the environment variables, we use Ant [30] to re-build the application by using the command:

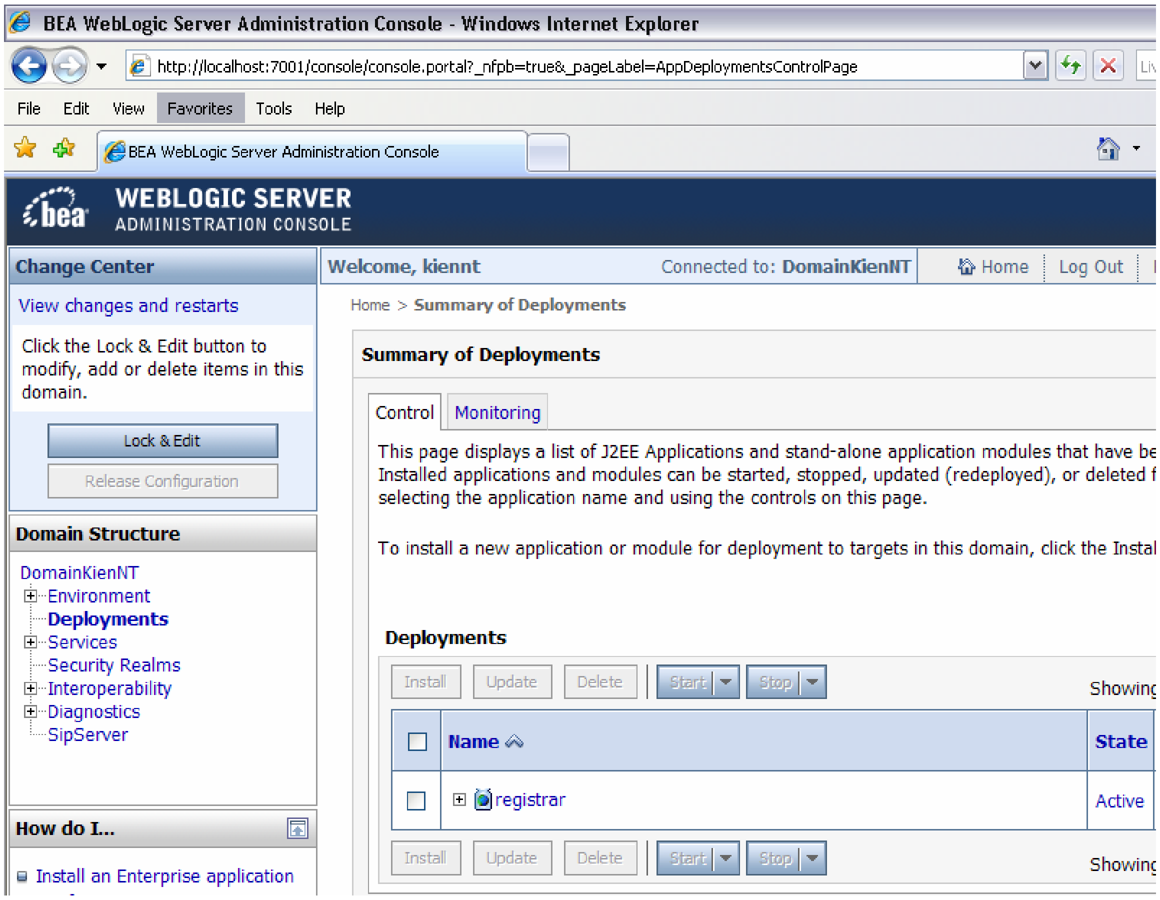
```
ant clean build
```

The next task will be adding this application to the domain *DomainKienNT* which was created before. After that, the final task is to deploy the Registrar Server within the specified domain using the command:

```
ant deploy
```

1.2.3. Implementation Results

The last server-side implementation task was completed. We have an active Registrar server running in *DomainKienNT* domain like below:


BEA WebLogic Server Administration Console - Windows Internet Explorer
http://localhost:7001/console/console.portal?_nfpb=true&_pageLabel=AppDeploymentsControlPage
File Edit View Favorites Tools Help
BEA WebLogic Server Administration Console
WEBLOGIC SERVER
ADMINISTRATION CONSOLE
Change Center
View changes and restarts
Lock & Edit
Release Configuration
Domain Structure
DomainKienNT
Environment
Deployments
Services
Security Realms
Interoperability
Diagnostics
SipServer
How do I...
Install an Enterprise application
Welcome, kiennt
Connected to: DomainKienNT
Home
Log Out
Home > Summary of Deployments
Summary of Deployments
Control
Monitoring
Deployments
Install Update Delete Start Stop
Showing
Name
State
registrar
Active

Click the Lock & Edit button to modify, add or delete items in this domain.

This page displays a list of J2EE Applications and stand-alone application modules that have be
Installed applications and modules can be started, stopped, updated (redeployed), or deleted f
selecting the application name and using the controls on this page.

To install a new application or module for deployment to targets in this domain, click the Instal

	Name	State
☐	registrar	Active

Figure 23. Registrar Server

1.3. SIP Client based on Proposed Solution

This is the most important part in our solution. As SIP servers, there are many SIP client applications available on the Internet such as X-Lite, SJ-Phone, MiniSIP, etc. but all of them are not suited to our proposed solution. Therefore, we will develop a SIP client which allows us to do the simplest task: Sending text messages between the SIP clients. Besides, the new feature as mentioned in the second solution (i.e. enabling transparent authentication and call-redirecting) should be supported as well. The following steps will discuss about the developing process in more details.

1.3.1. Prerequisites

We used Eclipse [31] with EclipseME [32] installed to work as a bridge between Eclipse and Sun Java Wireless Toolkit [33]. EclipseME is an Eclipse plug-in to help develop J2ME MIDlets. It does the "grunt work" of connecting Wireless Toolkits to the Eclipse development environment, allowing us to focus on developing application, rather than worrying about the special needs of J2ME development [32]. Besides, The Sun Java Wireless Toolkit for CLDC (formerly known as Java 2 Platform, Micro Edition (J2ME)

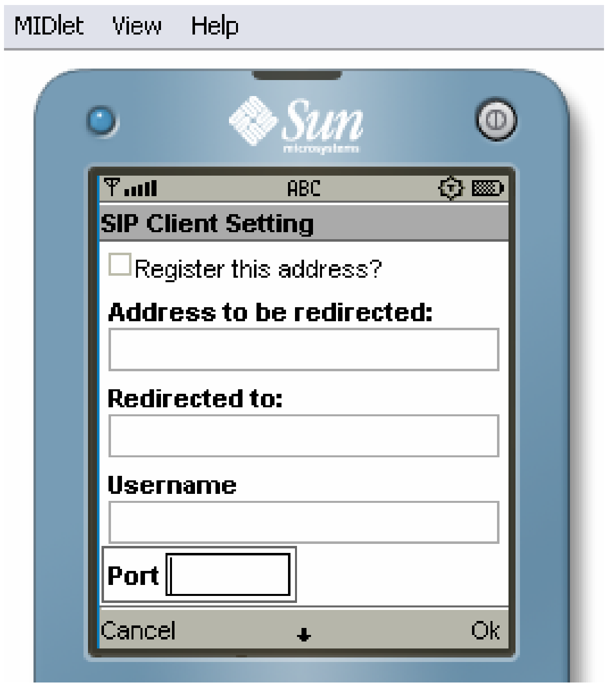
Wireless Toolkit) is a state-of-the-art toolbox for developing wireless applications that are based on Java ME's Connected Limited Device Configuration (CLDC) and Mobile Information Device Profile (MIDP), and designed to run on cell phones, mainstream personal digital assistants, and other small mobile devices. The toolkit includes the emulation environments, performance optimization and tuning features, documentation, and examples that developers need to bring efficient and successful wireless applications to market quickly [33]. The idea of using all those prerequisites to program SIP application based on JavaME is provided in [34] by Emmanuel Proux. The Sun Java Wireless Toolkit also provides us four (04) interfaces to choose when simulate the mobile device. The next part will discuss about the design of the SIP client's components that are suitable with those use cases analyzed in the part III.1.3.

1.3.2. SIP Client-Designing Use Cases

In accordance with the use cases analyzed in the part III.1.3, our SIP Client should have the similar use cases. As mentioned earlier, this SIP Client will allow us to send plain text messages through SIP protocol and have the features which were discussed in part III.2.2. We will present the interface of the SIP Client with the correspondent use case and then move to the detailed components of the application.

1.3.2.1. Registering

The Register setting interface of the SIP client is shown in the Figure below.



SIP Client Setting

☐ Register this address?

Address to be redirected:

Redirected to:

Username

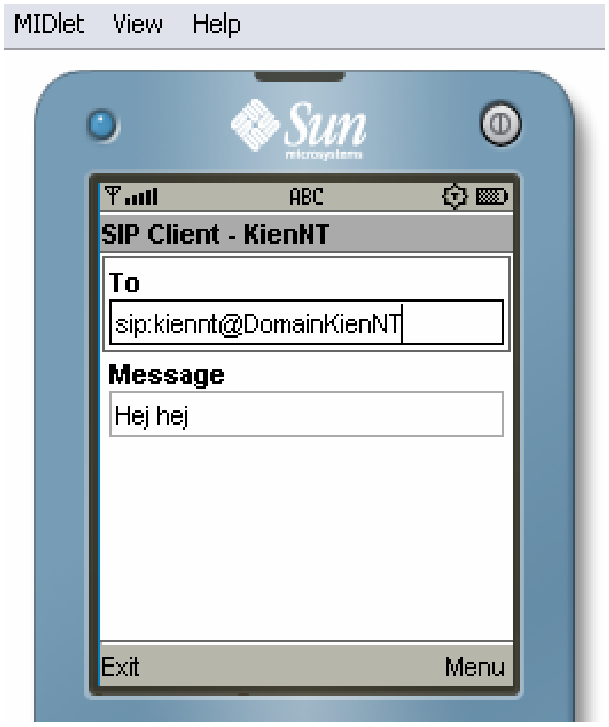
Port

Figure 24. Registering Setting Interface

When a user wants to register a new address that will bind to his current mobile device, he will use this menu to fill in the necessary information and the SIP Client will then connect to the Registrar server to send REGISTER message.

1.3.2.2. Sending and Receiving Messages

When a user wants to send a message, he enters the address which his message will be sent to along with the message in an interface like below:



To

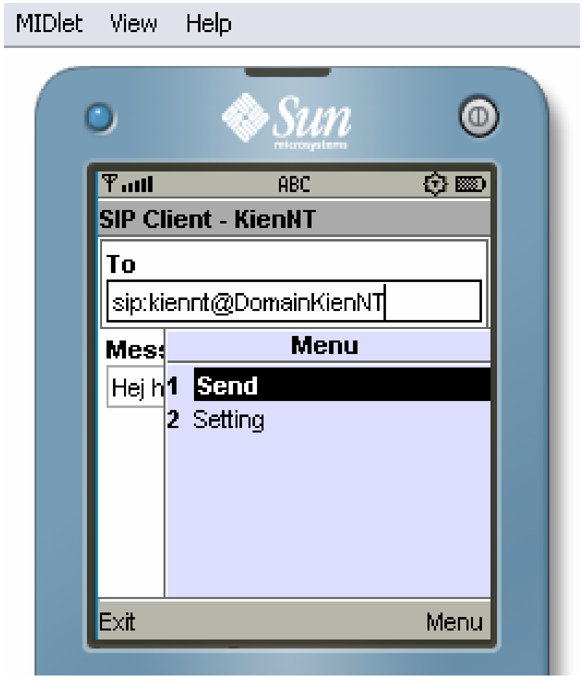
sip:kiennt@DomainKienNT

Message

Hej hej

Figure 25. Sending/Receiving Interface

After that, to send the message, the user will press the button corresponding to the *Menu* one and the new interface will allow him to choose *Send* command as in the Figure 26.



To

sip:kiennt@DomainKienNT

Mes

Hej h

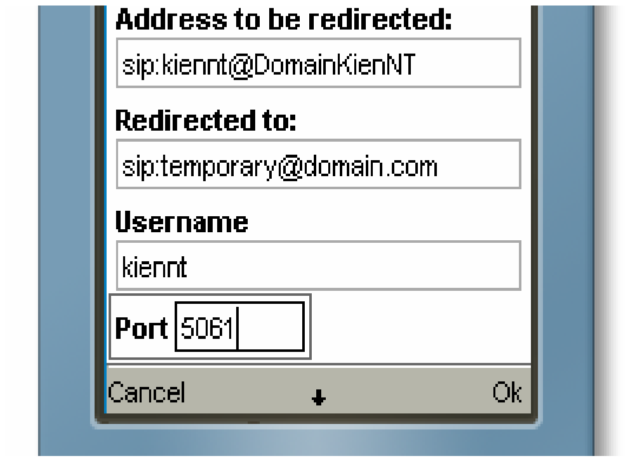
Figure 26. Send command Interface

The received message will appear on the same screen/interface as the sending interface.

1.3.2.3. User Profile Administrating

Within the same menu as when the user registers his address, he can also specify his personal profile such as whether he wants his address to be redirected to another one or not, which port he wants to use, etc.

The picture below will show the correspondent functions:



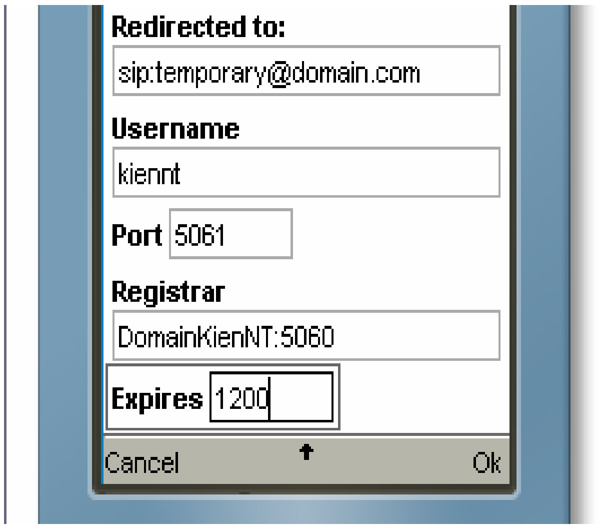
Address to be redirected:
sip:kiennt@DomainKienNT
Redirected to:
sip:temporary@domain.com
Username
kiennt
Port 5061

Figure 27. Profile Administrating Interface

In the above example, the user can set the following parameters:

- **Username:** The username that will be part of his SIP address
- **Address to be redirected:** To tell the server which address he wants to be redirected
- **Redirected to**: The address to which he wants to receive calls, can be a totally different address from the above one.
- **Port:** The port which need to be specified to avoid conflicts

Due to the limited space of the mobile device, we need to scroll down to see more fields that user can set. The rest of the interface is shown below:


Redirected to:
sip:temporary@domain.com
Username
kiennt
Port 5061
Registrar
DomainKienNT:5060
Expires 1200

And the user can specify the Registrar server's domain name as well as the port. The expiry time is also changeable.

1.3.2.4. Authenticating

The authentication of the SIP Client is done at mobile device level (HTTP-Digest or ISIM-based authentication). Storing authenticating information within the client's temporary database to support transparent authentication is just a coding work.

1.3.2.5. Call-Redirecting

When the user specified those addresses to indicate the redirecting and redirected destinations, our SIP client can do it with the help of the server. It can also replace the address being redirected to by the old one so that the user does not know he is talking to another address, hence transparent call-redirecting is achieved. This will be discussed in more details in the later section.

1.3.3. SIP Client Designing Processes

As we used SIP API for Java ME to develop this SIP client, we need to list here some important classes that were used in the application [35]:

- **SipClientConnection:** This class is used to send normal SIP messages that are not recurrent such as INVITE and MESSAGE messages.

- **SipServerConnection:** This class reads incoming messages

- **SipClientConnectionListener:** This will be implemented by classes processing SIP responses.

- **SipServerConnectionListener:** This will be implemented by classes receiving SIP requests.

- **Connector:** This is used to create all kinds of connection objects. For SIP connections, simply use an address that begins with "sip:" and the it creates either SipClientConnection or SipServerConnection objects.

- **SipRefreshHelper:** This class is used to manage recurring outgoing SIP messages such as REGISTER and SUBSCRIBE messages

1.3.3.1. Processes Executed When Sending a Request

The process of sending a message consists of two main phases. The first one is preparing and sending the message. The second one is the processing of the response.

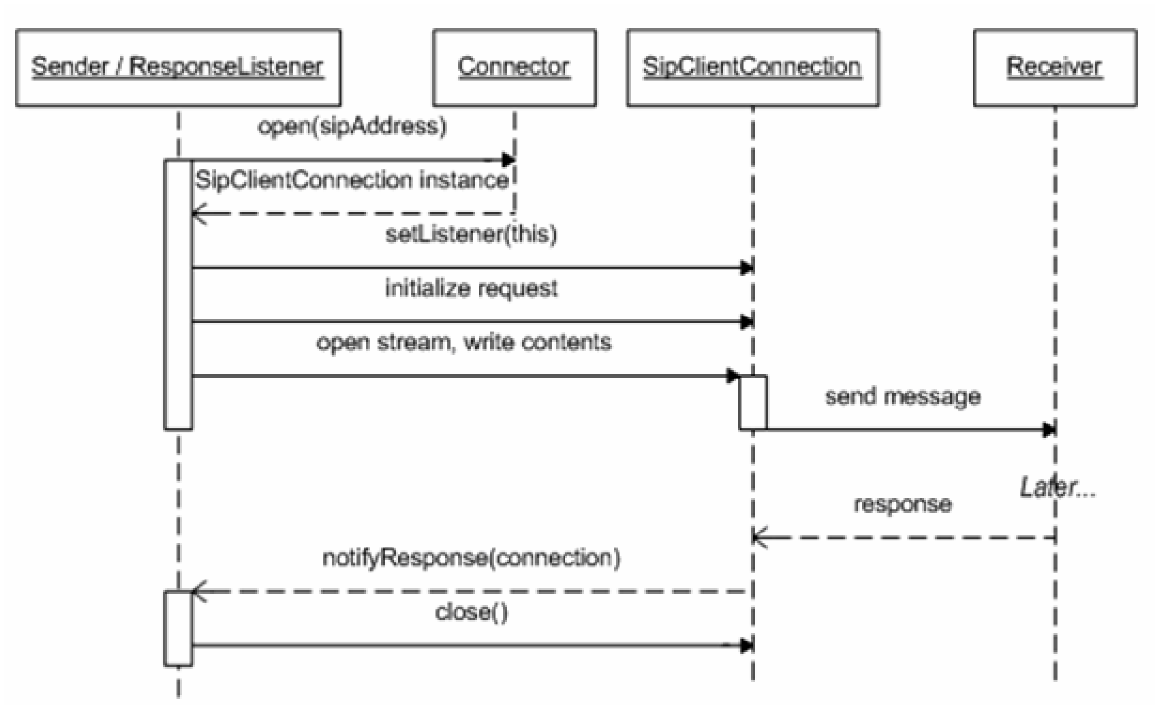


Figure 28. Sending Request Phases [34]

To do this in the SIP client, it would open a client connection and use this as the listener of responses. After that, the client initializes the request type and sets some mandatory headers. Most of the needed SIP headers for the request will be automatically populated with default values. It will then open an output stream, write the message to this stream and close the stream. But at this time the client does not close the connection since it have to wait for a response.

After a response to be processed using the method named notifyResponse() in the SipClientConnectionListener class which is invoked automatically when a response arrives, the SIP client closes the connection.

1.3.3.2. *Processes Executed When Receiving a Request*

As with sending a request, there are two phases for receiving a request. The first one is to register a listener of incoming messages on a server connection. The second one is being notified of the incoming request and sending back a response. The idea is to open a permanent server connection and get notified when a message arrives asynchronously.

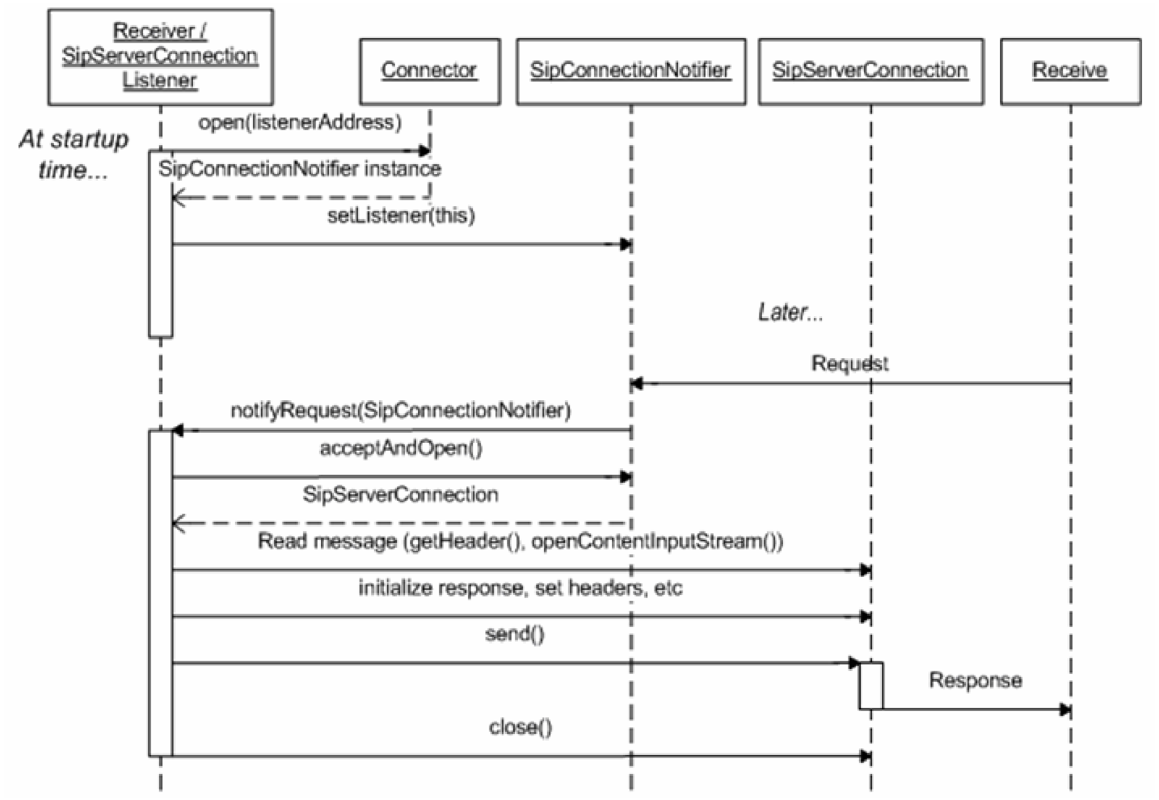

Receiver / SipServerConnection Listener
Connector
SipConnectionNotifier
SipServerConnection
Receive
At startup time...
open(listenerAddress)
SipConnectionNotifier instance
setListener(this)
Later...
Request
notifyRequest(SipConnectionNotifier)
acceptAndOpen()
SipServerConnection
Read message (getHeader(), openContentInputStream())
initialize response, set headers, etc
send()
Response
close()

Figure 29. Receiving Request Phases [34]

1.3.3.3. *Processes Executed When Sending a REGISTER Request*

REGISTER requests are recurrent messages, which are meant to be sent at regular intervals. This task could be done with our SIP client by using the refresh mechanism included in the SIP API.

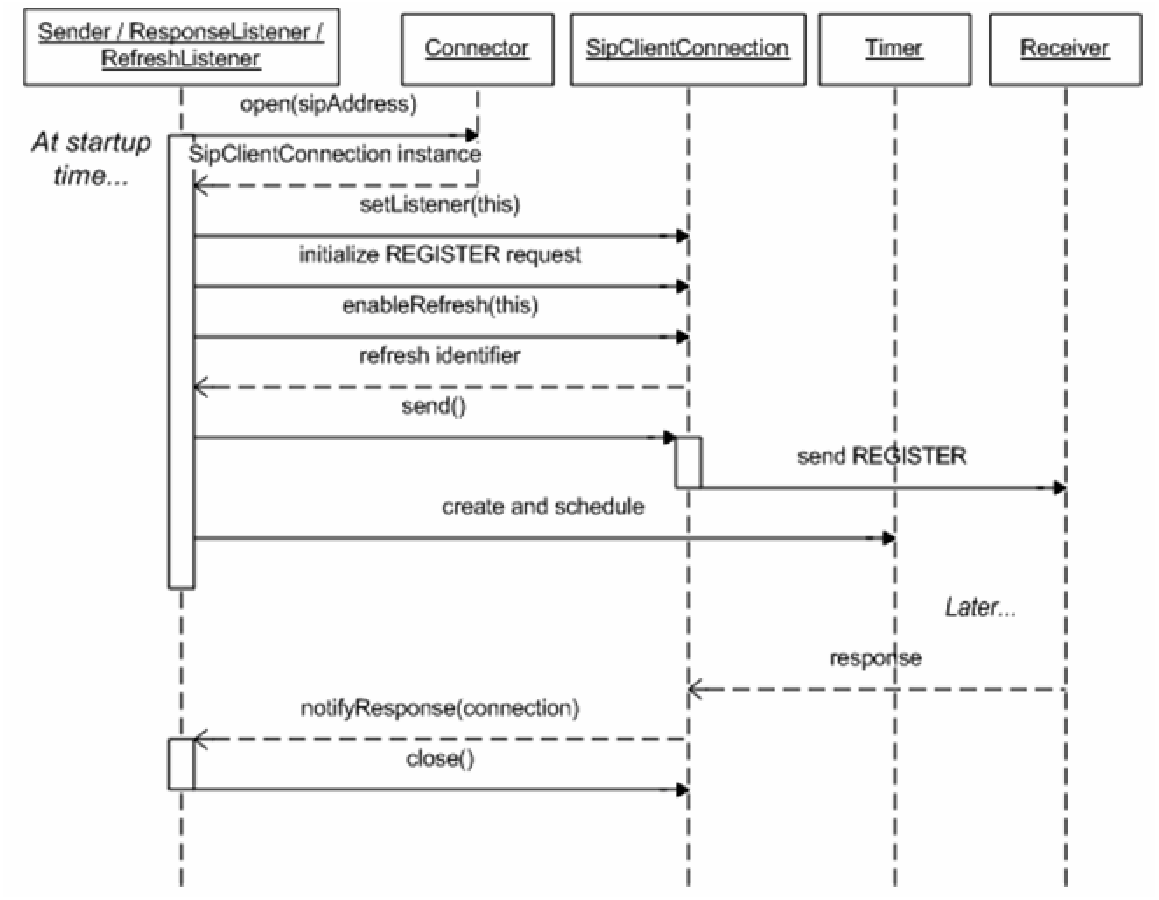


Figure 30. Sending REGISTER Request Phases (at the first time)

The SipClientConnection.enableRefresh() method is used to turn on automatic refreshing of requests and to specify a listener for refresh events. After the first registration, we have the following phases as in the Figure 31.

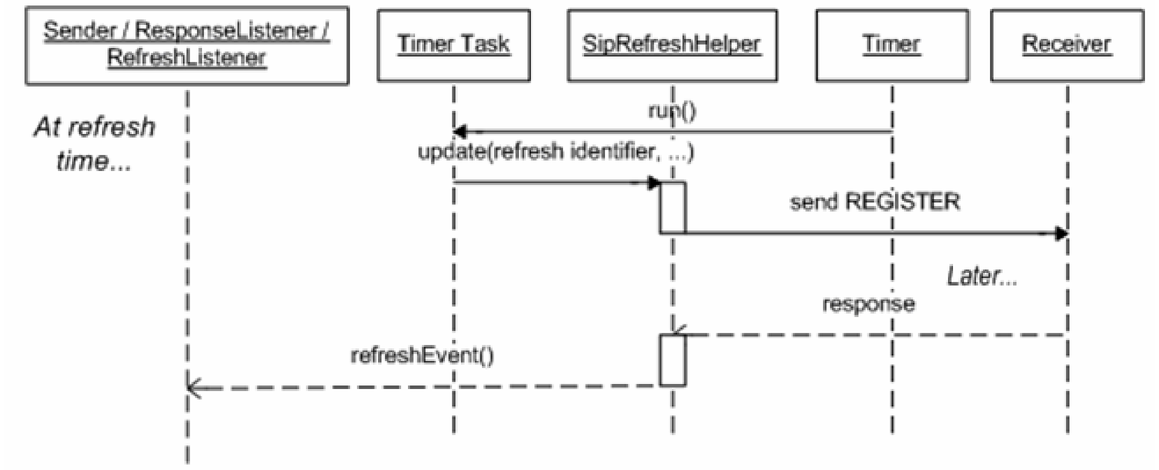


Figure 31. Sending REGISTER Request Phases (after the first time)

So, the three most important tasks in our SIP client are identified and analyzed in terms of phases and processes. In the next part, we will see how it works in each separate demonstration case.

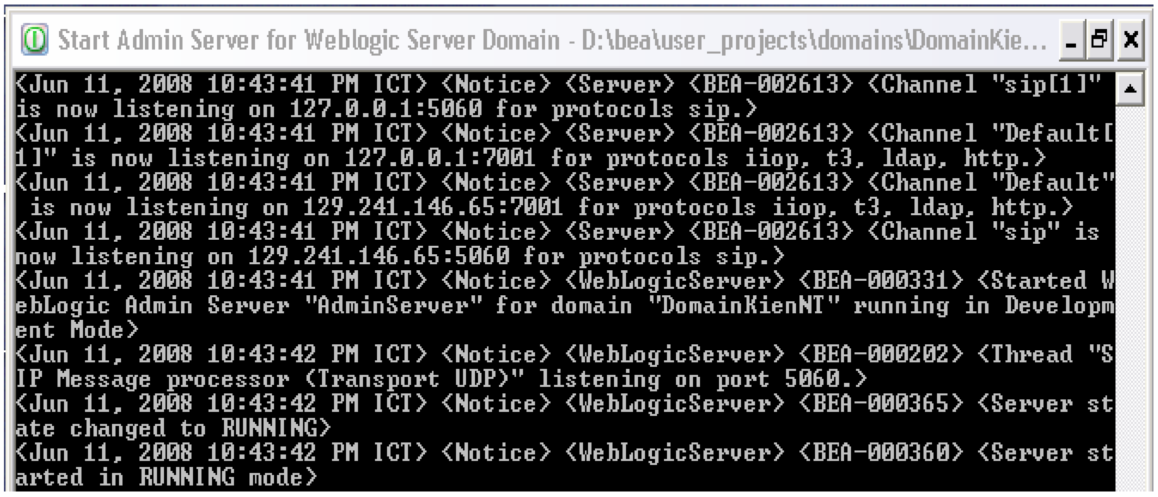
2. Demonstrations

In this part, we will see how our SIP client works in real SIP environment. But more importantly, we also will also examine the supported transparent features discussed in the second solution. We will demonstrate this through some scenarios.

2.1. Preparation

2.1.1. Starting the Servers

Before having everything working, we need to start the SIP/Registrar servers. Our servers will then listen on port 5060 after they move into the running state as illustrated in the Figure 32:


Start Admin Server for Weblogic Server Domain - D:\bea\user_projects\domains\DomainKie...

```
<Jun 11, 2008 10:43:41 PM ICT> <Notice> <Server> <BEA-002613> <Channel "sip[1]"
is now listening on 127.0.0.1:5060 for protocols sip.>
<Jun 11, 2008 10:43:41 PM ICT> <Notice> <Server> <BEA-002613> <Channel "Default[
1]" is now listening on 127.0.0.1:7001 for protocols iiop, t3, ldap, http.>
<Jun 11, 2008 10:43:41 PM ICT> <Notice> <Server> <BEA-002613> <Channel "Default"
 is now listening on 129.241.146.65:7001 for protocols iiop, t3, ldap, http.>
<Jun 11, 2008 10:43:41 PM ICT> <Notice> <Server> <BEA-002613> <Channel "sip" is
now listening on 129.241.146.65:5060 for protocols sip.>
<Jun 11, 2008 10:43:41 PM ICT> <Notice> <WebLogicServer> <BEA-000331> <Started W
ebLogic Admin Server "AdminServer" for domain "DomainKienNT" running in Developm
ent Mode>
<Jun 11, 2008 10:43:42 PM ICT> <Notice> <WebLogicServer> <BEA-000202> <Thread "S
IP Message processor (Transport UDP)" listening on port 5060.>
<Jun 11, 2008 10:43:42 PM ICT> <Notice> <WebLogicServer> <BEA-000365> <Server st
ate changed to RUNNING>
<Jun 11, 2008 10:43:42 PM ICT> <Notice> <WebLogicServer> <BEA-000360> <Server st
arted in RUNNING mode>
```

Figure 32. Running SIP servers

After this step, we will have Registrar server running in the *DomainKienNT* domain and listening on port 5060.

2.1.2. Starting and Binding the Clients

To see how the SIP client works, we need at least two clients running on two different mobile devices. Each mobile device will bind with a specific SIP-enabled address. Since the SUN Java Wireless Toolkit provides us some device interfaces to choose, we will choose two different interfaces for avoiding confusions.

The first device will bind with the SIP address **sip:kiennt1@DomainKienNT**. The registering/binding settings are shown in the picture below:

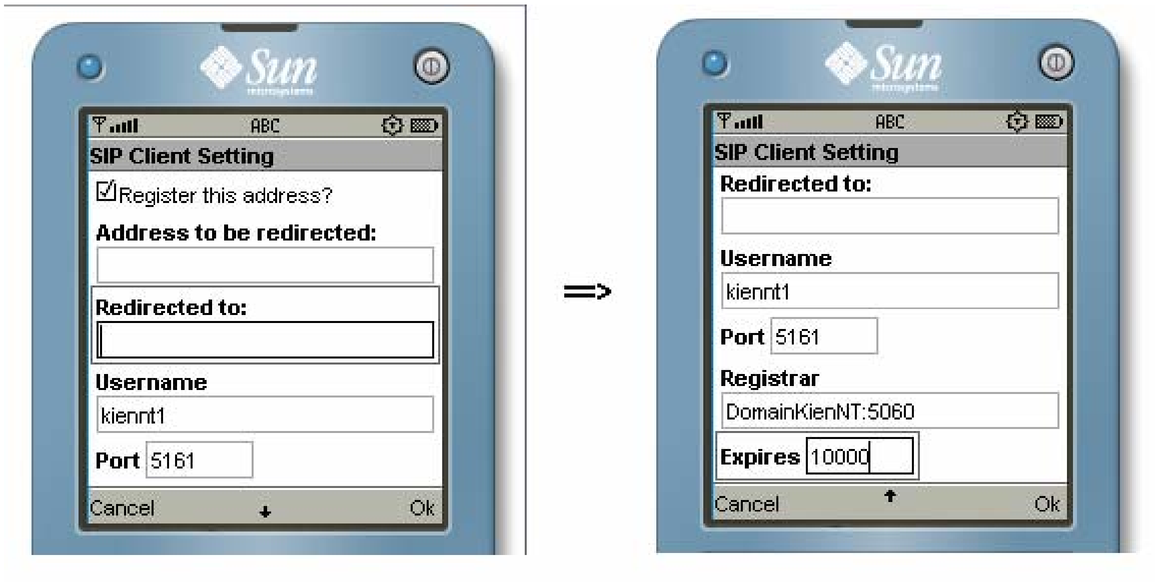
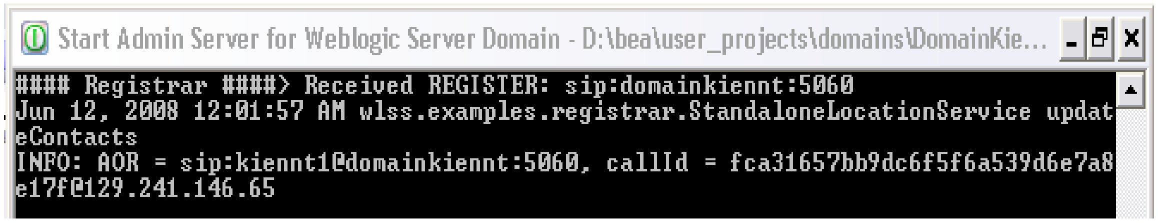


Figure 33. First Device Settings

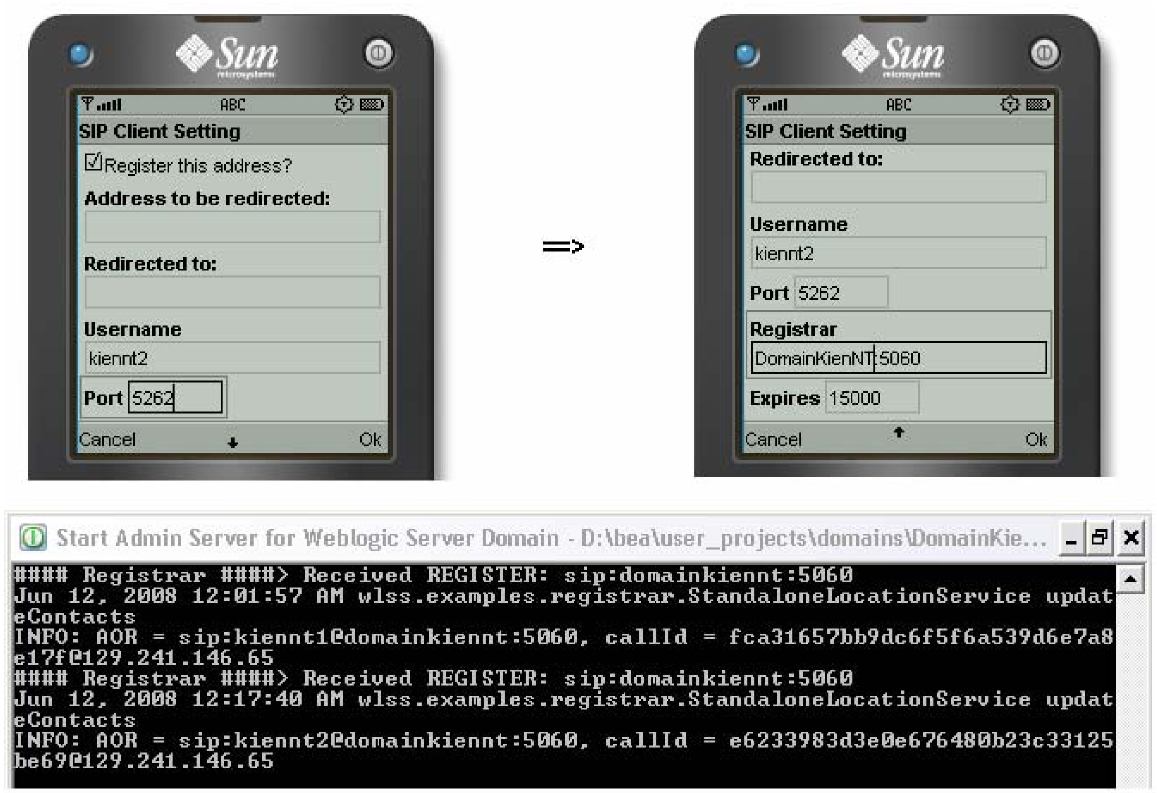
When we send those settings to register/bind the device with the Registrar server, it will report message notifying successful registering:


Start Admin Server for Weblogic Server Domain - D:\bea\user_projects\domains\DomainKie...

```
#### Registrar ####> Received REGISTER: sip:domainkiennt:5060
Jun 12, 2008 12:01:57 AM wlss.examples.registrar.StandaloneLocationService updat
eContacts
INFO: AOR = sip:kiennt1@domainkiennt:5060, callId = fca31657bb9dc6f5f6a539d6e7a8
e17f@129.241.146.65
```

Figure 34. Successful Register

The same thing goes with the second device, which is bind with the address **sip:kiennt2@DomainKienNT**.



☑Register this address?
Address to be redirected:
Redirected to:
Username kiennt2
Port 5262

Redirected to:
Username kiennt2
Port 5262
Registrar DomainKienNT:5060
Expires 15000

Start Admin Server for Weblogic Server Domain - D:\bea\user_projects\domains\DomainKie...

```
#### Registrar ####> Received REGISTER: sip:domainkiennt:5060
Jun 12, 2008 12:01:57 AM wlss.examples.registrar.StandaloneLocationService updat
eContacts
INFO: AOR = sip:kiennt1@domainkiennt:5060, callId = fca31657bb9dc6f5f6a539d6e7a8
e17f@129.241.146.65
#### Registrar ####> Received REGISTER: sip:domainkiennt:5060
Jun 12, 2008 12:17:40 AM wlss.examples.registrar.StandaloneLocationService updat
eContacts
INFO: AOR = sip:kiennt2@domainkiennt:5060, callId = e6233983d3e0e676480b23c33125
be69@129.241.146.65
```

Figure 35. Second Device Settings

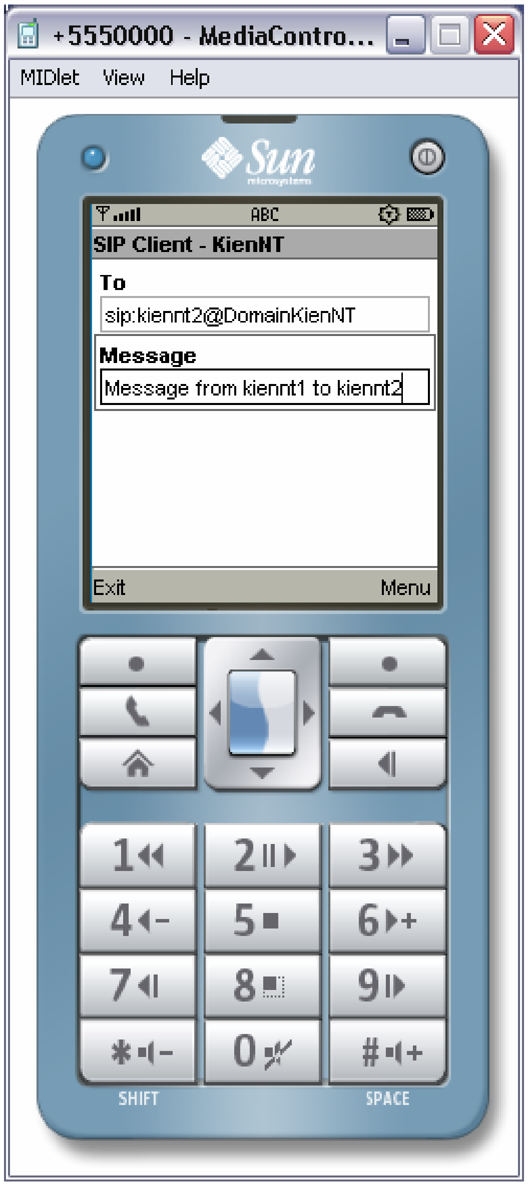
2.2. Scenarios

2.2.1. Sending and Receiving Messages

In this scenario, we demonstrate our SIP client text-messaging function by sending a message from the address **sip:kiennt1@DomainKienNT** to the address **sip:kiennt2@DomainKienNT** and vice versa.

The devices with the correspondent binding addresses are the same as in the previous section. Figure 36 will show the results.

Before sending the message:


+5550000 - MediaContro...

MIDlet View Help

SIP Client - KienNT

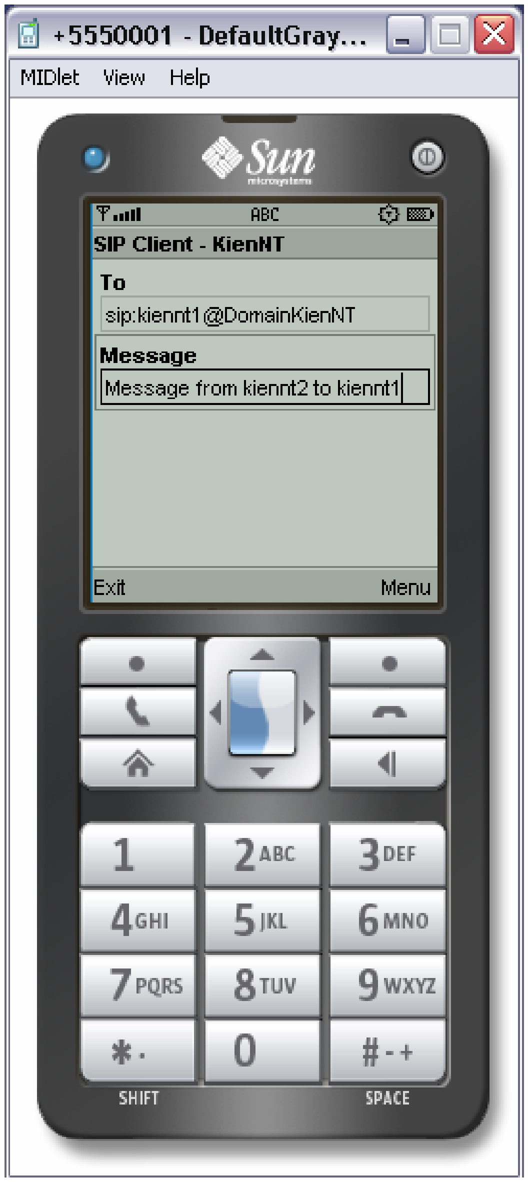
To

sip:kiennt2@DomainKienNT

Message

Message from kiennt1 to kiennt2

Exit Menu


+5550001 - DefaultGray...

MIDlet View Help

SIP Client - KienNT

To

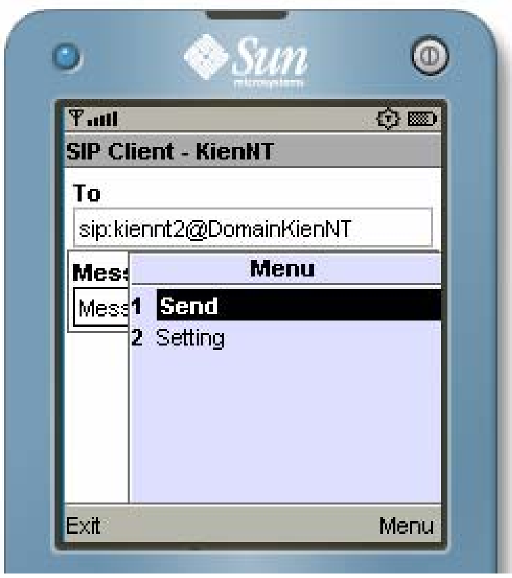
sip:kiennt1@DomainKienNT

Message

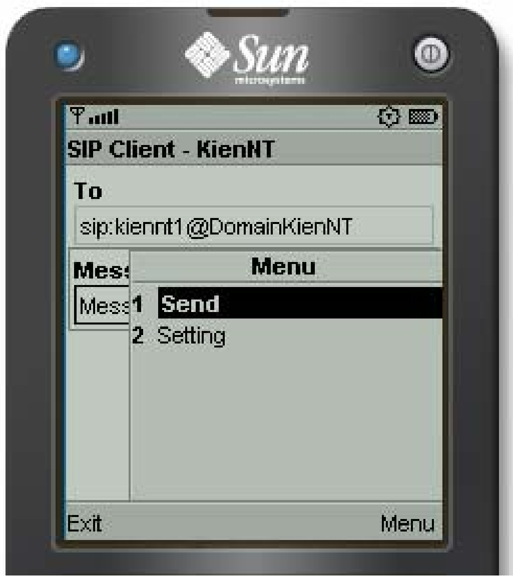
Message from kiennt2 to kiennt1

Exit Menu

Start sending the messages:

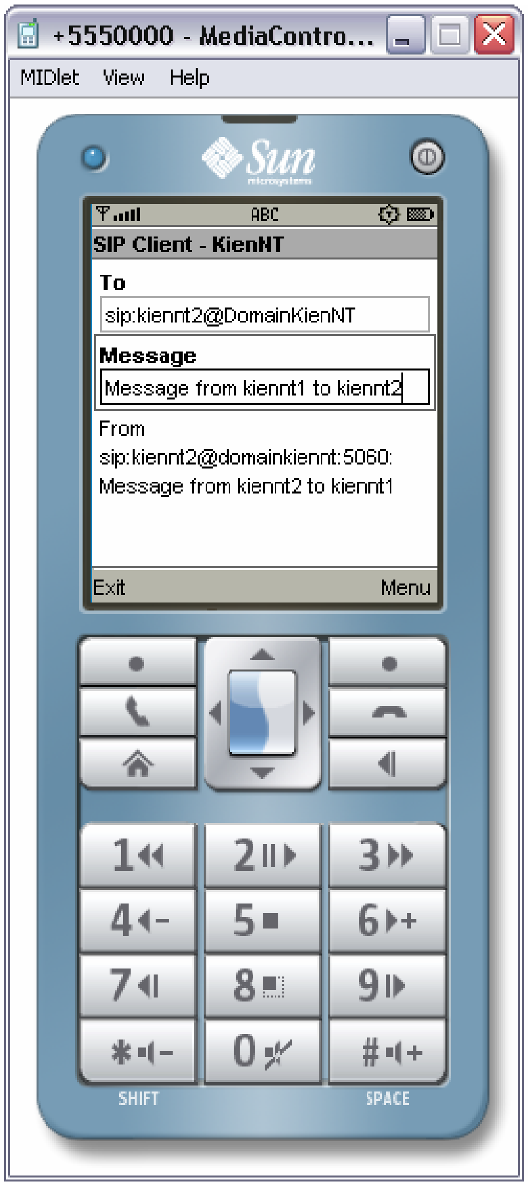


To
sip:kiennt2@DomainKienNT



To
sip:kiennt1@DomainKienNT

Receiving the messages:


+5550000 - MediaContro...

MIDlet View Help

SIP Client - KienNT

To

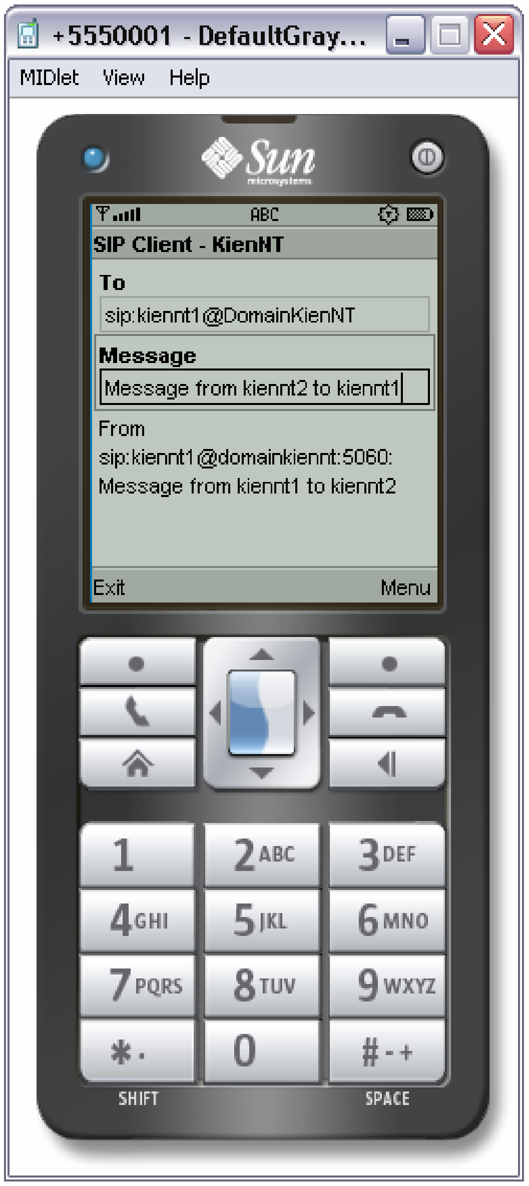
sip:kiennt2@DomainKienNT

Message

Message from kiennt1 to kiennt2

From
sip:kiennt2@domainkiennt:5060:
Message from kiennt2 to kiennt1

Exit Menu


+5550001 - DefaultGray...

MIDlet View Help

SIP Client - KienNT

To

sip:kiennt1@DomainKienNT

Message

Message from kiennt2 to kiennt1

From
sip:kiennt1@domainkiennt:5060:
Message from kiennt1 to kiennt2

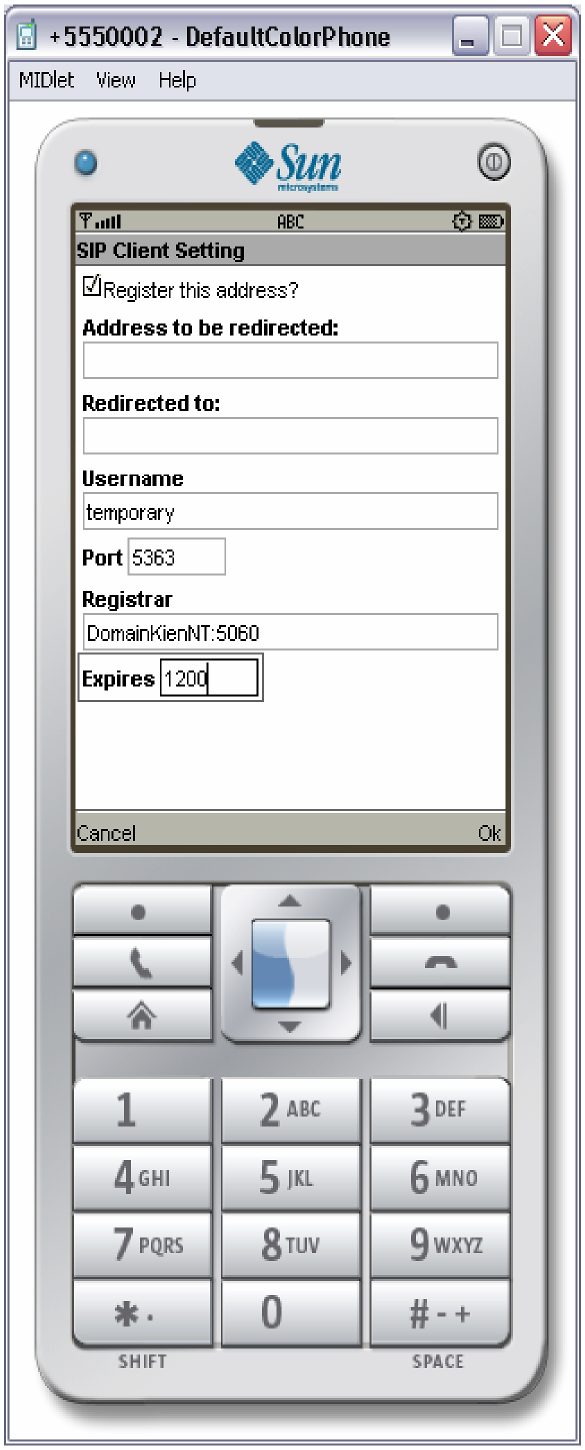
Exit Menu

Figure 36. Sending and Receiving messages

2.2.2. Transparent Call-Redirecting

In this scenario, we assume that the user of the address **sip:kiennt2@DomainKienNT** wants to have incoming calls redirected to his second address, for example **sip:temporary@DomainKienNT**.

To do this, we should have one more device that bind to the temporary address. For distinction, we choose a different mobile interface for that third device. The registering/binding steps are the same as before. Below is the third device with the correspondent address **sip:temporary@DomainKienNT**.



SIP Client Setting

☑ Register this address?

Address to be redirected:

Redirected to:

Username temporary

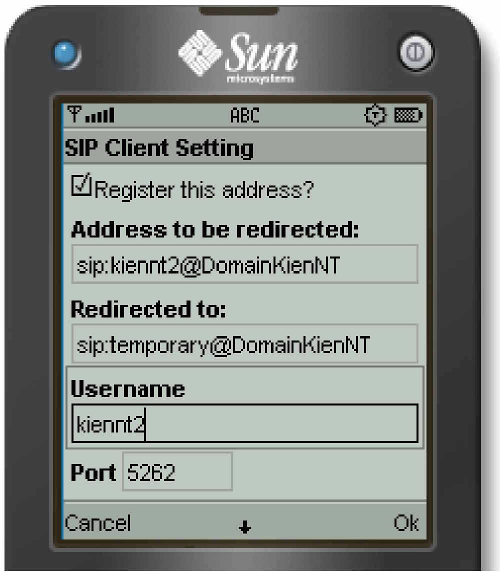
Port 5363

Registrar DomainKienNT:5060

Expires 1200

Cancel Ok

We also have to re-register the device that wants to have the incoming calls redirected in order to notify the change to the Registrar server. We do that by specify the *Address to be redirected* field (or **sip:kiennt2@DomainKienNT** in this scenario) and *Redirected to* field (or **sip:temporary@DomainKienNT** in this scenario) like below:



☑Register this address?

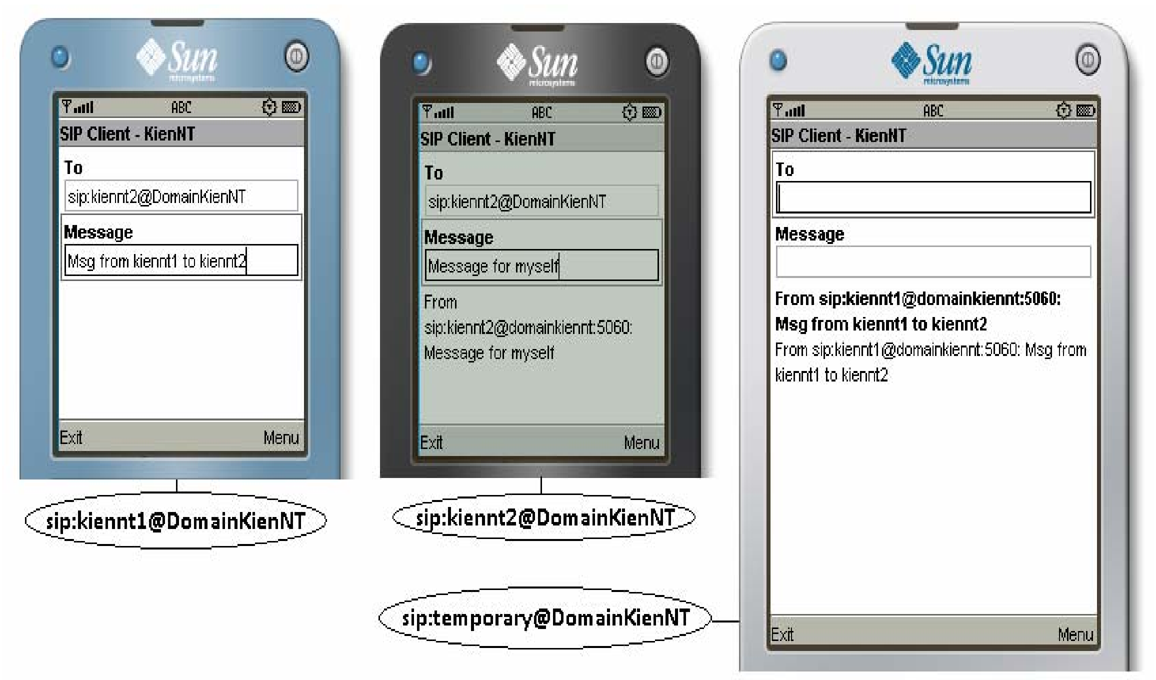
Address to be redirected: sip:kiennt2@DomainKienNT

Redirected to: sip:temporary@DomainKienNT

Username kiennt2

Port 5262

After all the configuring steps, we can start to send the message and see how it will be redirected:



To: sip:kiennt2@DomainKienNT
Message: Msg from kiennt1 to kiennt2

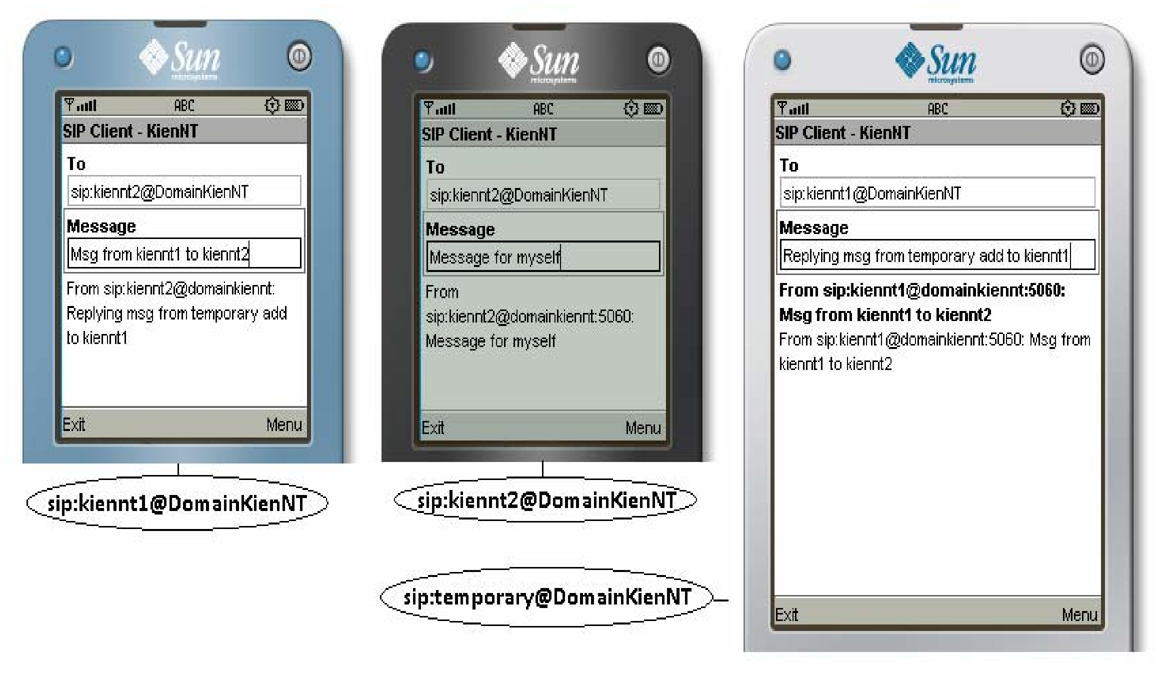
To: sip:kiennt2@DomainKienNT
Message: Message for myself
From sip:kiennt2@domainkiennt:5060: Message for myself

To
Message
From sip:kiennt1@domainkiennt:5060: Msg from kiennt1 to kiennt2
From sip:kiennt1@domainkiennt:5060: Msg from kiennt1 to kiennt2

Figure 37. Redirecting Messages

As we can see, the user **kiennt1** sends a message to user **kiennt2**, but that message was redirected to the address **temporary** (owned by the user **kiennt2**). The message that the user **kiennt2** sends to himself just for seeing which device belongs to whom more clearly.

Now the user **kiennt2** replies the user **kiennt1** using his **temporary** address. The user kiennt1 will not know about which “real” address that the user kiennt2 is using as in the Figure 38 below:


SIP Client - KienNT
To
sip:kiennt2@DomainKienNT
Message
Msg from kiennt1 to kiennt2
From sip:kiennt2@domainkiennt: Replying msg from temporary add to kiennt1

SIP Client - KienNT
To
sip:kiennt2@DomainKienNT
Message
Message for myself
From sip:kiennt2@domainkiennt:5060: Message for myself

SIP Client - KienNT
To
sip:kiennt1@DomainKienNT
Message
Replying msg from temporary add to kiennt1
From sip:kiennt1@domainkiennt:5060: Msg from kiennt1 to kiennt2
From sip:kiennt1@domainkiennt:5060: Msg from kiennt1 to kiennt2

Figure 38. Transparent reply

As we can see, when replying from the address **temporary**, the message still appear as it comes from the address **kiennt2** instead of **temporary** (see the *From* field in the mobile screen, it still appears that the message comes from **kiennt2**). Therefore, the user **kiennt1** will not know that he is talking to another address of the user **kiennt2**, otherwise he will be surprise since in fact the **temporary** address could be an strange address to him, and he may think he made call to a wrong address.

3. Discussion

From those demonstrations above, we can see how the SIP client works in sending/receiving normal messages and redirecting address through SIP protocol. In all the examples, we can see that only one domain name was used (which is *DomainKienNT*). In fact, the number of domain names can vary as we want to configure and if we have enough ports, computer’s resources. Hence, the address that a user wants his incoming calls to be redirected can be an address belongs to another provider (i.e. another domain like *ntnu.no*).

The *port* value when registering the devices is another point to note. When registering, we used different ports with different devices (5161, 5262 and 5363) but we did not specify those ports when sending to a SIP address. In theory, we can send a message to the address **sip:kiennt1@DomainKienNT** or **sip:kiennt1@DomainKienNT:5161**. They are the same addresses, and the ports have to be different from the server port (i.e. 5060 in our implementation) and other devices’ ports.

According to the proposed solution, our SIP client should support two transparent features (i.e. transparent authentication and call-redirecting) but only transparent call-redirecting was demonstrated. As we mentioned before, storing the authentication information in the SIP client's temporary database to use later for transparent authenticating is not a problem. The problem here is that we cannot setup a real fixed mobile convergence environment to do the test and generally, things working in theory will have problematic issues when applying to real environments. In our case, the problem of real conditions may come from how to let the devices access to the authentication database which is accessible by both HSS and S-CSCF, how to automatically tell the authenticating server to ask for the "proof" of authenticated devices, etc.

Another point is that since the SIP Client was developed in Java ME, it can run well in real mobile devices (assuming those devices are SIP-enabled). If these conditions are met, the mobile devices can send text messages to each other through SIP protocol without any problem.

V. Conclusion

In this thesis work, we have identified and studied thoroughly a central issue related to the deployment of IMS in a Fixed Mobile convergent environment, namely the identity management. Indeed, a user may have different identities in the fixed and mobile networks and several problems are observed as follows:

- A call initiated from the mobile network could not continue without disruption in the fixed network with WLAN support and vice-versa.
- A call addressed to the user at the mobile domain could not be delivered to the fixed network and vice-versa.

To solve the mentioned problems, this thesis work has performed the following tasks:

- A thorough study of the Identity Management and its standards.
- A careful analysis of the current identity management in both the fixed and mobile networks.
- An elaboration of identity management solutions for a Fixed Mobile convergent IMS environment.

These tasks are very interesting at the same time as they are quite challenging and demanding since the acquisition of knowledge from the different domains like security, user management, mobile networks, SIP, IMS, etc. are required.

1. Achievements

In addition to the two theoretical contributions which are:

- an introduction to Identity Management
- and a description of identity management in the fixed and mobile network,

the thesis work has a more practical and useful achievement:

- An identity management solution for a Fixed Mobile convergent IMS environment.

In the thesis work two Identity Management solutions have been proposed and analyzed as follows:

- Modified IMS and SSO-enabled SIP system
- Modified SIP-enabled Client

Due to time limitation, only the second solution is selected and a proof-of concept has been successfully designed and implemented. The proof-of-concept has demonstrates the following features:

- Single-Sign-On between the Mobile and Fixed domain enabling a mobile phone moving from the IMS mobile network to a SIP WLAN network without re-authentication.
- Identity Federation between the Mobile and Fixed domain enabling the delivery of calls addressed to one domain at the other domain.

The proof-of-concept proves the feasibility of the proposed solution.

2. Review and Future Work

Due to the time limitations, only of the two proposed solutions has been implemented. The implemented solution has also limitations. Although it works well with the SIP protocol at the client side, there should be more information exchange between the SIP client and the Proxy servers. Another limitation is that a transparent authentication is not yet demonstrated. For further works, it may be quite relevant to improve the proof-of-concept by removing the mentioned limitations.

Another qualified item for further works is the implementation of the first proposed solution which is more challenging and time consuming. A comparison and evaluation of the two solutions will also be a valuable contribution.

References

[1]. Phil Windley, *Digital Identity*, O'Reilly, August 2005.
[2]. Chris Kaler and Anthony Nadalin (Editors), *Web Services Federation Language (WS-Federation)*, http://www-128.ibm.com/developerworks/library/specification/ws-fed/, Last visited Jan 08.
[3]. OASIS organization, http://www.oasis-open.org
[4]. The Liberty Alliance - http://www.projectliberty.org/
[5]. Black, U., *ISDN and SS7*, Prentice Hall, 1997.
[6]. RFC 2916, E.164 number and DNS, available at http://www.ietf.org/rfc/rfc2916.txt, Last visited Jan 2008.
[7]. Kai Rennenberg, *IdM in mobile cellular network and related application*
[8]. Axalto, *SIM overview*, August 2005.
[9]. N.Komninos and B. Honarry, *Novel methods for enabling public key schemes in the future mobile systems.*
[10]. Moe Rahnema, *Overview of the GSM System and Protocol Architecture*, IEEE.
[11]. IPTEL, http://www.iptel.org, Last visited Jan 08
[12]. Niemi and Kaisa Nyberg, *UMTS Security*, Wiley 2003
[13]. 3GPP TS 23.228, v5.7.0, *IP Multimedia Subsystem (IMS)*, Release 5
[14]. Feng Boning, Do van Thuan, Ivar Jørstad, Tore Jønvik & Do van Thanh, *Identity Management in a Fixed-Mobile Convergent IMS Environment*,
[15]. Rakesh Khandelwal, *The Importance of Standard IMS Architecture*, TATA Consultancy Services, Ltd.
[16]. Feide project, Available at http://www.feide.no, Last visited Feb 08
[17]. Jay Lyman, Available at http://www.linuxinsider.com/story/51081.html
[18]. Ufinity Security & Mobile Solutions, Available at http://www.ufinity.com/news/20040716.php
[19]. Mobicome Project, Available at http://telenor.projectcoordinator.net/~mobicome, Last visited Feb 08
[20]. H. Tschofenig, J. Peterson, J. Polk, D. Sicker, and J. Hodges, *SIP SAML Profile and Binding*, Oct 2006, IETF Draft Standard.
[21]. Scott Cantor, Frederick Hirsch, John Kemp, and Eve Maler, *Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005
[22]. Jean-Daniel Aussel, *Smart Cards and Digital Identity*, Telenor's Telektronikk 3/4.07.
[23]. *Generic Authentication Architecture (GAA) and Generic bootstrapping architecture*, 3GPP, TS 33.220 3436, Dec 2006.
[24]. *Liberty Alliance and 3GPP security interworking*, 3GPP, TS 33.980, Sep 2006.
[25]. Noel Crespi and Mehdi Mani, *How IMS Enables Converged Services for Cable and 3G Technologies*, Telecom Sudparis
[26]. Tarkoma, Juha Tapio and Pin Nie, *Flexible SSO for SIP: Bridging the Identity Chasm*, Nokia NEON funded program.
[27]. 3GPP2 Project, *IMS Security Framework*, 3GPP2 S.R0086-0, December 2003
[28]. Salsano Stefano, Veltri Luca and Papalilo Donald, *SIP security issues: the SIP authentication procedure and its processing load*, IEEE Network, Vol 16, Dec 2002.
[29]. BEA System Inc., *BEA Weblogic SIP Server Datasheet*, Available at http://bea.com/sip, Last visited June 08.
[30]. The Apache Ant project, Available at http://ant.apache.org/

[31]. Eclipse Homepage, Available at http://www.eclipse.org/
[32]. EclipseME Homepage, Available at http://eclipseme.org/
[33]. Sun Microsystem, Sun Developer Networks, Available at http://java.sun.com/products/sjwtoolkit/
[34]. Emmanuel Proux, *An Introdution to SIP*, Available at http://dev2dev.bea.com/pub/a/2005/09/introduction-sip-part-1.html and http://dev2dev.bea.com/pub/a/2007/05/sip-javame.htm
[35]. Java Community Process, *SIP API for Java ME - Maintenance Release 2*, Available at http://jcp.org/aboutJava/communityprocess/mrel/jsr180/index2.html, Last visited June 08
[36]. Corrado Derenale, *An EAP-SIM Based Authentication Mechanism to Open Access Network*, April 2006