# INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

# Building Value-added Services using Mobile Agents in SIP based Internet Telephony

**Huan Adele Wang**

School of Computer Science
McGill University, Montreal
July, 1999

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Science

# Table of Contents

# List of Figures

## Résumé

La téléphonie par Internet attire beaucoup d'intérêt dans la société moderne. IETF SIP est une norme pour la téléphonie par Internet., mais l'architecture service de SIP a quelques points faibles. Elle ne supporte pas la mobilité des services par exemple.

Cette thèse propose une nouvelle architecture service basée sur les agents mobiles pour la téléphonie par Internet. Cette architecture est basée sur l'utilisation de deux agents mobiles pour chaque abonné. Un des deux agents contient les services origine et l'autre, les services terminaison. Ces deux agents agissent comme capsules et transportent les services auxquels l'usager est abonné. L'architecture apporte les avantages associés aux agents mobiles. Ceci inclut la réduction du traffic dans le réseau et la flexibilité dans l'approvisionnement des services a valeur ajoutée. L'architecture basée sur les agents est alors présentée. Le modèle conceptuel, le modèle d'implantation et les scénarios sont présentés. La mobilité des services est traitée de manière spéciale. SIP est utilisé comme exemple concret dans la thèse et «GrassHopper» comme base pour l'architecture d'implantation.

# Abstract

Internet Telephony is attracting a great deal of attention in modern society. IETF SIP is a standard for Internet telephony, but SIP service architecture has some shortcomings, for instance, it does not support service mobility.

This thesis proposes a novel mobile agent based service architecture for Internet Telephony. The architecture relies on the use of two mobile agents for each subscriber: one agent contains originating services and the other contains terminating services. These two mobile agents act as capsules and carry the services to which the user has subscribed. The architecture brings the advantage associated with mobile agents. This includes network traffic reduction and flexibility in service provisioning. We start the thesis by introducing value-added services. The MA based architecture for IP telephony is then introduced. Conceptual model, implementation model, and service scenarios are all described in this architecture. Special attention is paid to service mobility. SIP is used as concrete example throughout the thesis and "Grasshopper" as basis for the implementation architecture.

# Acknowledgments

I wish to thank my thesis supervisors Professor Petre Dini, Roch Glitho, and Nathan Friedman for their guidance, advice, and encouragement throughout the research. They provided insight into the novel MA service architecture for IP telephony. This thesis is benefited from their careful reading and constructive criticism.

I also truly thank CRIM (Centre de recherche informatique de Montreal) and Ericsson Research Canada. They provided wonderful research environments and financial support for this research.

Lots of people have helped me in the preparation of this thesis. First, I would like to thank Christophe Gourraud and Evelina Evloguieva of the team for the collaboration in the design and implementation of the MA service architecture for Internet telephony. I benefited greatly from formal and informal discussions with them.

I wish to thank the School of Computer Science for the graduate courses and the research environment. Thanks to Franca Cianci, Vicki Keirl, Lise Monogue, and Lucy St-James, for easing the procedure of dealing with the school.

Finally, I wish to thank my husband Jun Qiu, for his support and encouragement during my study.

# Chapter 1
# Introduction

Real-time transmission of voice over the Internet, also known as Internet telephony or IP telephony, is attracting a great deal of attention in modern society. In the development of IP telephony, people want to ensure that all the features and services (e.g. abbreviated dialing, call forwarding, and security screening etc.) supported by classical telephony systems can also be supported. To achieve this goal, an architecture that can facilitate service provision is required [ASA99].

Intelligent network (IN) is the currently implemented architecture for classical telephony [IN96]. A major difference between classical telephony and IP telephony is that, in the latter, call and connection controls do not have to be performed by the network node. They can be performed by the end systems located at the edges of the network. So, IP telephony services can be provided purely from end to end, without any intervention from network entities.

Session Initiation Protocol (SIP) is a simple signaling protocol proposed by Internet Engineering Task Force (IETF) for IP telephony. SIP can realize IP telephony services through implementing service logic and data at end systems: SIP proxy servers, redirect servers, or user agents. Although SIP can provide most value-added services, it is still immature, and does not provide enough flexibility and efficiency for those services. For example, SIP service architecture does not support service mobility and service on demand.

Mobile agent (MA) is an autonomous computer program that can migrate between physical locations within the network on behalf of a person or organization [IMA98]. Mobile agent application has several main advantages: reduction of network traffic, asynchronous interaction, robustness, and flexibility in service provision. Because of these advantages, mobile agent is suitable for providing advanced service architecture for SIP based IP telephony.

In this thesis, we will first propose a conceptual MA service architecture for SIP based IP telephony. Then, based on this architecture, we will illustrate the service

subscription, service execution, and service mobility scenarios. The architecture relies on the use of two mobile agents for each subscriber, one for originating services and the other for terminating services. This architecture has the great advantage of flexible and on demand service provision in an IP telephony environment. Finally, we will provide the implementation architecture using the Grasshopper platform, and explain the service subscription phase scenarios using this implementation architecture.

The rest of the thesis is organized as follows:

First, in Chapter 2, we will define different kinds of value-added services in classical telephony, and give an overview of IN architecture for those services. Second, in Chapter 3, we will introduce SIP infrastructure and its approach to value-added services. Then, in Chapter 4, we will present the state of art Mobile Agent technology and the existing MA projects in IN area.

In Chapter 5, we will present our conceptual MA service architecture for SIP based IP telephony. Not only does this architecture support flexible and instant service provision within a domain, but also supports service mobility and on demand service provision between domains.

In Chapter 6, we will describe the service scenarios and demonstrate how this architecture works. The service scenarios include subscription phase scenarios, execution phase scenarios, and service mobility scenarios.

In Chapter 7, we will present our MA based implementation architecture using the Grasshopper platform, then illustrate the mechanisms of using Grasshopper to realize service management unit (SMU) functionality for subscription phase scenarios.

Finally, in Chapter 8, we will summarize the whole thesis with conclusions and future work.

# Chapter 2

# Value-added Services in Classical Telephony

In classical telephony, the underlying network platform, consisting of switches, will offer basic services, such as audio, video, and data transmission services. On the other hand, the Intelligent Network (IN) platform will offer services to the end users representing enhancements of the basic services. The services provided by IN are referred to as value-added services, such as abbreviated dialing, call forwarding, security screening services, etc. A value-added service in Intelligent Network is a stand-alone commercial offering, characterized by the service features that enable the rapid construction of new services [IN96].

The rest of this chapter is organized as follows: In section 2.1 and 2.2 of this chapter, we first introduce value-added services and service features of Intelligent Networks. This is followed, in section 2.3, by a description of wireless intelligent network (WIN) services in Ericsson CMS 8800 system as a special subset of IN services. Then, we give a detailed description of IN architecture, key players, and its approach to value-added services using the IN conceptual model in section 2.4. Finally, we summarize this chapter with a conclusion in section 2.5.

## 2.1 Value-added Services

According to "Intelligent Networks" authored by Thomas Magedanz & Radu Popescu-Zeletin, the value-added services in IN can be divided into several categories [IN96]. They are *number translation services*, *alternate billing services*, *screening services*, *and other services*.

1. *Number translation services*. The primary advantage of this kind of service is its flexible numbering and routing functionalities. On the other hand, it may include customer control capabilities. The services are usually intended for the called party, but some services are also intended for the calling party. This category includes:

- Abbreviated dialing -- allows subscribers to call others by dialing an abbreviated number.
- Call forwarding -- enables forwarding of incoming calls to another destination.
- Call rerouting distribution -- allows incoming calls to be rerouted to a predefined destination in the case of a busy line, queuing overload, etc.
- Call distribution -- allows rerouting of incoming calls to different locations according to subscriber specific rules.
- Destination call routing -- allows routing of calls to different destinations depending on various conditions.
- Freephone -- allows reverse charging for a unique number.
- Follow me diversion -- allows redirection of incoming calls by remote control.
- Premium rate -- allows information service providers to obtain revenues from calls.
- Selective call forwarding on busy/don't answer -- allows pre-selected callers to be rerouted in case of a busy line.
- Universal access number -- supports one unique number for several terminating lines at different locations.
- Universal personal telecommunication -- enables a unique number for incoming and outgoing calls at any terminal.
- User-defined routing -- allows the subscriber to determine the routing for outgoing calls in agreement with a routing preference list.

2. *Alternate billing services*. The advantage of this category of services is its flexible charging capabilities. This category includes the following services:
   - Account card calling -- allows calls from any card-reading telephone by charging a specified account.
   - Automatic alternative billing -- allows calls from any telephone by charging a separate account.
   - Credit card calling -- allows calls from any telephone by charging a credit card.
   - Split charging -- allows charge splitting between calling and called parties of a call.

- Premium rate -- allows information service providers to obtain revenues from calls.

3. *Screening services.* This kind of service takes advantage of flexible screening capabilities in order to restrict call establishment, and includes:
   - Originating call screening -- supports restriction of incoming calls. The subscriber can construct a screening list to specify whether incoming calls are restricted or allowed.
   - Security screening -- enables screening of users seeking network access. It is intended to hinder unauthorized access to the subscriber's network, systems, or applications.
   - Terminating call screening -- supports restriction of outgoing calls. The subscriber can construct a screening list to specify whether outgoing calls are restricted or allowed.

4. *Other services.* These are services that cannot be included in any of the three categories above, because they provide complementary capabilities. Some services in this category may also incorporate flexible routing, charging and/or screening capabilities, but the basic service provided here is focusing on something particular.
   - Completion of call to busy subscribers -- supports automatic call back when the line becomes free.
   - Conference calling -- allows multiple parties within a single call.
   - Malicious call identification -- enables logging of incoming calls.
   - Mass calling -- supports handling of high call volume. It provides capabilities for instantaneous, high-volume traffic routed to one or more destinations depending on specific conditions.
   - Televoting -- enables voting via the network by allocating one or more temporary numbers.
   - Virtual private network -- simulates a private network by using public network resources.

## 2.2 Service Features

An IN service feature reflects a specific aspect of the functionality of an IN service, because a service feature is a specific part of a telecommunication service that can be used in conjunction with other telecommunication service features as part of a commercial offering. In other words, this means that the identified service features can be arbitrarily combined to define services.

Generally, IN service features can be divided into the following categories [IN96].

1. *Numbering features* allow the use of dedicated numbers for making calls, and include abbreviated dialing, one number, personal number, and private numbering plan.
2. *Routing features* allow the destination number to which an incoming call should be routed to be determined, and include call distribution, call forwarding, follow-me diversion, time-dependent routing, and origin-dependent routing.
3. *Charging features* permit control of service charging, and include premium charging, reverse charging, and split charging.
4. *Access features* supply functions for access control, and include authentication, authorization code, off-net access, and off-net calling.
5. *Restriction features* support the screening of calls regarding various conditions, and include call limiter, call gapping, closed user group, originating call screening, and terminating call screening.
6. *Customization features* allow the customer to define and modify service parameters, and include customer profile management, customer recorded announcement, and customized ringing.
7. *User interaction features* support dialogues with a called party during call setup, and include attendant, consultation calling, destination user prompter, and originating user prompter.
8. *Other service features* cannot be allocated to one of the above categories. Most of these features provide specific call handling capabilities, which include automatic call back, call hold with announcement, call logging, call queuing, call transfer, call waiting, mass calling, meet-me conference, and multiway calling.

A service feature is either a core part of a telecommunication service, which means that it is fundamental to the telecommunication service, or an optional part offered as an enhancement to a telecommunication service. By adding or dropping service features, different kinds of IN service can be created.

### 2.3 Ericsson WIN Services

Wireless Intelligent Network (WIN) is the mobile adaptation of the Intelligent Network concept. Ericsson has implemented WIN services product line for its CMS 8800 product family. The purpose of this product is to develop and deploy services in a quick cycle time without demanding a major upgrade of the entire wireless network. Ericsson WIN services are divided into two categories: **subscriber services** and **network services** [WINS].

i) **Subscriber Services** are services provided to individual users. They can be triggered at three different places in the call model – originating part, terminating part and transfer part. Based on different trigger points, WIN subscriber services are grouped into three sub-types: AIN, BIN and CIN services, which stand for originating services, terminating services and transfer services separately [CMS88] [WINS].

   a) **AIN Services**: also known as originating services, are triggered in the originating part of the call, which include:
      - *Customer Control Access (CCA)* allows subscribers to modify certain service subscription data using their terminals. The CCA service is available with the Outgoing Call Allowance (OCA), Outgoing Call Restriction (OCR), and Selective Call forwarding (SCF) services described later in this section.
      - *Bulk Number Translation (BNT)* is a call translating service that allows subscribers to define an abbreviated dialing code for a range of numbers. The subscriber dials the abbreviated dialing code and an additional suffix number to compose the destination number.

- *Private Numbering Plan (PNP)* is a call translating service that allows subscribers to use abbreviated dialing codes to reach frequently called numbers. In CMS 8800, the PNP service is available to both individual subscribers and groups of subscribers. Individual subscribers can specify up to 50 abbreviated dialing codes, and a Selective User Group (SUG) can specify up to 49 Private Numbering Code (PNCs) for frequently called numbers external to the SUG. A SUG can also define an unlimited number of PNCs for the members of its group.
- *Outgoing Call Allowance (OCA)* is a call screening service that allows subscribers to specify how outgoing calls are processed. Calls placed from a subscriber's telephone are routed to the dialed number only if it is on the OCA screening list of authorized numbers. The OCA service is also available with Customer Control Access (CCA). OCA with CCA allows subscribers to define 5 profiles tailored to meet the subscribers' needs.
- *Outgoing Call Restriction (OCR)* is a call screening service that allows subscribers to specify how outgoing calls are processed. Calls placed from a subscriber's telephone are verified against the OCR screening list of unauthorized numbers. If a dialed number is found on the list, the call is blocked and routed to an announcement informing the caller that the call could not be routed as dialed. The OCR service is also available with Customer Control Access (CCA).

b) **BIN services:** also known as terminating services, are triggered in the terminating part of the call, which include:

- *Selective Call Acceptance (SCA)* is a call screening service that allows subscribers to receive calls from specific telephone numbers. Calls are routed to the subscriber only if the calling party is on the SCA screening list of authorized numbers.
- *Selective Call Rejection (SCR)* is a call screening service that allows subscribers to reject incoming calls from specific telephone numbers. Calls are

routed to the subscriber only if the calling party is not on the SCR screening list of unauthorized numbers.

- *Selective Call Forwarding (SCF)* is a screening service that allows subscribers to use their phones more efficiently by specifying how incoming calls are processed. The SCF screening service allows subscribers to specify a screening list of calling party numbers, designate where calls are forwarded for numbers on the screening list, and specify where calls are forwarded for numbers not on the screening list. The SCF service is also available with Customer Control Access (CCA).

c) **CIN services:** also known as transferring services, are triggered by call transfer conditions in the intelligent network, which include:

- *Flexible Call Forwarding (FCF)* is a call transferring service that allows subscribers to forward their calls to various destinations based on a subscriber – defined forwarding schedule. The forwarding services are activated under the following circumstances: inactive or does not answer, unconditional call transfer, and call transfer on busy.

- The forwarding destinations are used in forwarding schedules defined by the subscriber. Forwarding schedules consist of time periods that are assigned to destination numbers. The FCF subscriber can define two types of schedules.

  Weekday schedule allows the subscriber to define a forwarding schedule for regular operations. Date schedule allows the subscriber to define a forwarding schedule for special days. The date schedule takes precedence over the weekday schedule, even if date is also defined for the weekday schedule.

ii) **Network Services** are applied to the network instead of individual user, which include following services [CMS88] [WINS]:

- *Toll Free Calling (TFC)* is a translation service. TFC number owners can offer users a means to contact them free of charge. TFC allows mobile subscribers to

have calls to special numbers, such as an 800 number, routed to various destinations based on an owner defined forwarding schedule.

- *The Wireless Virtual Private Network (WVPN)* is not a single service, but a collection of CMS88 WIN services that allows operators to provide subscribers with the benefits of private network numbering plans with the advantage of wireless mobility. The WVPN is created by assigning subscribers within an organization to a Selective User Group (SUG). SUG members have access to a number of WIN services provided by the WVPN. WVPN consists of the BNT and PNP core services, and can optionally include the OCA, OCR, SCA, SCF and SCR services.

## 2.4 Intelligent Network

Intelligent Network (IN) is a service-independent telecommunication network architecture, which separates service logic from network switching equipment [Index]. That is, intelligence is taken out of the switch and placed in computer nodes that are distributed throughout the network. This allows the service provider to develop and control services more efficiently. The IN architecture allows a variety of different services to be provided to customers independent of the underlying network technologies. The IN defines a service-oriented functional architecture that enables the provision of a set of generic service components. These service components can be combined to construct new telecommunication services. The goal of this service control architecture is to provide an open platform supporting the uniform creation, provision, control, and management of services in telecommunications. The IN architecture approach has the following major objectives [IN96]:

- The IN architecture should be *service independent* to enable the realization of an open set of telecommunication services based on a common architecture. This target is addressed by the IN through the definition of generic service building blocks – service-independent building blocks (SIBs), which representing modular and reusable functions, and could be used for the construction of many different services.

- The IN architecture should be *network independent* to allow the uniform implementation of services on top of any bearer network (public switched telephone network, integrated services digital network, and mobile network etc.). This target is achieved through definition of service-oriented functional network elements, in which service switch functions are separated from service control functions. These functions could be flexibly allocated among physical network entities.
- The IN architecture should support *vendor independence* to insure the interoperability of IN equipment provided by multiple vendors. This target is achieved through the definition of unique interfaces and protocols between the defined IN network elements.

### 2.4.1 Intelligent Network Key Players

To realize those objectives, four key players can be identified in an intelligent network environment: *network operator*, *service provider*, *service subscriber*, and *service user*.

- The *network operator* is a public or private company or organization that provides the structured network and its resources for the execution of IN services. The structured network provides the underlying basic services, such as video, audio and data transmission services.
- The *service provider* is a public or private company that develops and provides IN services commercially over the common structured network and its underlying basic services. The service provider is responsible for the provision and continuous availability of subscribed services. This means that its role is mainly in service management, service creation, service deployment, service administration, and service operation.
- The *service subscriber* is usually an organization that obtains an IN service from a service provider on a contractual basis and has to pay the charges to that service provider. However, residential users may also be service subscribers for some services.

- The *service user* is the person who has access to and makes use of a service, i.e. represents the called or calling party depending on the type of IN service, but will not necessarily be the service subscriber. For example, the service subscriber may be a company subscribing to a virtual private network (VPN) service and its employees may be the service users.

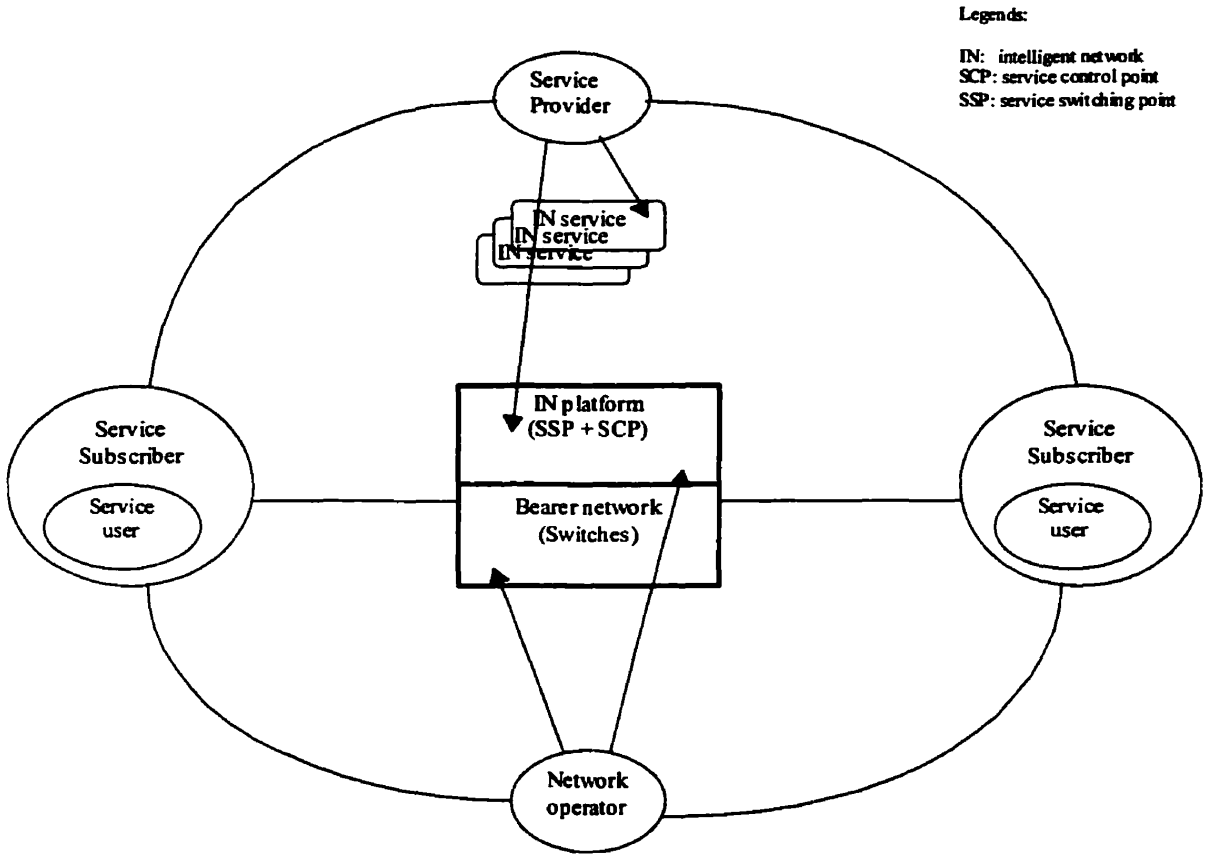



Figure 1: Intelligent Network Key Players

### 2.4.2 IN Architecture

To realize IN architecture, people retrieve the service logic and data from network switches to "intelligent nodes" and define a switch-independent interface/protocol. The advantage of this approach is obvious: by putting a new service program and the related service data outside the switching network, it is possible to introduce a new service ubiquitously throughout a given network.

However, the fundamental prerequisite of this approach is the real time connection between the switches referred to as service switching points (SSP), and the "intelligent nodes" known as service control points (SCP). This fast, reliable and standardized interconnection of SSP and SCP forms the basis of the IN architecture, and it became possible through the introduction of "common channel signaling system no. 7 (SS7)" systems based on the concept of out-of-band signaling. Relying on this SS7 network, "intelligent nodes" that contain service logic and data can remotely control the establishment of call connections at the request of the switches. Therefore, services can be easily introduced or modified in an "intelligent node". Within this approach, service switching and control are separated, requiring the definition of the corresponding interfaces between the switches and the "intelligent nodes" [IN96].

In order to fulfill IN architecture, two basic items are addressed. First, the basic call model in the switches should be enhanced to facilitate the basic and supplementary service provision. Trigger event logic in the switch has to be added to the basic call model to start interactions with the external IN service logic. Secondly, a corresponding protocol has to be defined for the dialogue between the SSP and SCP on top of the signaling network.

To support multiple services by the IN architecture, both the call model and the protocol used between the switch and the service control node should be service independent. IN architectures separate intelligent network capabilities from specific services by providing generic service features and switch call models that are applicable to different IN services. The IN architecture has the following components:

1. SSP is a network switch, which contains limited service access logic required to suspend a call that requires special handling. The SSP recognizes IN service calls and routes the corresponding queries to the SCP via the SS7 signaling network. SCP commands will be used by the SSP to accomplish the call.
2. SCP is an on line, fault tolerant, transaction-processing database that provides call-handling information in response to SSP queries. The SCP memory can be added to an in-service SCP without interrupting service handling. SCP is designed to support multi-service operation.

3. The SMS is a service management system containing the reference service database used to manage IN platforms. SMS is responsible for maintenance and supervision SCPs, and also remote software downloading. Transactions from the SCP to the SMS include performance measurements, traffic statistics, and billing data. Both network operators and customers can communicate with the SMS to retrieve service reports or update data. The SMS is integrated in an operation support system that supports network operation, administration, and maintenance functions, and normally resides in a host computer.
4. IP is an additional intelligent peripheral connected to an SSP to provide enhanced services, such as announcements, speech synthesizing, and speech recognition, under the control of an SSP or SCP. IP functionality is usually required for interactive intelligent network services.
5. STP is the signaling transfer point in the SS7 packet network. STP is a very high capacity, very reliable packet switch that transports signaling messages between the network nodes.

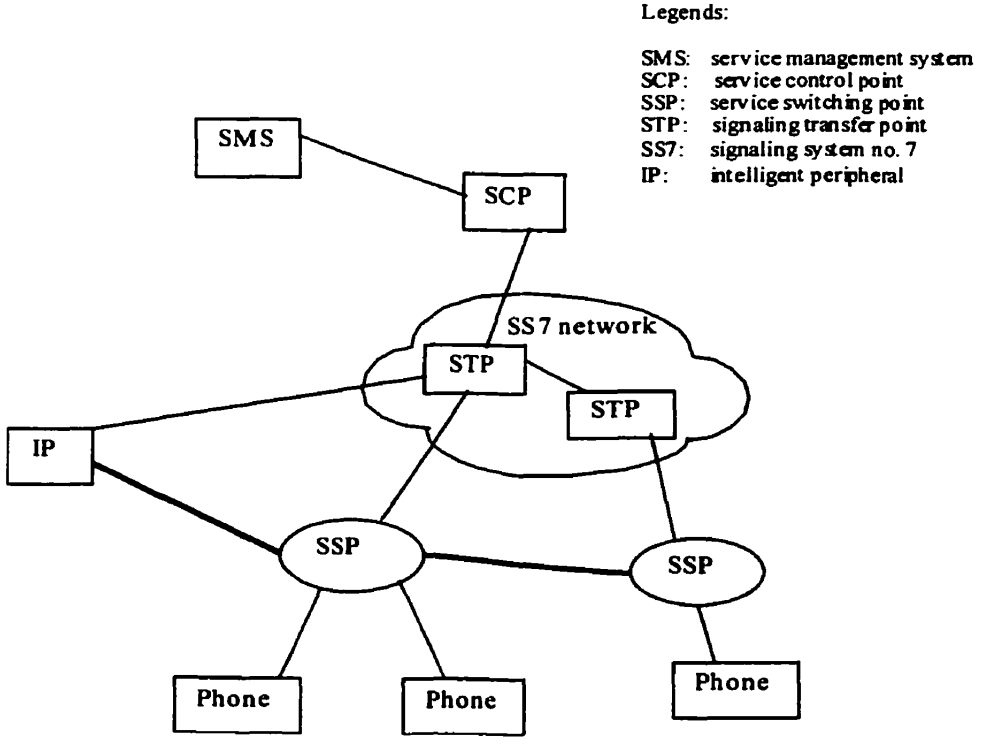



Figure 2: Intelligent Network Architecture

### 2.4.3 Service Provision with IN Conceptual Model

An intelligent network can be considered as an additional network layer on the top of any bearer network, such as public switched telephone network (PSTN), integrated services digital network (ISDN), or broadband ISDN (B-ISDN). The intelligent network provides a service-oriented network architecture that separates service control functions from service switching functions, with both types of functions being implemented in different physical equipment. This is supported by a clear definition of the relationships between these functions, thus providing for network and vendor independence. On the other hand, another major target of intelligent network is service independence. During the development of many advanced telecommunication services, it becomes clear that all of these services have similar functionality, i.e., they are based on a set of "service components". Hence, it requires a generic set of reusable service components that could be used for the construction of a new service. It also requires an IN programming interface that can be used for easy service creation. Therefore, a service provider can make use of these service components and combine them to implement a new service. These service components are referred as service-independent building blocks (SIBs).

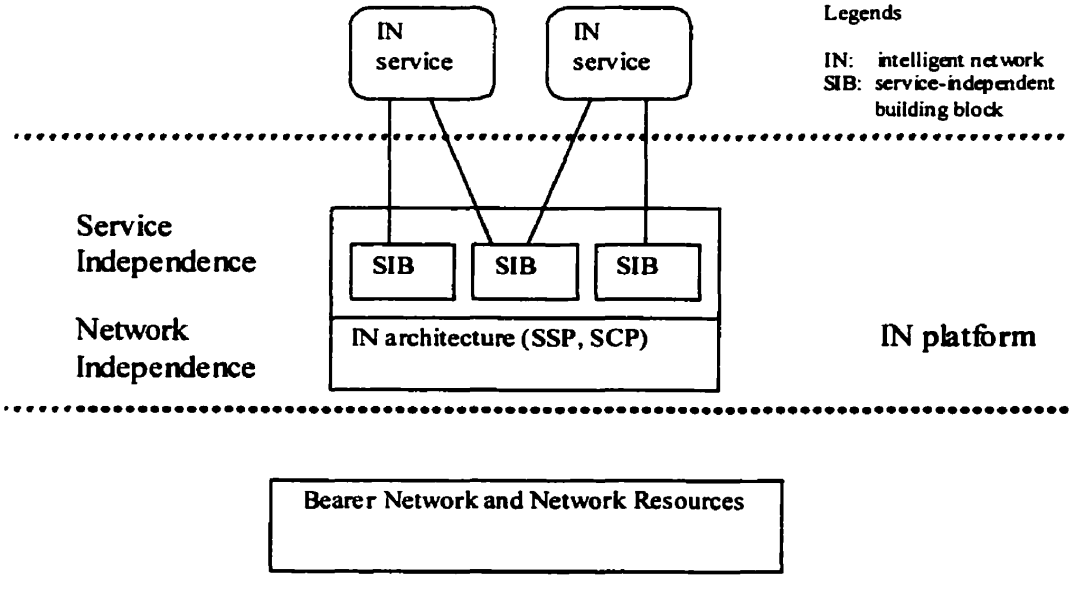


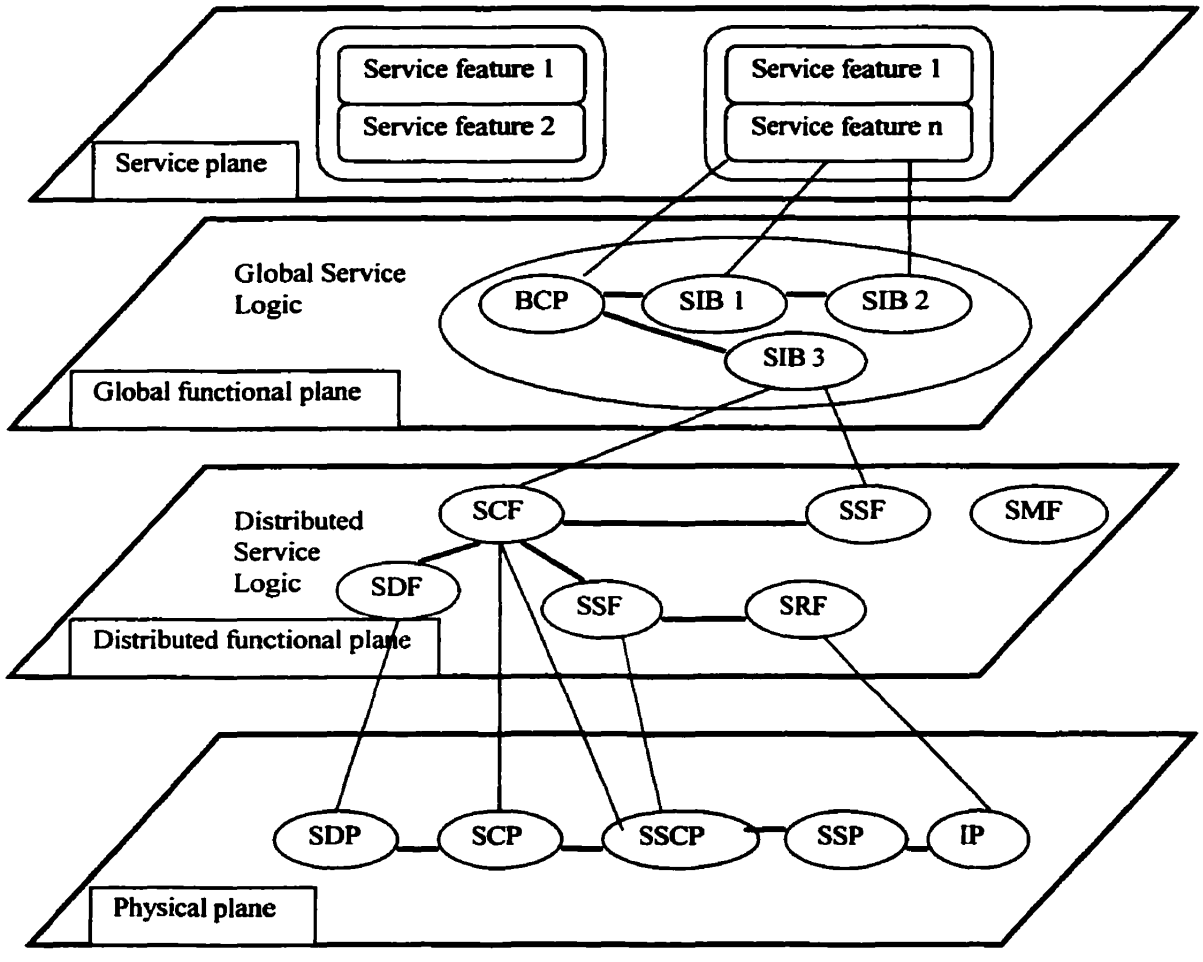

Figure 3: Network Independent and Service Independent IN Platform

To realize intelligent network services, the whole engineering process of intelligent network could be captured in a reference model known as the IN conceptual model (INCM). The basic idea of INCM is to define a top-down approach for the definition of IN architectures based on IN service capabilities to be supported. Thus the INCM defines four planes which address service design aspects, global service provisioning functionality, distributed service provisioning functionality, and physical aspects of an intelligent network.

INCM is only a modeling tool for describing the capabilities and characteristics of IN structured network, it is not IN architecture itself. Only the lower two planes of the INCM address the IN architecture, whereas the higher two planes focus on the creation and implementation of IN services, which are independent of any IN architecture. INCM should be regarded as a road map to be followed while defining IN architecture. Following INCM, people can start with the definition of the services, which should be supported in a network independent way. After the definition of the functional scope of the service, the INCM leads to a further decomposition of these service capabilities into smaller functional blocks (SIBs) to achieve some degree of service independence. This means that the services guide the definition of SIBs, which define basic service capabilities in a network independent way.

CCITT developed the INCM to provide a framework for the design and description of each IN component and the target IN architecture [INO93] [IN96]. The INCM is structured into four planes: a *service plane*, a *global functional plane*, a *distributed functional plane*, and a *physical plane*. Following the INCM, the identified basic service capabilities will be used for the definition of IN architecture. This means that the IN architecture must be able to support the distributed implementation of these service capabilities. The IN architecture definition is subdivided into two stages. In the first stage the network entities are defined in terms of functional elements and their interactions. In the second stage these elements are allocated to specific physical entities. This means that the services can be regarded as the requirements for the architecture definition. Thus, the terms "INCM" and "IN architecture" must be distinguished, as INCM is used to define a particular IN architecture.

INCM has the following four planes [IN96]:




Legends:

BCP: basic call process
SCF: service control function
SSF: service switching function
SDF: service data function
SRF: specialized resource function
SMF: service management function
SIB: service-independent building block
SCP: service control point
SSP: service switching point
SSCP: service switching and control point
SDP: service data point
IP: intelligent peripheral

Figure 4: Intelligent Network Concept Model (Adapted from [IN96])

- The *service plane* is the uppermost plane, which describes the services from a user's points of view. The service implementation and the underlying network technology are transparent to the users. Each service consists of one or more generic blocks called

service features, and each service feature could be a complete service or part of a service.

- The *global functional plane* defines the network from a global high level perspective as a single programmable entity, hiding the complexity of the distribution of functions. This plane deals with the service creation and contains SIBs that will be used as standard reusable components for efficient and fast service feature implementation. Global service logic describes how SIBs are chained together in order to achieve service features. It also describes the interactions between basic call process and appropriate SIB chains.
- The *distributed functional* plane is of interest to network designers and providers. It models the distributed view of an intelligent network in terms of network functional units, known as a set of functional entities (FEs), such as the service switching function (SSF) and the service control function (SCF). The distributed functional plane provides transparency to the physical network elements. Each SIB of the global functional plane is decomposed into a set of client-server relationships between one or more functional entities in the distributed functional plane. Each functional entity may perform a variety of actions.
- The *physical plane* models the physical aspects of the intelligent network, and contains the real view of the physical network. The physical plane defines different physical entities and their interfaces to which the relevant functional entities are located. A complete functional entity must be implemented in the same or different physical entities according to the different characteristics of the underlying network technologies and service-specific access requirements. Examples of physical elements are SSPs and SCPs. Physical entities communicate through Intelligent Network Application Protocol (INAP) for IN service execution.

As described above, the INCM can be conceptually divided into two parts. The upper two planes of the INCM focus on service creation and implementation by means of generic service building blocks and provide the desired service independence. While the lower two planes define a generic intelligent network service provisioning architecture, which is independent of specific bearer networks.

## 2.5 Conclusion

Until now we have introduced value-added services in classical telephony, as well as intelligent network architecture and its approach to those services. Internet telephony is different from classical telephony in that call and connection controls can be performed by the end systems located at the edges of the network. SIP (session initiation protocol) is the Internet Engineering Task Force (IETF) standard for Internet telephony, and it can provide value-added services by implementing the service logic and data at the end systems. In the next chapter, we will describe the idea of Internet telephony as well as the SIP approach to Internet telephony services.

# Chapter 3

# SIP Approach to Value-added Services in IP Telephony

As mentioned in the first chapter, real-time transmission of voice over the Internet, also known as IP telephony, is attracting a great deal of attention. In the development of IP telephony, service providers want to ensure that all the services and features supported by classical telephony systems can also be supported. There are a few protocols that support IP telephony, and SIP (session initiation protocol) is one lightweight protocol among them. SIP is a simple signaling protocol for Internet conference and telephony, which can realize value-added services through the implementation of its proxy server, redirect server, or user agent.

This chapter describes IP telephony, SIP for IP telephony signaling, and SIP approach to IP telephony services. First, in section 3.1, we introduce the basic idea of IP telephony. Then, in section 3.2, we discuss SIP, its infrastructure, its addressing and naming, its requests and responses, and its headers. In section 3.3, we discuss, SIP basic operations and how SIP can be used for IP telephony services. Finally, we conclude this chapter with section 3.4.

## 3.1 IP Telephony and its Protocol Stack

Voice over IP is a digitized voice signal sliced into packets and sent across a packet-switched network. At the receiving end, the re-assembled packets arrive as normal sounding voice call. The IP network can be any packet switched network - including ATM, frame relay, the Internet, a corporate Intranet, a T-1 or a 56Kbps line.

Internet telephony, also known as IP telephony, is the real-time delivery of voice and other multimedia data types between two or more parties across networks using the Internet protocols, and the exchange of information required to control this delivery [AAP99]. IP telephony offers the opportunity to design a global multimedia communication system that may eventually replace the existing telephony infrastructure.

Early reaction to IP telephony has focused on their ability to reduce dramatically or eliminate the long distance charges incurred when calling over the public telephone network. Now users are increasingly attracted to the benefits that IP telephony can offer, such as voice and data conference, transparent routing of calls to end users, and decreased administration costs. IP telephony offers a larger degree of freedom to allocate functions between network servers and user-supplied and operated end systems.

IP telephony also achieves a better utilization of the available bandwidth than the circuit switched telephony. In the circuit-switched telephony, a circuit is reserved during the call, and it cannot be used by anybody else even if no traffic is being generated at that moment. IP telephony consumes bandwidth only if it has some data to send.

Unfortunately, unlike voice transmission over PSTN (public switched telephone network), voice over IP networks is frequently unreliable, non-deterministic, and constrained, because there are few defined standards implemented for IP telephony. Therefore, we need standards and architectures for multimedia distributed applications and IP telephony services that support information flows in terms of a generic platform for open distributed environments.

Two protocol architectures have served as the basis for the development of interoperable communications standards: the TCP/IP protocol suite and the Open Systems Interconnection (OSI) reference model. TCP/IP is the most widely used interoperable architecture, which is a result of protocol research and development conducted on the experimental packet-switched network, ARPANET.

The IP telephony protocol stack incorporates many protocols. TCP, UDP, IP and the protocols beneath are used to serve the protocols above them with a possibility to send/receive data across the network.

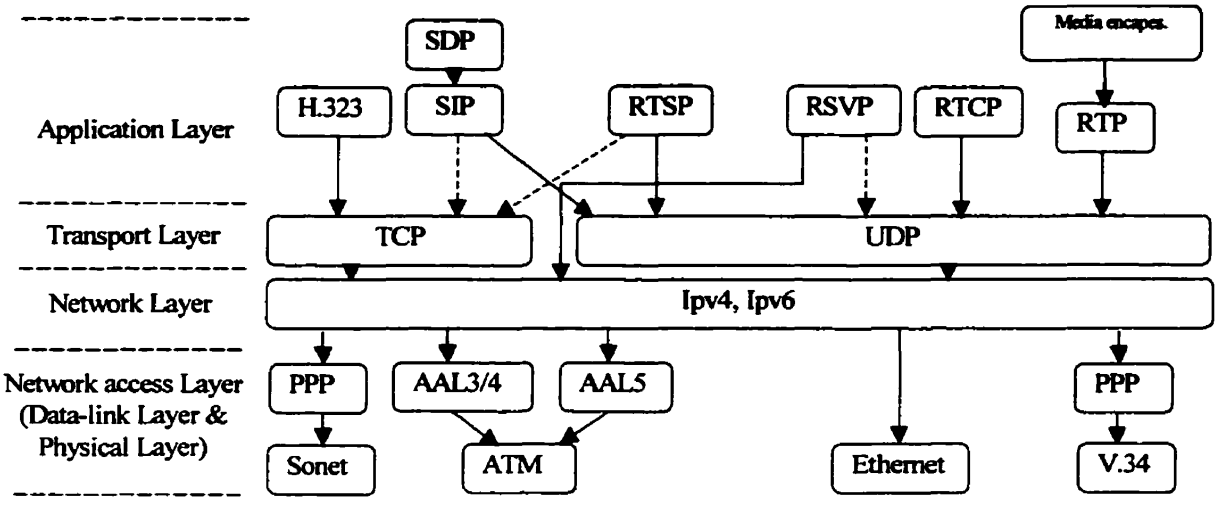



Figure 5: The IP Telephony Protocol Stack

As shown in the above figure, IP telephony protocol stack is composed of the following layers:

a) *Application layer* includes all the functions of each concrete application.
   1. RTSP (Real Time Streaming Protocol) [RTSP98] is the standard protocol for controlling multimedia streams over the Internet.
   2. H.323 is the International Telecommunication Union (ITU) protocol for Internet multimedia communication. Before the transmission of all the real time packets containing the audio samples, the connection between the two end systems has to be established. H.323 protocol performs this signaling operation. After the establishment of the connection, this protocol controls the call, and when it is finished, it indicates that the resources can be released [AAP99].
   3. SIP is designed as a part of the overall IETF (Internet Engineering Task Force) multimedia data and control architecture. It is an application-layer signaling and control protocol for creating, modifying and terminating sessions with one or more participants. It handles basic call signaling, user location and basic registration. Call control signaling can also be added by using an extension. Billing, Quality of Service (QoS), session content description and other functionality are handled by other protocols.

4. SDP (Session Description Protocol) is used for describing multimedia sessions.
5. RSVP (Resource Reservation Protocol) is used to reserve resources over the Internet [RSVP97]. RSVP is part of the Internet Integrated Service (IIS) model, which ensures best-effort service, real-time service, and controlled link sharing.
6. RTP (Real-Time Transport Protocol) is the protocol for the transport of real-time data, including audio and video [RTP96] [AVC96]. Although RTP can run over connection oriented or connectionless lower layer protocols, which are in charge of framing and segmentation, it is generally used in conjunction with UDP. Unlike TCP, it does not support reliability mechanism. When a host wishes to send a media packet, it takes the media, puts it in a packet, adds any media-specific packet headers, adds the RTP header, and places it in a lower-layer payload. It is then sent into the network to another participant.
7. RTCP (Real-Time Control Protocol) provides support for real-time conference of groups of any size within Internet [RTP96] [AVC96]. This support includes source identification and support for gateways like audio and video. It offers QoS feedback from receivers to the multicast group as well as support for the synchronization of different media streams. Media senders and receivers periodically send RTCP packets to the same multicast group, but different ports, which is used to distribute RTP packets.

b) *Transport layer* provides a flow of data between the two end systems involved in the communication. There are two different kinds of protocols in this layer: TCP (Transmission Control Protocol) and UDP (User Data Protocol).

TCP provides a reliable connection oriented transport layer service. Before transferring any data, a connection is established between the two end systems. After this connection, TCP takes care of all the packets to assure that all of them arrive at the destination in order. Timeouts and retransmissions are implemented in order to provide connection-oriented service. TCP includes flow control and error detection - the packet rate can be adjusted depending on the level of load of the network, and corrupted packets are discarded and retransmitted.

On the other hand, UDP is a transport layer protocol that provides a connectionless service to higher layers. UDP does not assure that the packets will reach their destination. However, this lack of reliability makes UDP suitable for some applications, like real time audio or video stream, since the reliability mechanisms are built on top of UDP. Thus, the application can decide whether retransmission of packets is suitable, and a better control of the data flow is achieved from the application point of view.

c) *Network layer* provides communication between two systems attached to different network. It deals with the routing of packets through the path to the destination system.

The Internet Protocol (IP) belongs to the network layer. It provides a connectionless service between end systems, so there is no established connection between the end points that are communicating. Each packet that travels through the Internet is treated as an independent unit of data with out any relation to any other unit of data. All the intermediate systems between the two ends have to implement IP and the layers below it, but not necessary the higher layers. IP receives data from higher layers. It adds a header containing information related to the data received and passes it to the lower layer.

The most important service of IP is to send the packets to the proper next hop. All the routing information necessary for this purpose is contained in the IP header.

There are two IP formats, IPv4 and IPv6, which support 32-bit and 128-bit address separately. IPv6 supports resource allocation through labelling flows of packets. Hence, special flows like audio or video packets with low delay requirements can be treated in a different way than packets without real time data.

d) *Network access layer* includes Data-link layer and Physical layer.

Data link layer handles the access to the physical medium. It is concerned with the exchange of data between an end system and the network to which it is attached.

Physical layer is concerned with the mechanical and electrical characteristics to access the physical medium.

In the TCP/IP architecture, every layer uses the services of the lower layers and provides services to the higher layers. The user data is encapsulated by every layer, from the application layer to physical layer, adding its control information for handling the packet.

Since the purpose of this thesis is to realize value-added services in IP telephony, and since SIP is a signaling protocol for this purpose, we will focus on SIP and its approach to value-added services in this chapter.

## 3.2 SIP

SIP, the Session Initiation Protocol, is a simple signaling protocol for Internet conference and telephony [SIP99]. It is used to establish, modify and terminate multimedia sessions. It is currently under development within the IETF MMUSIC (Multiparty Multimedia Session Control) working group.

An IP telephony signaling protocol must accomplish a number of functions [AAP99][ATS98]:

- *Name translation and user location* involve the mapping between names of different levels of abstraction, e.g., a common name at a domain and a user name at a particular Internet host. This translation is necessary in order to determine the IP address of the host to exchange media with. This translation is more than just a simple table lookup. The translation can vary based on time of day, caller, or the status of the callee, among other criteria.
- *Feature negotiation* allows a group of end systems to agree on what media to exchange information and their respective parameters, such as encoding. The set and type of media need not be uniform within a call, as different point-to-point connections may involve different media and media parameters.
- *Call participant management* allows any call participant to invite new users into existing call and terminate connections with other participants. During the call, participants should be able to transfer and hold other participants.

- *Feature changes* make it possible to adjust the composition of media sessions during the course of a call, either because the participants require additional or reduced functionality or because of constraints imposed or removed by the addition or removal of call participants.

Therefore, SIP is a lightweight signaling protocol that can accomplish all these functions.

### 3.2.1 SIP Infrastructure

SIP is a client-server protocol. This means that the requests are generated by one entity - the client, and are sent to a receiving entity - the server to process them. In addition to the user agent acting as a client that initiates calls or acting as a server that accepts calls, SIP also provides the network server to proxy or redirect calls on behalf of the subscribers. Therefore, there are two components in a SIP system – the user agent and the network server.

As a call participant may either generate or receive requests, SIP-enabled user agents include a user agent client and user agent server. The user agent client is used to initiate a SIP request. The user agent server receives a request and responds to it based on human interaction or some other kind of input.

There exist two modes of operation in SIP when network servers are used: using a proxy server or using a redirect server.

- The proxy server forwards requests to the next hop server and returns responses on behalf of it. SIP requests can traverse many proxy servers, each of which receive a request and forwards it towards the next hop, and finally the request get to the user agent server. Because the proxy server takes care of the location of the user, the process is transparent to the client.
- The redirect server informs the client of the location of the next hop server, so that the client can contact it directly. Because the redirect server only locates the user and responds with the location, the client must issue a second invitation addressed to that new location.

### 3.2.2 Addressing and Naming

SIP uses these addresses as part of SIP URLs, such as sip: hwang@cs.mcgill.ca [AAP99]. This URL may be placed in a web page, so that clicking on the link initiates a call to that address.

Most users will be able to use their e-mail addresses as their published SIP addresses. E-mail addresses already offer a location-independent form of addressing, in that the host part does not have to designate a particular Internet host, but can be a domain, which is then resolved into one or more possible domain mail server hosts via Domain Name System (DNS) MX (mail exchange) records.

SIP has to be able to resolve name@domain to user@host. A user at a specific host will be derived through zero or more translations. A single externally visible address may well lead to a different host depending on time of day, media to be used, and any number of other factors.

### 3.2.3 SIP Requests and Responses

SIP is a client-server protocol. Clients issue requests and servers respond to them. Therefore, there are two types of messages - requests and responses in SIP.

1. *SIP requests*

The current version of SIP contains six types of requests. They are referred to as methods: INVITE, ACK, OPTIONS, REGISTER, CANCEL and BYE.

- INVITE method is used to ask for the presence of a certain party in a multimedia session and establishes a new connection. The negotiation of the parameters of the session is carried out using this method. In the middle of a call, it is also possible to change the current parameters of the media stream by sending a new INVITE request.
- ACK method is sent to acknowledge a new connection. It confirms that the client has received a final response to an INVITE. It can contain a session description describing the parameters of the media stream.
- OPTIONS method is used to get information about the capabilities of a server, but does not set up a connection. The server returns the methods that it supports.

- REGISTER method informs a SIP server about the current location of a user. This way, the user can be reached where he is logged in at that moment.
- BYE method is sent by a client to a server to terminate a connection between them. For two party calls, it terminates the call.
- CANCEL method terminates a search for a user. When a server is trying to reach a user, it can try several locations. When the user is reached, the rest of the searches can be cancelled. The CANCEL method cancels a pending request, but does not affect a completed request.

2. *SIP responses*

When a server receives a request, it sends back a response - which is identified by a number. The server keeps the client informed of the status of the call by means of responses. The responses can be of different kinds, and the type of response is identified by a status code, a 3-digit number. The first digit defines the class of the response. There are 6 main classes of response, which can be categorized by provisional and final responses.

- 1xx – Call progress, which is a provisional response, is followed by other responses indicating the final outcome of the request
- 2xx – Success, final response
- 3xx – Redirection, final response
- 4xx – Client Error, final response
- 5xx – Server Error, final response
- 6xx – Global failure, final response

### 3.2.4 SIP Message and Message Headers

SIP is a textual protocol based on Simple Mail Transfer Protocol (SMTP) and Hyper Text Transfer Protocol (HTTP). A SIP message is either a request from a client to a server, or a response from a server to a client [SIP99].

SIP-message = Request | Response

- Both request and response messages use the generic-message format of RFC 882 for transferring entities. Both types of messages consist of a start-line, one or more header fields, an empty line indicating the end of the header fields, and an optional message-body. The following is the formula of a SIP message [SIP99]:

  Generic-message = Start-line
  Message-header
  CRLF (empty line)
  [Message-body]

  Start-line = Request-Line | Status-Line

  Message-header = (general-header | request-header | response-header | entity-header)

- The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with the CRLF. Each element is separated by SP character.

  Request-Line = Method SP Request-URI SP SIP-Version CRLF

  Method = "INVITE" | "ACK" | "OPTIONS" | "BYE" | "CANCEL" | "REGISTER"

  Example: INVITE sip:hwang@cs.mcgill.ca SIP/2.0

- The Status-Line consists of the protocol version, followed by a numeric Status-Code and its associated textual phrase, with each element separated by SP character.

  Status-Line = SIP-version SP Status-Code SP Reason-Phrase CRLF

  Status-Code = 1xx | 2xx | 3xx | 4xx | 5xx | 6xx

  Reason-Phrase = <TEXT-UTF8, excluding CR, LF>

  Example: SIP/2.0 200 OK

- General-header = Accept | Accept-Encoding | Accept-Language | Call-ID | Contact | Cseq | Date | Encryption | Expires | From | Record-Route | Timestamp | To | Via

  Request-header = Authorization | Contact | Hide | Max-Forward | Organization | Priority | Proxy-Authorization | Proxy-Require | Route | Require | Response-Key | Subject | User-Agent

Response-header = Allow | Proxy-Authenticate | Retry-After | Server | Unsupported | Warning | WWW-Authenticate

Entity-header = Content-Encoding | Content-Length | Content-Type

Some of the most fundamental headers are shortly explained as follows:

| | |
|---|---|
| Call-ID | Uniquely identifies a particular invitation or all registrations of a client. |
| Contact | Contains locations that are used in different purposes depending on the message. |
| Content-Length | Indicates the message body length in bytes. |
| Content-Type | Indicates the media type of the message body. |
| Cseq | Command Sequence uniquely identifies a request within a Call-ID. |
| From | Indicates the initiator of the request. |
| Require | Used by clients to tell the user agent server about options that the client expects the server to support in order to properly process the request. |
| Subject | Indicates the nature of the call. |
| To | Specifies the recipient of the request. |
| Via | Indicates the path taken by the request so far. |

### 3.2.5 SIP Call Extensions and Extension Headers

SIP call extensions are provided by Internet Engineering Task Force (IETF) to facilitate the multi-party call and other IP telephony services. While using the extensions, the client must include the org.ieft.sip.call extension name in a *Require* header [CCS98]. The SIP call control service has the following extension headers:

Extension headers = *Also* | *Replaces* | *Accept-Location* | *Call-Disposition* | *Requested-By*

*Also* and *Replaces* are both request and response headers. *Accept-Location*, *Call-Disposition* and *Requested-By* are request headers.

- The *Also* request and response header advises the recipient to issue INVITE requests to the addresses listed. Each of these invitations should contain a Requested-By header that contains the From field of the message containing the Also field. The also header only be processed by the calling of called user agent, not by any intermediate proxy of redirect servers
- The *Replaces* request and response header is analogous to the Also header except that it asks the recipient to issue a BYE to the addresses listed, with the same Call-ID as the request containing the Replaces header.
- The *Accept-Location* request header allows the caller to provide hints to proxy and redirect servers.
- The *Call-Disposition* request header field allows the client to indicate how the sever is to handle the call.
- The *Requested-By* request header is only used in requests triggered by Also or Replaces. It contains the URI of the entity that issued the request containing the Also header.

### 3.3 SIP Approach to IP Telephony Services

The architectural model of Internet telephony is rather different from that of the traditional telephone network, because all signaling and media flow go over an IP-based network, either the public Internet or various Intranets instead of circuit switched network [IINS99]. In the traditional telephone architecture, nodes can generally only communicate with those other nodes to which they are directly connected. IP-based networks, on the other hand, present the appearance at the network level that any machine can communicate directly with any other.

The SIP infrastructure transforms the locations at which many services are performed. In general, end systems are assumed to be much more intelligent than those in the traditional telephone model; thus, many services that traditionally had to reside within the network can be moved out to the user agents, without requiring any explicit support for them within the network. Other services can be performed by widely separated specialized servers, which can proxy or redirect the calls on behalf of the subscribers.

### 3.3.1 SIP Server Basic Operations

A server can act as a proxy server, a redirect server, or both [SUA98]. It may be appropriate to proxy some users, while redirecting other users. Server process incoming calls (INVITE) in several stages:

1. If the request is INVITE or OPTIONS, check if the name represents a local alias for either a single user of a mailing list through location server.
2. If the name is a list, the call handling depends on the value of the Call-Disposition header.
3. If the request is for an existing Call-ID, handle it as designated for the call and skip the next step.
4. Check for global handling directives, which are specific to the caller or callee.
5. If the call has not been handled by the previous step or the user location has been left for the server to determine, find the location of the user by invoking a default user location script.
6. If the script or general server configuration called for proxy, issue the necessary commands.
7. If the user does not respond, the server MAY generate a MIME email message with the call request.

As previously mentioned, there are two different ways – proxy and redirect - for handling an incoming SIP request at a SIP server. The following is the function of a proxy server [SFIT98]:

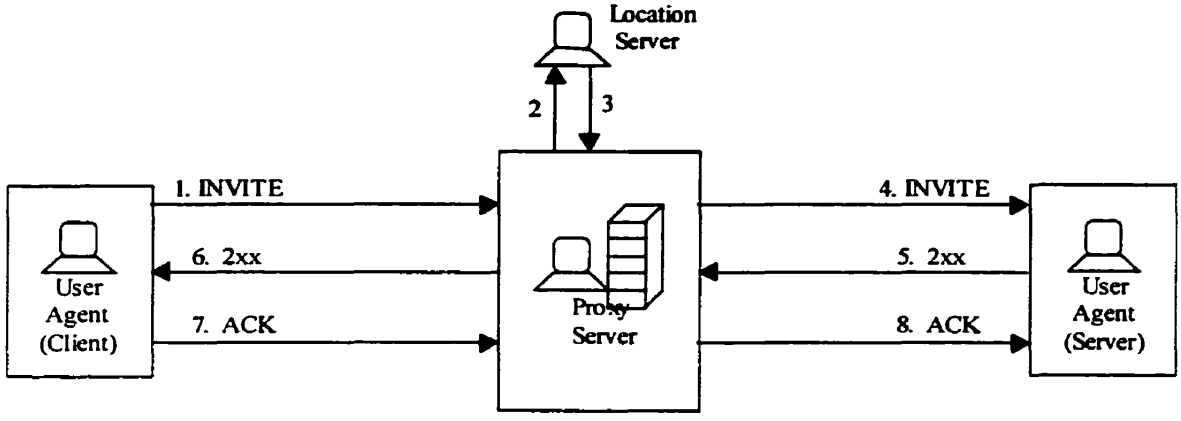



Figure 6: SIP Proxy Server Basic Operations

1. A SIP client first gets the address of a new participant of the form name@domain. The client then tries to translate this domain to an IP address where a server may be found. This translation is done by trying DNS Service (SRV) record, MX Canonical Name (CNAME), and finally Address (A) records. Once the server's IP address is found, the client sends it an INVITE message enclosed in either UDP or TCP.
2. The server that receives the message is not likely to be the user agent server where the user is actually located - it may be a proxy server. The proxy server queries the location server to get the address of the next server that the message should be sent.
3. The location server returns the address of the next hop proxy server or user agent.
4. Proxy server forwards the INVITE request to the address given by the location server. A via header traces the progress of the invitation from server to server, allows the responses to find their way back and helps servers to detect loops.
5. The user agent server accepts the request, and returns an acceptance to the proxy server, by a 2xx response.
6. The proxy server sends a successful response (2xx) to the user agent client (caller).
7. The user agent client confirms the response with an ACK request.
8. The proxy server forwards the ACK to the user agent server.

The redirect server performs differently from the proxy server as follows:

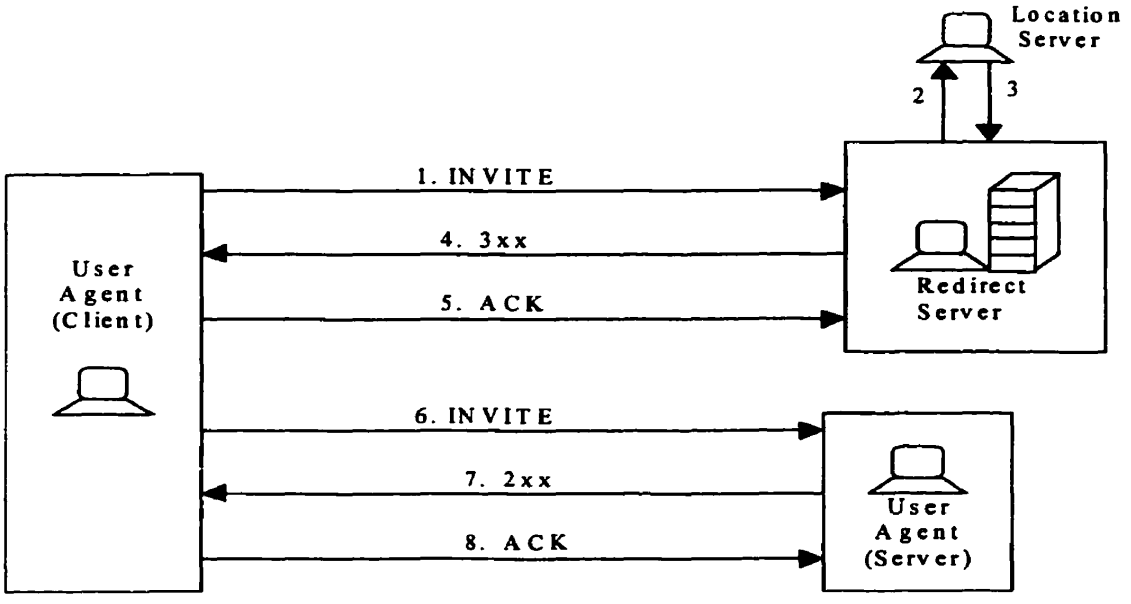



Figure 7: SIP Redirect Server Basic Operations

1. A SIP client first gets the address of a new participant of the form name@domain. The client then tries to translate this domain to an IP address where a server may be found. Once the server's IP address has been found, the client sends it an INVITE message enclosed in either UDP or TCP.
2. The redirect server queries the location server to get the address of the next server that the message should be sent.
3. The location server returns the address.
4. The redirect server responds to the INVITE request with 3xx response, telling the user agent client to contact the user agent server directly.
5. The user agent client acknowledges the 3xx response with an ACK request to the redirect server.
6. The user agent client issues a new INVITE request with the same Call-ID but a higher Cseq number to the user agent server – the address of the user agent server is given by the redirect server in step 5.
7. The user agent server sends a call successful response (2xx) to the user agent client.
8. The user agent client sends ACK to indicate the call establishment.

### 3.3.2 SIP User Agent Basic Operations

After receiving an incoming call, the SIP user agent server will operate as follows:

1. Invoke any per-user script.
2. The script may refuse, accept or defer decision on the call. If a decision is deferred, the user agent should return a 1xx response.
3. If the script does not decide, the user is asked via a use interface element how to handle the call.
4. The user agent should be configured with a timeout interval. If the user does not answer the call within a preset time interval, the user agent may log the call to a file and send an e-mail message to the owner. It then returns status 408 (Request Timeout).

For outgoing call, the caller must be able to choose whether to initiate a call that uses multicast or unicast for media distribution, regardless of the number of immediate callees. The user should be able to choose his or her preference of the call being handled by the server in Call-Disposition header.

The following is the basic operation of a user agent client on outgoing calls:

1. Remove any locations that the user has defined as being undesirable.
2. Remove any URLs that require a priority higher than the current call priority.
3. If the user has indicated he wants to confirm redirected calls, he should be presented with an ordered list of URLs and be allowed to remove any of the locations from further consideration.
4. Then, SIP URLs are tried in order or in parallel.
5. For non-SIP URLs, the user is presented with a pop-up stating that the callee was not reachable via an Internet call, but left alternate instructions.

### 3.3.3 SIP Approach to WIN Services

Just as mentioned before, WIN services include *subscriber services* and *network services*. Now, we give a brief introduction of SIP approach to those services.

i) The ***subscriber services*** include AIN, BIN and CIN services.

1. AIN are originating services, which are triggered at the originating part of the call. AIN services can be provided by the end system - SIP user agent client.
   - Customer Control Access (CCA) can be provided by implementing some database file in the end system and providing a GUI (Graphics User Interface) to the customer.
   - Bulk Number Translation (BNT), Private Number Plan (PNP), Outgoing Call Allowance (OCA), and outgoing Call Restriction (OCR) can be provided by checking some database files, and translating or screening according to that file in the end system.

2. BIN are terminating services, which are triggered at the terminating part of the call. BIN services also can be provided by the end system - SIP user agent server.
   - Selective Call Acceptance (SCA) and Selective Call Rejection (SCR) can be provided by checking some screening list file in the end system, then accepting (200 OK response) or rejecting (403 forbidden response) the call according to the checking result.
   - Selective Call Forwarding (SCF) can be provided in two steps: First, the end system should compare the caller address with the screening list. If that address is on the screening list, the call is forwarded to a specific address; otherwise, the call is forwarded to another default address. Second, call forwarding is done by SIP user agent server through a rejecting (403 forbidden response) with an Also header indicating the new address the caller should try.

     SIP proxy or redirect server also can provide SCF service. Instead of user agent, SIP proxy server store the screen list for every end user, and forward the calls to the new address on half of that user. The redirect server will return a 302 response to the caller with the new address in the Contact header.

3. CIN are transferring services, which are triggered by call transfer conditions. The CIN service currently offered in WIN is Flexible Call Forwarding (FCF). FCF is a call

transferring service that allows subscribers to forward calls to various destinations based on a subscriber-defined forwarding schedule.

- The forwarding services are activated under the following circumstances: Inactive or does not answer (TRN), Unconditional Call Transfer (CTR), Call Transfer on Busy (TRB). The forwarding destinations are used in forwarding schedules defined by the subscriber. Forwarding schedules consist of time periods that are assigned destination numbers. The FCF subscriber can define two types of schedules: weekday schedule and date schedule, date schedule takes precedence over the weekday schedule.

- FCF can be provided by proxy server or end system. If FCF is provided by end system, the end system should store a script of forwarding schedule. According to the circumstances (TRN, CTR, or TRB) and the forwarding script, end system can forward calls by sending a 4xx response with the new address in Also header to the caller, then the caller will INVITE the new address.

- If FCF is provided by proxy server, the proxy server should store a script of forwarding schedule. According to the forwarding script and the response it received from the callee (408 request timeout, 464 busy here), the proxy server can forward the call to the new address.

ii) The ***Network Services*** include Toll Free Calling and Wireless Virtual Private Network.

- Toll Free Calling include two core features: one number (ONE) and reverse charging (REVC). Charging is out of the scope of SIP protocol. ONE allows a subscriber with two or more terminating lines to have a single telephone number. The subscriber can specify which calls are to be terminated on which terminating lines based on the area from which the calls originate. This service can be achieved through the use of a proxy or redirect server. User script of how the call should be handled based on calling address is stored in Proxy or Redirect server.

When a call is coming, the proxy server will run the user script, find the forwarding address according to the caller area, and forward the call. On the other hand, the redirect server redirects the call by sending 302 response to the caller with the new address in the Contact header.

- The Wireless Virtual Private Network (WVPN) is not a single service but a collection of services that allow operators to provide subscribers with the benefit of private network numbering plans. The WVPN consists of the BNT and PNP core services, and can optionally include the OCA, OCR, SCA, SCF and SCR services. Therefore, it can be realised by combining the functionality of those services provided by proxy server, redirect server, and user agent.

### 3.4 Conclusion

SIP is a signaling protocol for multimedia transmission and IP telephony services. But IP telephony is still immature, it is most likely that there will be need for additional signaling capabilities in the future.

Although SIP can support most WIN services, it does not provide enough flexibility and efficiency. For example, it is true that AIN services can be provided by end systems, but every end system should support all service logic and data for all subscribers because customers may migrate to different end systems at different time. This requires intelligent end systems to have very large memory, which is not quite feasible.

It seems that SIP does not provide enough support for IP telephony, especially at the current time. Mobile agent technology together with SIP may provide a better solution for IP telephony services.

# Chapter 4

# Mobile Agents and Their Applications

In this chapter, we first introduce Mobile Agent (MA) concept and platform, then we describe the state of art MA applications, and their impact on classical telephony.

## 4.1 Mobile Agent Concept and Platform

An agent is a self-contained software element responsible for performing part of a programmatic process. It acts typically on behalf of a user or a process enabling task automation. Agents operate autonomously and may communicate with the user, system resources and other agents as required to perform other task. Moreover, more advanced agents may cooperate with other agents to carry out tasks beyond the capability of a single agent. Finally, as transportable or even active objects, they may move from one system to another to access remote resources or to meet and cooperate with other agents. Therefore, agents can be characterized by the following attributes: agent intelligence, asynchronous operation, agent communication, agent cooperation, and agent mobility, which include remote execution and migration.

### 4.1.1 Definition of Mobile Agent

A mobile agent is an autonomous and identifiable computer program that can migrate between physical locations within the network and act on behalf of a person or an organization [IMA98]. There are two kinds of mobile agent functionality [IOD96]:

- *Remote execution*: a mobile agent can be transferred to a remote system for local activation and execution.
- *Migration*: during its execution, a mobile agent may move from node to node in order to progressively accomplish its task. In other words, agents are capable of suspending their execution, transporting themselves to another node in the network, and resuming execution from the point at which they were suspended. In addition, agents may

launch new agents during their journey in order to distribute the tasks more efficiently. The migration of software processes could be regarded as an extension to the concept of remote execution. Apart from its code and data, a migration agent also has an explicit execution state, allowing it to suspend execution at a specific state at one node, and resume it after migrating to another node.

### 4.1.2 Advantages of using Mobile Agents

Mobile Agent has two main advantages [IA96]: *reduction of network traffic* and *asynchronous interaction.* In principle, the following advantages of mobile agent technology can be identified [IMA98].

- *Reduction of network traffic*: Versus the client server approach, which can generate a high level of network traffic and can be susceptible to congestion delay, the mobile agent paradigm proposes bringing the requesting client closer to the source and hence reducing the network traffic.
- *Asynchronous and cooperative processing of task*: The mobile agent technology proposes to treat the network as multiple agent friendly environments and the agents as programmatic entities that move from location to location, performing tasks for users. Agents can be function independent of each other and cooperate to solve problems.
- *Robustness*: By moving the required logic and data, a reduction in dependence on network availability is achieved.
- *Customization and configuration of services*: In the light of an electronic market place, agent technology allows to instantly provide new services by customization or configuration of existing services. Agents can act as service adapters, which can be easily installed.
- *Instant service usage and active trading*: Mobile agents can provide flexible services by sending the service agents dynamically and on demand to customers.
- *Decentralization of network management*: Mobile agents are distributed to the managed systems to enhance network management.

- *Intelligent communications*: Agents provide the basis for advanced communications. They support the configuration of a user's communications environment, where they perform communication control on behalf of the end users.
- *Information retrieval and support of dynamic information types*.

### 4.1.3 Requirements for MA System Platform

In order to support mobile agent execution, the MA system platform should meet the following ten requirements: security, portability, mobility, communication, resource management, resource discovery, identification, life-cycle control, transport, and data management [MAS98].

- *Security*: MA platform guarantees certain security levels, so that the agent is protected from the host, the host is protected from the agent, agents are protected from each other, and hosts are protected from each other.
- *Portability*: Platforms are generally heterogeneous; therefore, all current MA systems must deal with porting agent code and data to work on multiple platforms. After Java, the Java virtual machine represents a better compromise because platform-neutral bytecode can be compiled just in time to alleviate the performance problem.
- *Mobility*: Agent mobility mechanisms include remote execution, migration, cloning, and programming language support etc.
- *Communication*: Agents can communicate with each other as well as with the system platform.
- *Resource management*: Agents are executing programs that may require access to low-level system resources and higher-level resources.
- *Resource discovery*: An agent is far more efficient if it can dynamically discover the resource it needs to accomplish its tasks. Resource discovery covers an area complementary to resource management.
- *Identification*: Agents must be identified uniquely in the environment in with they operate. The purpose of an identification scheme is to generate unique identifiers and establish some infrastructure to allow convenient and accurate access to the agents that carry them.

- *Life-cycle control*: Life-cycle control is used to provide ways in which the agent may be created, started, stopped, duplicated, or terminated. Control also covers issues such as how the activities during the MA life cycle can be coordinated.
- *Transport*: Transport allows agents to migrate from one place to another place.
- *Data management*: An MA can carry data with it, and it also needs to store itself in a persistent form for fault tolerance or other purposes.

### 4.1.4 MASIF: Mobile Agent System Interoperability Facility

The mobile agent technology is suitable for the improvement of distributed object systems. However, an integrated approach is more desirable, combining the benefits of both client/server and mobile agent technology, and eliminating the problems that arise if one of these techniques is used as a standalone solution. The following chart shows this integrated approach by means of a distributed agent environment (DAE) built on top of a CORBA distributed processing environment (DPE). The DPE will be used as a communication channel between the various agencies that are part of the DAE.

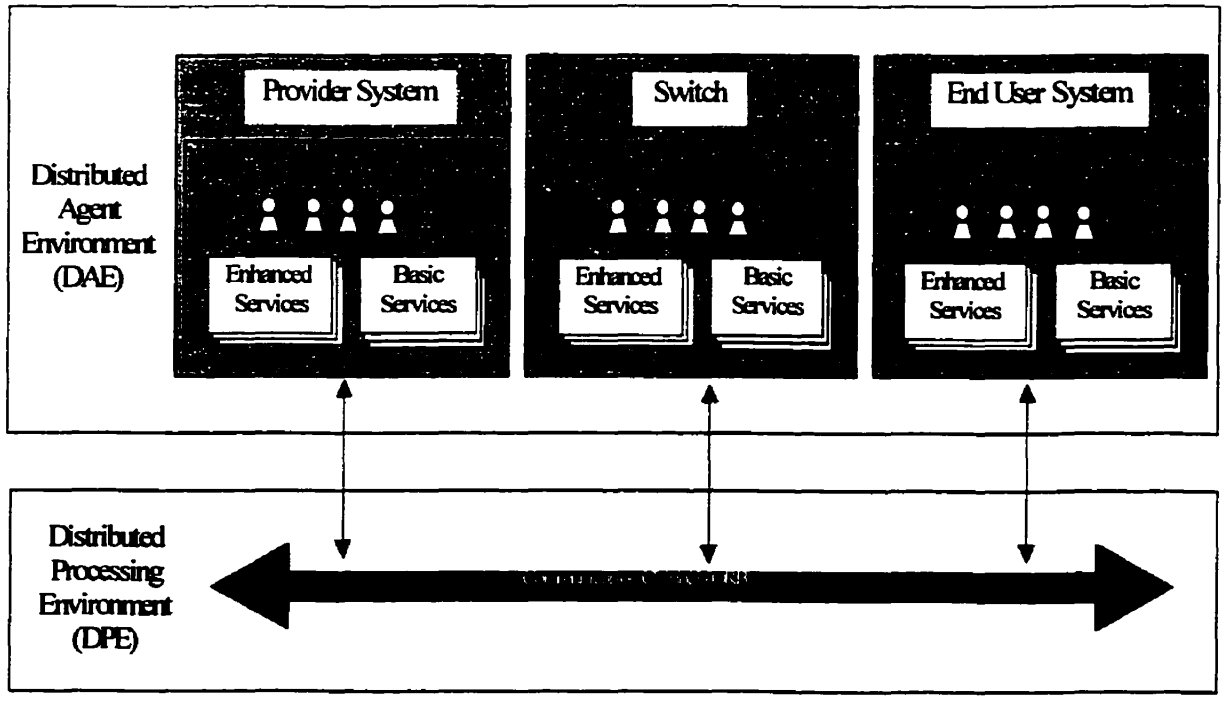



Figure 8: DAE and DPE in MASIF

MASIF is an OMG specification [IMA98]. The idea behind the MASIF standard is to achieve a certain degree of interoperability between mobile agent platforms by different manufacturers without requiring radical platform modifications.

MASIF has adopted the concepts of places and regions used by various existing agent platforms. The actual runtime environments for mobile agents are called *agencies*, which collectively form the DAE. A *place* groups the functionality within an agency, encapsulating certain capabilities and restrictions for visiting agents. A *region* facilitates platform management by specifying sets of agencies that belong to a single authority. Two interfaces are specified by the MASIF standard: the MAFAgentSystem interface provides operations for the management and transfer of agents, whereas the MAFFinder interface supports the localization of agents, agencies, and places in the scope of a region of the whole environment, respectively.

Apart from the agencies, additional tools are required. An *agent creation environment* enables the plug-in and play composition of mobile agents out of reusable functional building blocks. An *agent testing environment* allows the simulation of the whole distributed environment by means of a single agency. A *graphical agent management tool* enables the monitoring and control of agents and agencies in the scope of one or more regions.

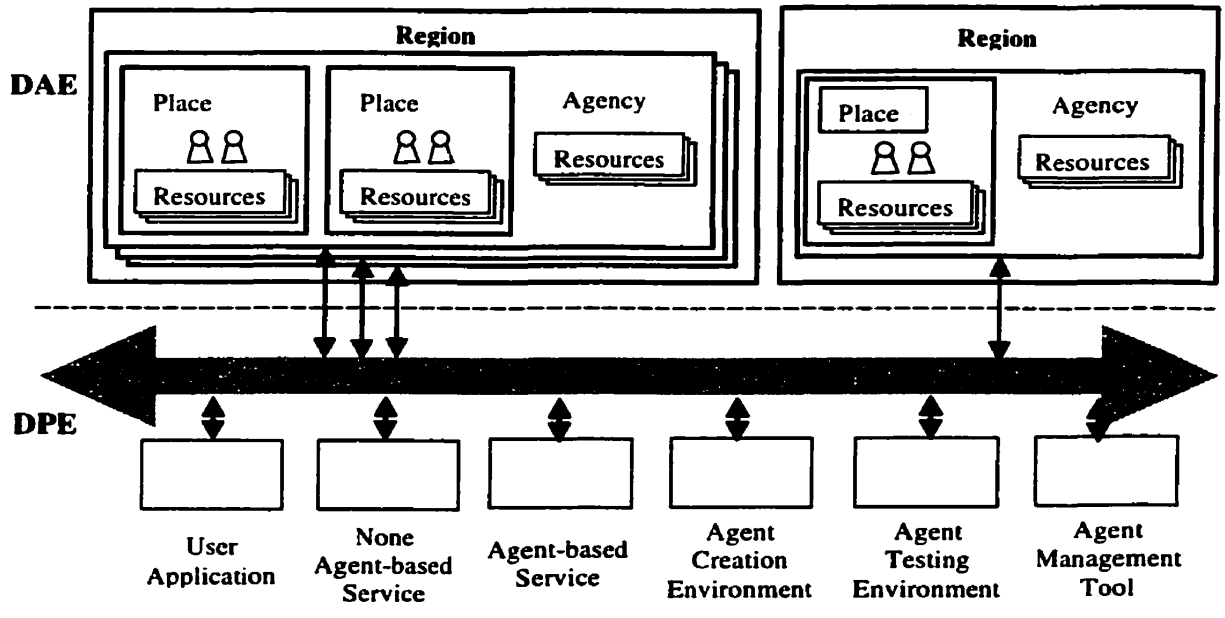



Figure 9: MASIF Regions, Agencies, Places, and Additional Tools

## 4.2 Mobile Agent Applications

There are many MA applications in network management and classical telephony. Now we introduce the MA applications and their impact on network management, service architecture, classical IN, and Broadband IN separately.

### 4.2.1 Mobile Agents Impact on Network Management

Today's telecommunications management is based on centralized intelligence in management systems. The managing systems query the management information located in numerous distributed managed systems in the telecommunication network. Therefore, the central manager can be regarded as the bottleneck and the reason for large communication overhead [NM98].

Mobile service agents offer a possible solution. We can move mobile management service agents into the managed systems, thereby increasing the autonomy of the management agents. This approach features the delegation of management activities from the managing system to the managed systems. The managed systems must provide the necessary agent execution support [IOD96].

The execution of dynamically downloaded mobile service agents containing management application scripts, enables network elements to perform specific management tasks independent from their operation systems. Different mobile agents could be sent to different management agents in accord with the desired management task. Cooperation between mobile agents located at different managed systems or the managing system may be possible, if the agent platform supports cooperation [MSA98].

Finally, a managing system, i.e. an operations system, could generate a mobile service agent performing specific management tasks autonomously and purposefully at specific or multiple managed systems, e.g. collecting, downloading or modifying specific distributed managed objects in a coordinated way. This type of mobile agent is able to collect information from different managed systems and to return it to the managing system.

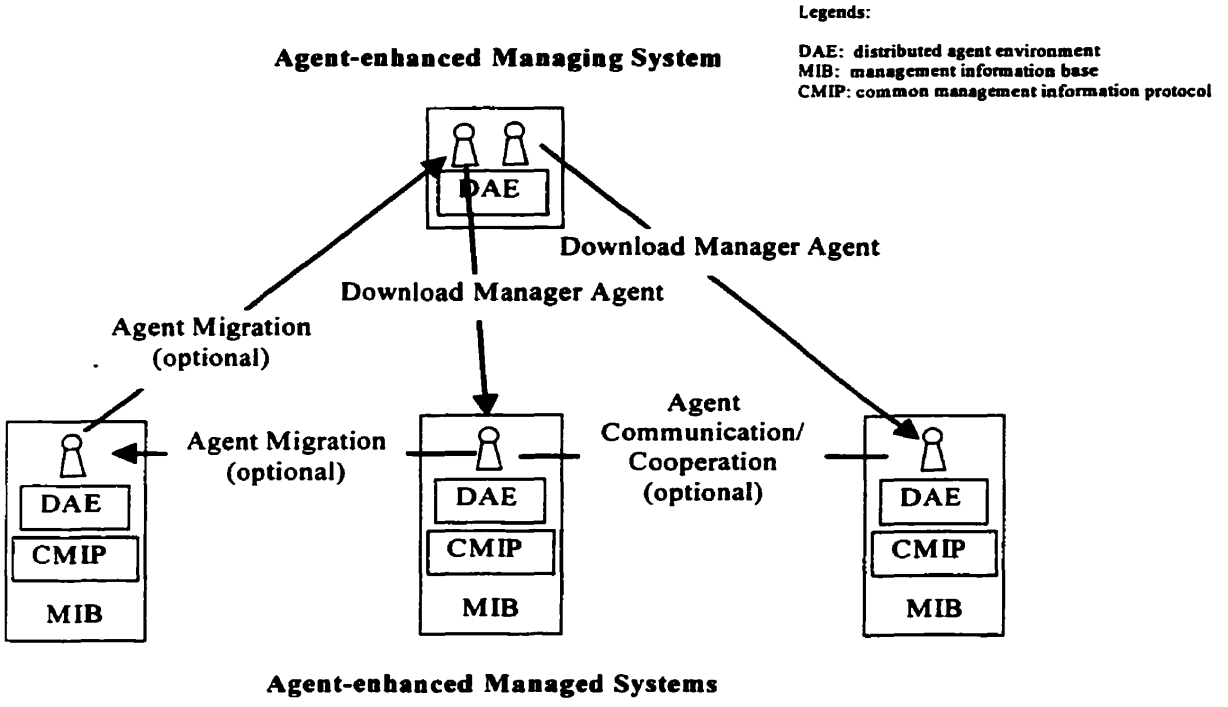



Figure 10: Mobile Agent Application in Network Management

### 4.2.2 Mobile Agents Impact on Service Architectures

Except for network management, mobile agents also have great impact on service architecture as follows:

*Customization and Configuration of Services*: Due to the mobility property of agents, existing services can be customized and new services can be established by configuring existing ones. To do this the remote execution style of mobility is sufficient - agents are moved to a node and remain there until they terminate. In order to provide a customized service, an enhanced service provider simply has to develop an appropriate agent and transfer it to the corresponding server machine.

*Instant Service usage*: In order to interact with potential customers, a service provider has to create an appropriate service client, which travels to the customer. When receiving a service client for some service, the end-user can immediately use this service.

*Service on demand*: In contrast to the exchange of service requests between static clients and servers, mobile agents implementing the desired client functionality, carry this functionality to the server site, where client agent/server interaction will be handled

locally. On the other hand, mobile agents representing a server may also be transported to a remote client. This enables customized telecommunications services to be provided instantly and exactly at the location where the service intelligence is needed, which means intelligence on demand [IOD96].

### 4.2.3 Mobile Agents Impact on Classical IN

Generally, mobile agents enable to dynamically place control and management software processes at the most appropriate locations within the telecommunication environment. This will have significant impacts on the architecture and the related signalling protocols of the existing telecommunication systems.

A centralized SCP and the necessary usage of INAP (Intelligent Network Access Protocol) represent potential bottlenecks when the number of IN services increases [IOD96]. Therefore, a really distributed realization of IN services will become important, where service intelligence should be installed dynamically and as close to the customer as possible.

Mobile agents used in intelligent network would be responsible for the dynamic downloading of customized service scripts (i.e., dynamic migration of service intelligence form the SCP to SSP), ultimately allowing the provision of service intelligence on demand [TIOD96].

With agent-based approaches, services can be provided instantly, customized, and distributed. Therefore, a mobile agent contains all necessary control information (i.e. the service logic of a particular telecommunication or management service), instead of the corresponding end systems and/or switches within the network. This means that corresponding agent execution environments have to be provided by the potential end user systems and the switches within the network, in order to perform the execution of agents and thus the realization of the intended services.

An MA-based intelligent network has following advantages [MAS98]:

- Enable the provision of flexible software solutions, where IN service software is partitioned into mobile service agents to realize dedicated functionalities (e.g. IN service features).

- Automate service deployment and provision and thus reduce the required efforts and time for the installation, operation and maintenance of IN services.
- Enable on demand provision of customized IN services by dynamic service agent downloading from the service provider system to the network nodes and/or end user systems.
- Allow for a decentralized realization of IN services, by means of bringing the service control agents directly onto the resources to be controlled, i.e. the switching nodes.
- Reduce or eliminate the influence of SS7 signaling network faults during IN service processing.

Distributed agent environment can be provided to end user systems or intelligent nodes or both. Section 4.2.3.1 and section 4.2.3.2 describe these two situations separately.

#### 4.2.3.1 Mobile Agent in End User Systems

The following figure shows that the Distributed Agent Environment (DAE) is provided only to end user systems. In this situation, mobile agents will provide value-added services at the agent-enhanced end systems.

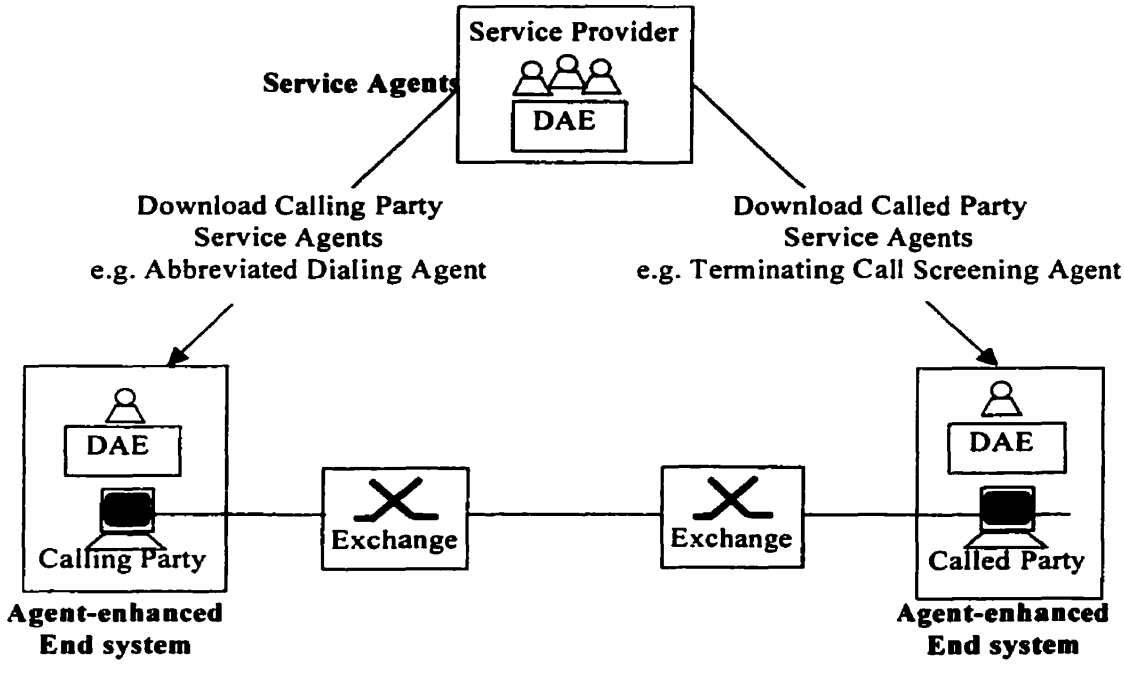



Figure 11: Distributed Agent Environment Provided to End User Systems

#### 4.2.3.2 Mobile Agent in Intelligent Network

Mobile service agents are able to find the best location inside the network in order to provide the services with minimum usage of the signaling network. IN service features are related to the calling party or to the called party; therefore, services can embody service features for the calling party, called party, or both. Once injected into the network, a mobile service agent heads autonomously for the serving switch of the subscriber acting as calling party or acting as called party. In the case of services embodying both calling party and called party features, two options exist: [MAS98]

- The service agent may clone itself in accordance with the number distributed users, and the children migrate to the appropriate service switches.
- The service agent may migrate to the centralized node in order to be globally available via the signaling network. This approach is also important for services related to called parties but which require global availability.

Actually, a mobile service agent does not necessarily remain at the same location (e.g. serving switch) for its whole life. Instead, it can have following usage:

- A mobile service agent, which has initially migrated to a centralized SCP, may decide during its life time to clone itself and send a child service agent to one or multiple SSPs from which most the service calls originate and there by reduce the load on the signaling network.
- A mobile service agent can support mobile users – a service agent may follow a mobile user by migrating to the next serving switch.
- A mobile service agent may move from a home network to a visited network in order to avoid inter-network signaling and/or for the provision of customized services to mobile users in foreign network.
- A mobile service agent can provide Virtual Home Environment (VHE), which provides roaming users with a unified service look and feel even when using foreign networks.

- A mobile agent can be used for pre-establishement of real-time information exchange, such as multimedia communication service. Here, a mobile agent will be used primarily for signaling and system configuration, i.e. the establishment of a real-time communication service. This means that an agent will be generated by the calling user's end system prior to the real communication session and sent to the destination user's end system in order to set up the required communication path.

The following picture depicts the architectural enhancement required to realize an MA based IN environment. Basically, each IN element provides an agency, which collectively form a distributed agent environment in which the service agents are able to roam. Thus service agents can easily migrate in this environment from the service provider system to the most appropriate network node.

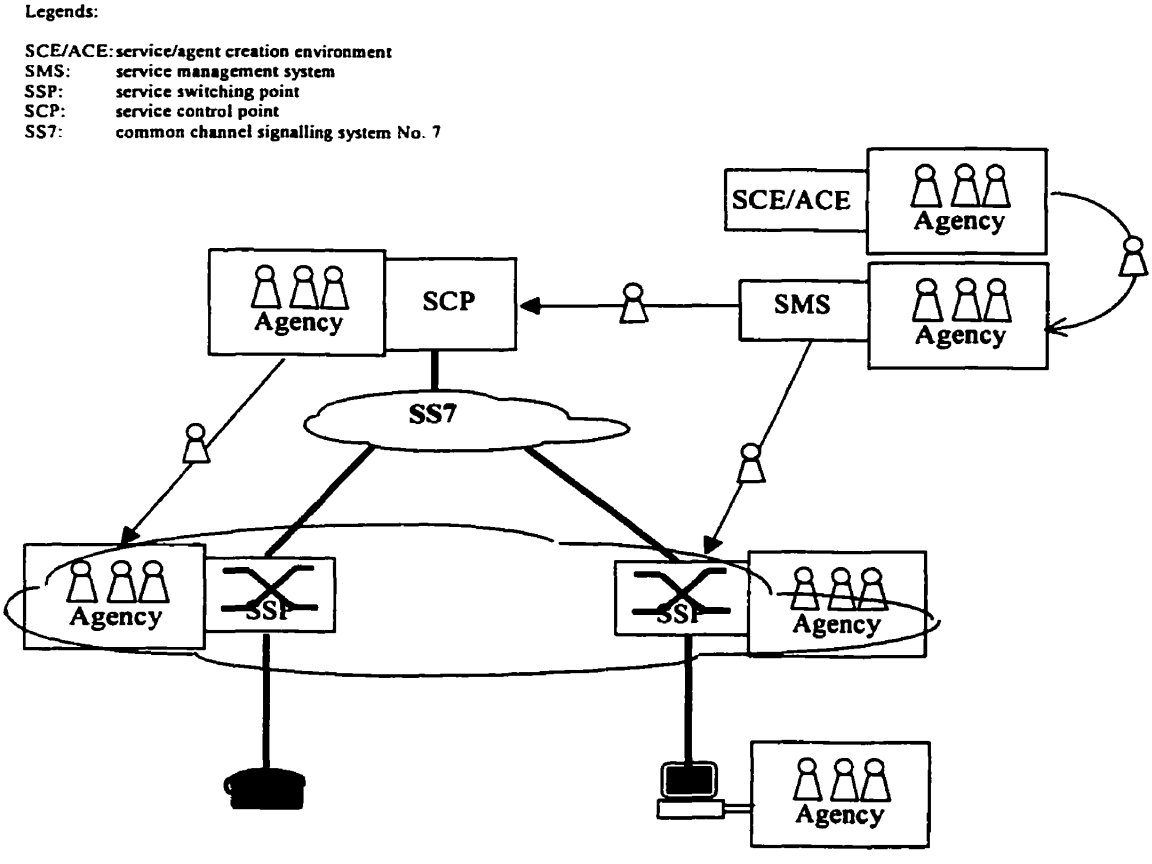



Figure 12: Agent-based IN Architecture

An agent-enhanced service management system (SMS) allows one to introduce the service agents into the MA-based IN infrastructure. It is a key component with respect to providing the service to potential customers and configuring service agents accordingly. After agent release the SMS keeps track of an agent's location and its status.

An agent-enhanced Service/Agent Creation Environment (SCE/ACE) allows one to develop appropriate IN service agents.

A mobile service agent comprises two parts in principle. The first part – the application part – is related to the provision of real service, i.e. contains service control logic and data that make use of external switch control interfaces. Furthermore, this part also includes functionalities for service subscription, customization and service logic maintenance. The second part – the core agent part – is devoted to the agent's nature of being an autonomous mobile entity. This means that this part comprises all functionalities required for agent life cycle management and mobility support.

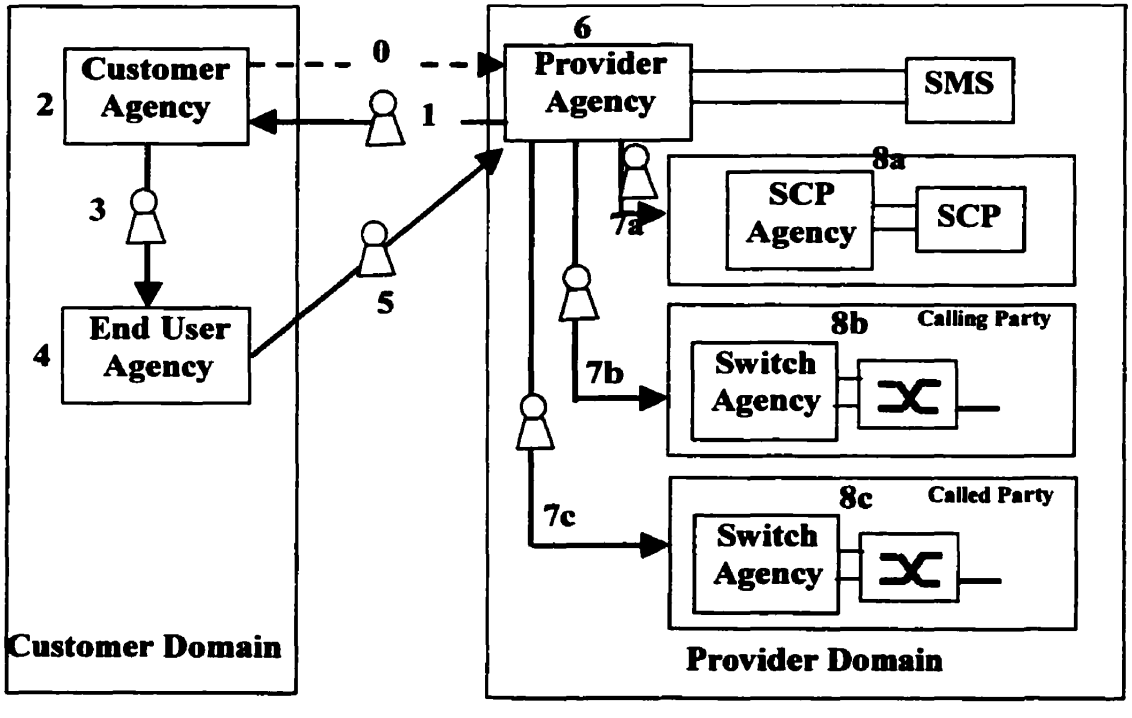



Figure 13: Agent based Application Scenarios

Three kinds of actors are involved in the agent based application scenarios [MAS98]: An *agent provider* who is accessible via the World Wide Web, a *customer* representing a company or organization, and various *end users*. Each actor requires access to an agency. Additional agencies are provided at the network elements (SCP and SSP).

The scenario is initiated by the customer accessing the provider agency via the WWW and asking for an IN service agent (0).

1. The provider sends the requested agent to the customer.
2. The customer pre-configures the mobile agent. The pre-configuration comprises the selection of end users who are able to use the agent, and the specification of various access restrictions concerning the selected end users.
3. Afterwards, the service agent is sent to the end users.
4. The end user supplies the mobile service agent with individual service logic configurations.
5. The service agent automatically migrates back to the provider before it executes its designed task.
6. The service provider does security checks to the modified agent.
7. If the security checks have been successful, the agents move to a specific network node. Three possibilities are taken into account:
   - Agents representing a global service migrate to the agency connected to the centralized SCP.
   - Agents realizing a called party service move to the agency at the called party switch.
   - Agents representing a calling party service move to the agency at the calling party switch.
8. After reaching their destination agency, the agents connect themselves to specific IN service adapters, which can either be realized by enhanced agency services or in turn by special agents. Finally, the service execution starts.

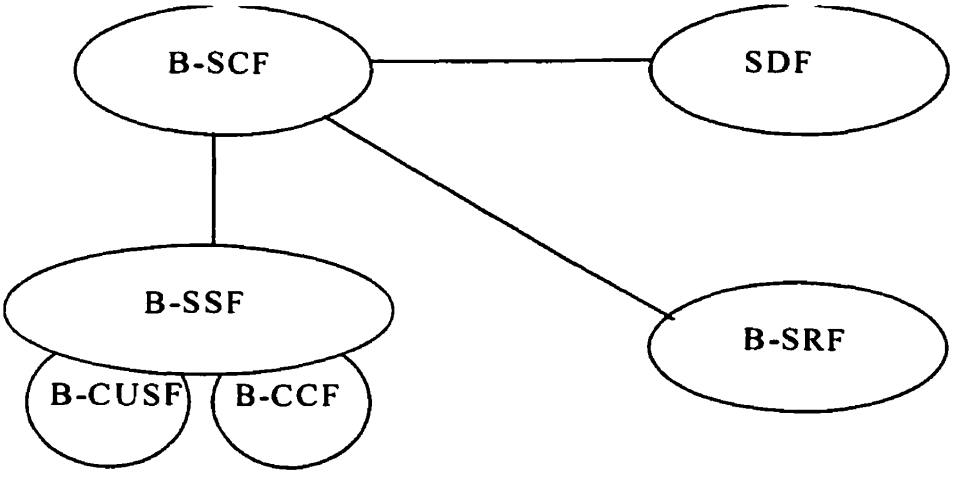
Mobile agents have a great potential to influence the design of future telecommunications, since they represent an alternative to the traditional client/server

interaction model. However, agent technology will probably not replace traditional client/server computing entirely. Rather mobile agents should be regarded as an "add on" to existing computing/service platforms, providing more flexibility for the realization of services within next generation telecommunication environments.

### 4.2.4 Mobile Agents Impact on Broadband IN

B-ISDN IN architecture provides a flexible solution to allow easy service provision and customization, without affecting the network structure and the applications inside the terminal [IW98].

IN provides a good solution also in overtaking some limitations, which affect at least the current version of B-ISDN signalling standards. While signalling considers only one connection within one call, IN can take care of grouping connections related to a single service instance, thus providing a coordinated handling of resources required by specific services. This leads to the enhancement of the role of the Service Switching Function (SSF) in the IN Functional model.




Legends:

| | |
|---|---|
| B-CCF: | Broadband Call Control Function |
| B-CUSF: | Broadband Call Unrelated Service Function |
| B-SSF: | Broadband Service Switching Function |
| B-SCF | Broadband Service Control Function |
| SDF: | Service Data Function |
| B-SRF: | Broadband Specialized Resource Function |

Figure 14: Broadband IN Function Model

#### 4.2.4.1 DOT and MAT Impact on Broadband IN Architecture

In the classical IN architecture, the relationships between functional entities like SSF, SCF and SRF are based on the standardized Intelligent Network Application Protocol (INAP). The main drawback is that the location of the software is tied to the network topology [IW98]. For example, if a new IN service is successfully enjoyed by the users, the number of queries to the SCP increases beyond the original planning, and mechanisms for controlling the traffic load are required.

IN flexibility can be significantly enhanced by the introduction of distributed object technology (DOT). IN is suitable to be considered as a distributed environment, where a clear functional separation between calls and services exists as part of the foundations of the IN paradigm.

The distributed processing environment (DPE) allows on one side to reuse as much as possible the software components developed until now, and on the other side to introduce new features according to the well known concepts of objects oriented programming: decomposition in reusable modules, inheritance, polymorphism, information hiding. The actual location of objects in the network is hidden by the DPE itself, which takes care of resolving object requests.

A further enhancement is given by modeling services as mobile service agents, which can be dynamically distributed in the network using Mobile Agent Technology (MAT). In particular Service Logic Programs (SLPs) and data can be spread where and when needed across the core telecommunication network as an alternative to the centralized classical IN. Decentralization of the service logic helps to achieve a better exploitation of the computational resources available to the network and to spread network-provided services to a higher number of users.

The distribution of services within the network can be extended to switching systems instead of being limited to SCPs. This can be achieved by incorporating a MA runtime environment based on a DPE within the nodes.

### 4.2.4.2 DOT/MAT Reference Architecture

The distributed Architecture [SNA98] is based on both Mobile Agent Technology and CORBA. The following figure is the basic reference model of the new distributed IN architecture.

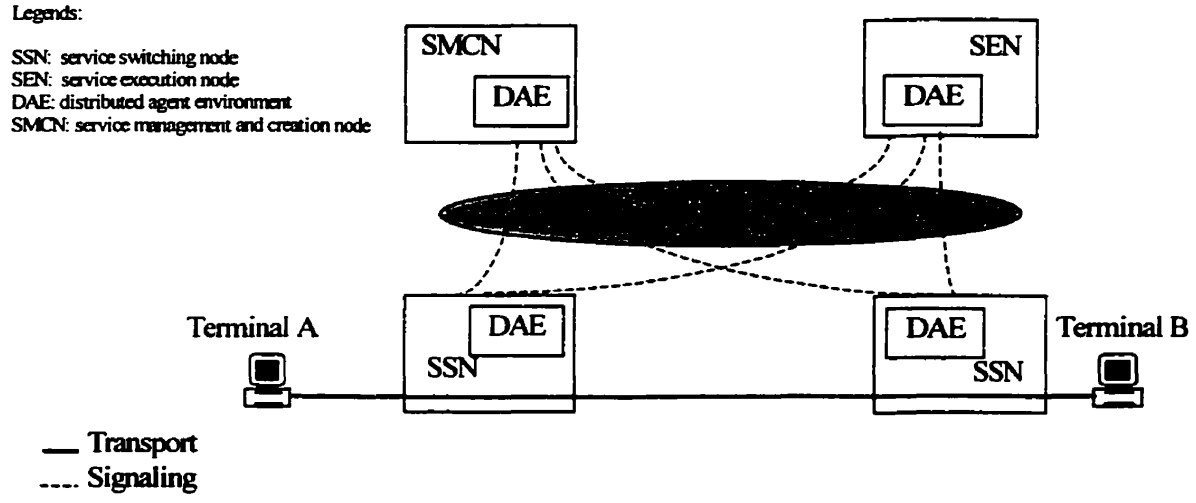



Figure 15: The Distributed IN Architecture

- Service Switching Node (SSN): is the device responsible for providing users with access to the network and allows access to the IN services. The SSN can be considered as an enhanced SSP. The difference is that in conventional IN systems the entire service logic is executed in the SCP while the use of MA-based Service Logic Programs allows download and execution of a part of the service to the SSN. In this way, a large number of signaling messages between SSN and Service Execution Node (SEN) are omitted.
- Service Execution Node (SEN): is the physical entity responsible for the proper operation of the Service Logic and Service Data as well as for providing Specialized Resources. In terms of the classical IN Physical Plane, the SEN can be considered as a unified Service Control Point (SCP) and Intelligent Peripheral (IP).
- Service Management and Creation Node (SMCN): is responsible for providing the management functions required for network as well as service operation. It also provides the environment for Service Creation and Testing. Apart from these

straightforward tasks, the SMCN plays the role of a permanent repository of the agents needed by all other physical element.

- Terminals: need to support for accessing Broadband networks. This means that they need to have special hardware (e.g. ATM cards) and firmware (e.g. UNI signaling stack). The idea of having Mobile Agent Technology and CORBA in the IN can be applied to all involved network physical entities with no exception for the terminals as well.

MA has following advantages in telecommunication service provision:

- *Faster handling of service requests*: once the agent/service is present in the SSP, call handling does not require the establishment of a transaction with a SCP and the consequent exchange of messages across the network.
- *More secure handling of services*: the influence of network faults is considerably reduced since the network is accessed only for the retrieval of the SLP.
- *Scalability*: centralized bottlenecks are overcome by uploading the service code to peripheral switches when needed, assuming that the SSPs are much higher in number with respect to the SCPs, which is indeed the case for the current IN. Dynamic SLP/SDT uploading allows IN services to be spread across the network, thus being able to satisfy a higher demand.
- *Reduction of traffic on the signaling network*: if services are moved closer to the customers, messages related to service control are handled locally.

The integration of IN and B-ISDN provides flexible service control opportunities and significantly enhances the broadband network. In particular, the introduction of a powerful object-oriented session model allows overcoming the limitations existing in the current signalling systems by means of IN. These capabilities are further enriched by incorporating DOT/MAT within the IN context. Actually, broadband IN can serve as a transitional architecture towards a public network based on distributed objects and mobile agents. Multimedia services in such a network can be efficiently provided via the transformation of network elements into service execution nodes.

# Chapter 5

# Conceptual MA Architecture for SIP IP Telephony

Until now, we have introduced Value Added Services (VAS) in the context of Intelligent Networks (IN). We also have given the description of the SIP protocol, its approach to VAS in IP telephony, and its shortcomings. In the last chapter, we introduced mobile agents and their impact on classical telephony. Actually, mobile service agents also can provide great advantages in IP telephony.

## 5.1 Introduction

Just as mentioned before, today distributed object technology has gained considerable acceptance in telecommunications systems. In addition, mobile agent technology is currently gaining momentum in the telecommunications domain [GRA98]. The recent OMG work on a Mobile Agent System Interoperability Facility (MASIF) specification can be regarded as a milestone on the road toward unified distributed mobile object middleware, which enables technology and location transparent interactions between static and mobile objects. There are already some projects and results of Mobile Agent (MA) applications in intelligent networks (IN), such as Mobile Agent enviRonments in Intelligent NEtworks (MARINE) project [GRA98] [IW98] by GMD FOKUS, IKV++ and Italtel. Those projects and the related results have shown the great advantages of MA application in classical telephony, such as reduction of network traffic, asynchronous and cooperative processing of task, robustness, and service on demand, etc.

The purpose of our research is to use MA to realize VAS in the IP telephony environment. A major difference between classical telephony and IP telephony is that, in the latter, call and connection controls are not required to be performed by the network. They may be performed from the end-to-end, in the servers or user agents located at the edges of the network, so value-added services can be provided purely from end to end, without any intervention from network entities.

Currently, there are two protocols that address the issue of IP telephony, one is International Telecommunication Union – Telecommunication Standardization Sector

(ITU-T) H.323 and the other is the Internet Engineering Task Force (IETF) Session Initiation Protocol (SIP).

Carleton University has been doing research on MA application in IP telephony, and has proposed an MA based advanced service architecture for H.323 Internet protocol telephony. In this architecture, agencies are provided to the end user systems and gatekeepers, so the end systems can take an active part in service provision. The main advantage of this architecture is flexible service creation and provision. This architecture is only related to service provision and execution in one enterprise LAN, but is not related to service mobility between different domains.

Adapted from Carleton service architecture for IP telephony, we will present a conceptual MA architecture for SIP, and show that this architecture not only supports flexible and instant service provision to one domain, but also supports service mobility and on demand service provision to different domains.

The rest of this chapter is organized as follows. Section 5.2 summarizes the Carleton MA service architecture for H.323 IP telephony. Adapted from the Carleton architecture, section 5.3 proposes a conceptual MA service architecture for SIP IP telephony. Section 5.4 presents the service mobility issues in IP telephony, and illustrates how our conceptual MA architecture solves this problem. Finally, Section 5.5 evaluates our conceptual architecture against SIP service architecture and concludes this chapter.

## 5.2 Summary of Carleton MA Architecture for H.323 IP Telephony

In H.323, Gatekeepers and H.323 terminals are connected to the Enterprise LAN. User terminals join one gatekeeper's zone through gatekeeper discovery and the end point registration process.

In the Carleton Architecture, mobile agent platforms are introduced into the devices that are connected to the enterprise LAN [CARL99], so agencies that provide execution environments to the agents are attached to gatekeepers and terminals. H.323 supplementary services are realized by means of User Service Agents (USA) and Call Agents (CA). USAs are deployed to the end users, which provides an open, distributed and flexible service architecture.

In this architecture, each end user has one user service agent (USA). This USA contains service proxy objects of this end user, including originating service proxy object and terminating service proxy object. The USA also contains Code Repository URL where the implementation code of service object can be found. USA defines how a call will be processed - how to create a Call Agent (CA) to initiate or terminate a call, and to provide value added services in the mean time.

There are two situations where a USA will instantiate a CA to handle a call [CARL99]. One is when a caller wants to initiate a call using originating services. USA will create a CA by attracting implementation codes from Service Implementation Repository (SIR), then initiate that CA and hand the control over to that CA for originating service provision. The other is when an incoming call arrives at the gatekeeper and would like to trigger terminating services. This time, USA will create a CA by attracting implementation codes from Service Implementation Repository (SIR), then initiate that CA and hand the control over to that CA for terminating service provision.

The Carleton service architecture is good for open service creation and deployment. Supplementary services can be created by a Service Component Creator (SCC) using service components from a Service Component Repository (SCR) [CARL99]. An SCC is responsible for creating services that are made available to Enterprise Service Creator (ESC), and advertising its services on an LUS. The SCC and SCR bring opportunities for third party service creators and providers - ESC, make them able to compete in the service market and provide services to end users.

Service Management Unit (SMU) manages service subscription and update. It can be placed in the gatekeeper, or completely separated from the gatekeepers. If a service would be dynamically upgraded, the SMU would be involved.

### 5.3 Adaptation to SIP and Extensions

The Carleton Architecture separates the idea of service creator and service provider into two different levels. Level one is Service Component Creator (SCC) together with Service Component Repository (SCR) and LUS to create and advertise services for level two service providers. Level two is Enterprise Service Creator (ESC) together with

Service Implementation Repository (SIR) and Enterprise LUS to create and provide services for service subscribers and end users. Service Management Unit (SMU) is used to manage service subscription and update. SMU can discover LUS, and help the user subscribe to update their services.

Adapted from the Carleton Architecture, we first present the key players in our MA service architecture for SIP IP telephony in section 5.3.1. Then we describe the motivation of adapting USA and CA to ASA and BSA in section 5.3.2. Finally, we describe the components of ASA and BSA in section 5.3.3.

### 5.3.1 Key Players in MA Service Architecture for SIP IP Telephony

Adapted from the Carleton Architecture, four actors play important roles in MA based SIP service architecture. They are First Level Service Creator, Second Level Service Provider, Service Management Unit (SMU), and End User.

1. *First Level Service Creator* is a company that creates and advertises services for second level service providers. It consists of SCC, SCR, and LUS. It brings opportunities for second level service providers to compete in the service market.
2. *Second Level Service Provider* is an enterprise that creates, advertises, and provides value-added services directly to users. It consists of ESC, SIR, and Enterprise LUS.
3. *Service Management Unit (SMU)* is responsible for managing service subscription and creating mobile agents for end users. SMU can discover LUS, and help end users to subscribe and update their desired services. After end users subscribe to the designed services, SMU will create mobile agents that contain subscribed services for them.
4. *End user* is the person who actually uses the services. It may or may not be the same as the service subscriber. For example, a company may subscribe to VPN service, but the employees in that company will serve as end users. On the other hand, an individual can serve as both service subscriber and end user when he subscribes to AIN or BIN services.

The Carleton H.323 Service Architecture has already given a detailed description of the first two actors and their service creation and subscription scenarios [CARL99]. In our Conceptual MA Architecture for SIP, these two actors act in the exactly same way.

SMU is responsible for discovering LUS and the subscribed service components in SIR, and the procedure of SMU service subscription is also described in detail in the Carleton Architecture. To simplify our MA architecture and scenarios for SIP IP telephony, we assume that SMU has already discovered all service components in a desired SIR, and it can fetch those components directly from this SIR to create mobile service agents. SMU is an agent provider, which provides and updates mobile service agents according to end user's subscription. In our architecture, there is only one SMU that provides mobile service agents to end users in different domains all over the network. We provide a mobile agent platform, SMU agency, to this agent provider.

The AIN and BIN services are individual services, so the person who subscribes to the service is the end user who uses it. We provide end user agency to every terminal in the network, so that end user can subscribe to services from any terminal, and he or she can also utilize those services from any terminal.

An end user can migrate to and register at different domains over the Internet. We assume that there is only one proxy server in every domain, and we name the agency connected to the proxy server as service agency.

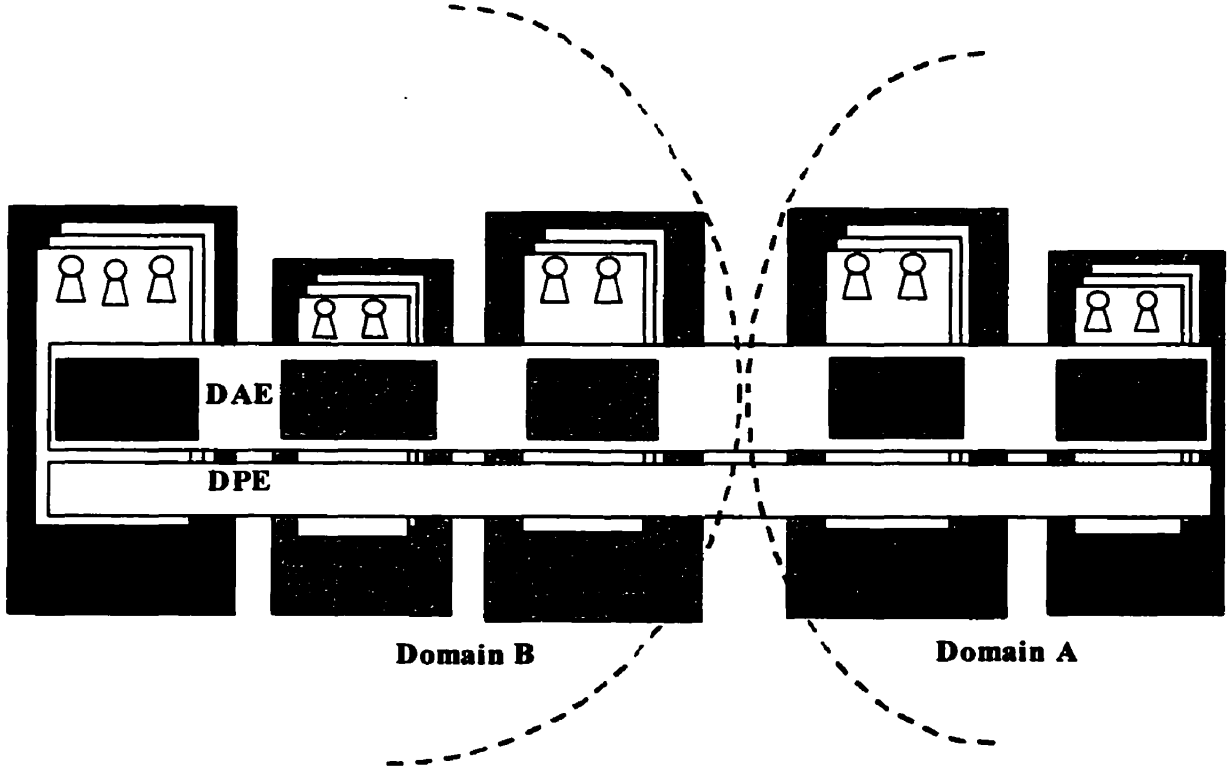



Figure 16: MA Service Architecture for SIP IP Telephony

### 5.3.2 Motivations of Adapting USA and CA to ASA and BSA

In the Carleton architecture, USA contains proxy objects for both originating and terminating services. There is no implementation code in USA, so USA has to create and activate CA, which contains service implementation logic and provides value added services directly to end users.

In our MA based architecture for SIP IP telephony, we will implement service logic and data internally in the mobile service agent, so USA can provide value added services directly to end users, and CA is not needed.

This approach has the advantage of immediate and on demand service provision. In the Carleton architecture, for terminating services, USA moves to the gatekeeper and creates CA by retrieving service logic and data from SIR after incoming call signalling arrives. Because it takes time for USA migration and CA creation, this approach is not very good for immediate and on demand service provision. On the contrary, in our MA service architecture, service logic and data is contained internally in the mobile service agent, and the mobile service agent is already at the proxy server agency before the incoming call arrives. Therefore, our architecture has a great advantage in immediate and on demand service provision.

In our MA based service architecture, instead of using one USA to provide all kinds of services, we use two kinds of mobile service agent, ASA and BSA, to provide originating services and terminating services. Because originating and terminating services should be provided at different places in the network, splitting the USA into two mobile service agents will facilitate the service mobility.

Normally, individual services are divided into originating services and terminating services. Originating services will be executed when a caller wants to initiate a call, and terminating services will be executed when an incoming call arrives. Ericsson CMS 8800 WIN *Subscriber Services,* based on different trigger points, are grouped into two types – AIN services stand for originating services, and BIN services stand for terminating services; accordingly, we provide AIN Service Agent (ASA) for originating services and BIN Service Agent (BSA) for terminating services.

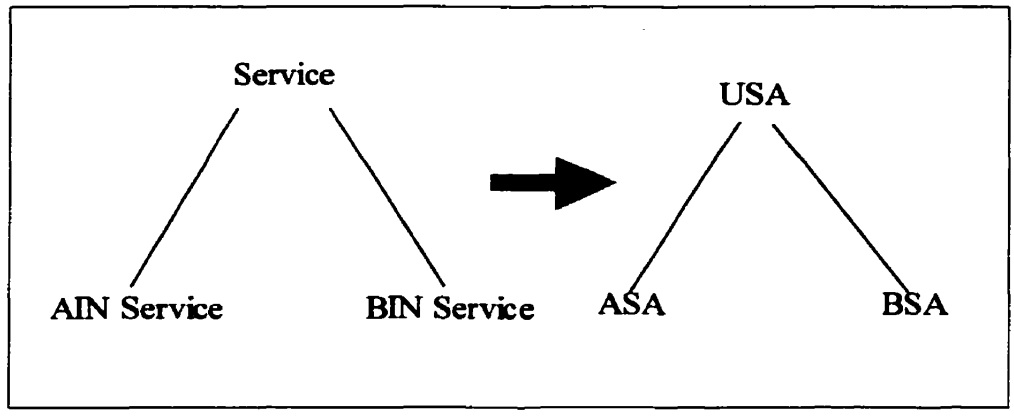



Figure 17: Splitting USA into ASA and BSA according to different kinds of services

In SIP based IP telephony, the ASA should enable individually subscribed and customised originating services to follow their associated users wherever they migrate. Realised as an agent, the ASA will be responsible for handling outgoing call services. First the ASA should follow the user to the domain he has moved to, then to the terminal he has moved to, so it can provide the user with subscribed AIN services whenever the user wants to initiate a call from that terminal.

The BSA, in contrast to the ASA, will handle incoming calls. Through the BSA, the user can define and customize a call handling policy. An initiated incoming call will first contact the BSA, which will decide what to do with this call. The BSA can follow the user to optimize the user's call handling. Instead of following the user to the end terminal, BSA will follow the user to the domain proxy server and wait for incoming calls in that proxy server. This makes incoming call handling more flexible and efficient than following the callee to terminals.

Therefore, the best place to provide originating services is the terminal that the user is currently at, and the best place to handle incoming call services is the domain proxy server that the user has migrated to. AIN and BIN services are provided at different places, so we should use two kinds of User Service Agent (USA) - ASA and BSA - to provide different kinds of services in different places.

### 5.3.3 Components of ASA and BSA

AIN includes Customer Control Access (CCA), Bulk Number Translation (BNT), Private Numbering Plan (PNP), Outgoing Call Allowance (OCA), and Outgoing Call Restriction (OCR). AIN provides a customized originating user profile to the user anywhere in the network. No matter which domain a user moves to and registers at, the AIN service profile should always be available for that user when he initiates a call.

BIN includes Selective Call Acceptance (SCA), Selective Call Rejection (SCR), Selective Call Forwarding (SCF). BIN is used for incoming call handling, so that the user has the ability to specify a call screening and handling policy. This means the user can specify what to do with incoming calls under the following well-defined conditions.

- User-dependent handling (screening): is a flexible communication screening function. A user can specify a list of people from whom he will be reachable, from whom the call is to be blocked, and from whom the call is to be forwarded.
- Time-dependent call handling: is a flexible call handling function. A user can specify a time-dependent condition of during what time the call should be forwarded to which specific phone number.

BIN is used for incoming call handling, so the BIN service profile should also be available at the domain the user moves to and registers at. Therefore, AIN and BIN service profile should follow the user to the domain he migrates to and registers at. Mobile Agent has a great advantage in service on demand, and can provide a good solution for AIN and BIN services in IP telephony environment.

Just as mentioned before, instead of providing one USA for all subscriber services, we provide two kinds of agents: one for originating services and the other for terminating services. If the subscribed services are AIN services, the service agent will contain a script for part or all of CCA, BNT, PNP, OCA and OCR service logic and data depending on the user's subscription. We call this service agent AIN Service Agent (ASA). If the subscribed services are BIN services, the service agent will contain a script for part or all

of SCA, SCR and SCF service logic and data depending on the user's subscription. We call this service agent BIN Service Agent (BSA).

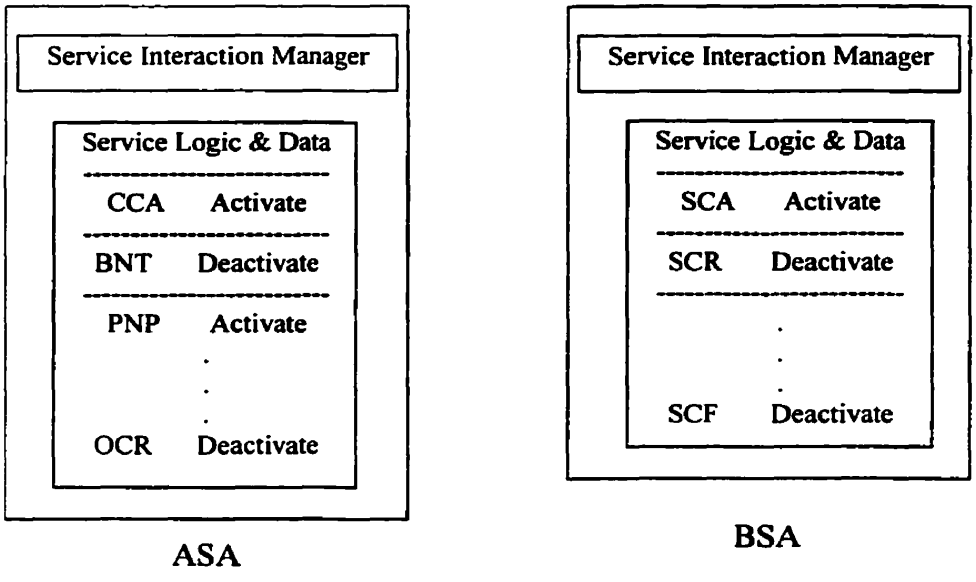



Figure 18: The Composition of ASA and BSA

An ASA consists of a Service Interaction Manager and Service Logic & Data. Service Interaction Manager is the main method that defines which AIN service class should be executed first, which AIN service class should be executed next, and which one should be executed last. Service Logic & Data contains many classes, and each class defines one AIN service (CCA, BNT, PNP, OCA, or OCR). The service class can be activated or deactivated depending on the end user's requirement. For example, if BNT class is deactivated in the ASA, then BNT service logic will not be executed when the end user initiates a call.

A BSA consists of a Service Interaction Manager and Service Logic & Data. Service Interaction Manager is the main method that defines which BIN service class should be executed first, which BIN service class should be executed next, and which one should be executed last. Service Logic & Data contains many classes, and each class defines one BIN service (SCA, SCR, or SCF). The service class can be activated or deactivated depending on the end user's requirement. For example, if SCA class is deactivated in the BSA, SCA service logic will not be executed when an incoming call arrives.

## 5.4 Service Mobility

The Internet is geographically divided into areas called domains. In SIP, each domain may have one or more proxy or redirect servers responsible for call handling. In our architecture, we assume that there is only one proxy server in each domain.

### 5.4.1 Mobility Management in SIP

In SIP IP telephony, when a user enters a new domain, he registers himself with that domain proxy server from a local terminal [ATS98]. If the domain he enters is not his home domain, except for local domain proxy server, he also registers himself remotely with his home domain proxy server [ATS98].

In IP telephony, an end user can migrate to and login at different terminals all over the Internet. A user may register with the home domain proxy server from different terminals in his home domain, and he may migrate to another domain and register both with his home domain proxy server and the other domain proxy server. All moving users are assumed to have a permanent "home proxy" that never changes. They also have a permanent home address that can be used to determine their home locations. The proxy or redirect server is responsible for forwarding or redirecting incoming calls of a user's home address to wherever the user may migrate in the network.

Service mobility means that the same kind of services should be provided to different domains or even different networks. Although SIP can provide value-added services at end systems in different domains, it is not flexible and efficient to support service mobility. For example, SIP user agent can provide originating services at end systems, but every end system in the network should support service logic for all subscribers because end users may migrate and login at different end systems in different domains from time to time. This requires intelligent end systems to have very large memory to store all user profiles, which is not quite feasible.

### 5.4.2 Service Mobility in MA based Architecture for SIP IP Telephony

Service mobility is an important issue in IP telephony environment. An Internet user may migrate within a domain or between domains, and uniform services should be provided to that user no matter where he migrates. Therefore, there are two kinds of service mobilities: service mobility within domain, and service mobility between domains.

If a user migrates within the same domain, BSA should stay at the local domain proxy server, but ASA should follow the user and migrate to the terminal that the user is currently at. When a user moves to a new terminal within the same domain, first he sends a REGISTER request to the local domain proxy server, indicating his new address in the Contact header. The local domain proxy server checks this user's register file in the database, and appends a new record of the user's current address at the end of the file. Then, the domain proxy server finds the old address of the user in the register file, which the ASA is currently at, and sends the user's new address to the ASA. Finally, the ASA moves to the end system the user is currently at according to the new address it received.

If a user migrates to another domain, in order to provide service on demand, both ASA and BSA should follow the user and migrate to the domain the user is currently at. This migration procedure can be realized by copying the service agent from the user's home domain, and sending the new-copied agent to the current domain. But normally, the service agent will follow the user from the previous domain to the current domain after the home proxy server gets the registration information from the user in the current domain. When a user moves to a terminal at a new domain, first he sends a REGISTER request to the home domain proxy server, indicating his new domain address in Contact header. He also sends a REGISTER request to the local domain proxy server, indicating his new terminal address in Contact header. The home domain proxy server finds the user's register file in its database, and appends a new record at the end of the file to indicate the user's current domain address. Then, the home domain proxy server finds the user's old address in the register file, indicating the ASA or BSA's current location, and sends the user's new domain address to the ASA or BSA. Finally, the ASA or BSA moves to the new domain proxy server the user is currently at according to the new address it received.

After migration to the new domain proxy server, BSA will stay at this domain proxy server, and ASA will move to the terminal that the user is currently at accordingly.

## 5.5 Evaluation

Internet telephony should enable personal mobility and service mobility. A service user can access the IP telephony services from different locations, even from different networks using the same identification. This user can also be reached in different locations or different networks by this personal identification.

The advent of IP telephony will multiply the number of terminals an end user may employ, and will therefore make the issue of personal mobility more accurate than ever. VAS in IP telephony should be provided ubiquitously in the home or roaming domains of the customer. Mobile agent platform can provide better support of personal and service mobility than SIP by providing intelligence on demand to the domain proxy server or user agent the user is currently at.

As mentioned before, SIP is immature and still has some shortcomings for IP telephony services. For example, as a client server protocol, SIP depends on network availability, produces network traffic, and is difficult to support service on demand when the end user migrates to another domain. It is not feasible for SIP user agents to use intelligent end systems that have enough memory to store all service logic for all users. In order to reduce network traffic and support service mobility, the service intelligence related to the calling party or the called party should be brought flexibly as close as possible to those parties, thereby minimizing the use of signaling network resources.

The mobile agent technology is suitable for the improvement of distributed object systems. Due to the benefits of MA technology, such as dynamic, on-demand provision and distribution of services, a reduction in network traffic and independence regarding server failures, various problems and inefficiencies of SIP client/server architectures can be handled.

Our MA based architecture enables on demand provision of customized supplementary services by dynamic constructing and updating user service agent using service components from SIR. Our MA based architecture also supports service mobility,

so the corresponding intelligence for service control can be placed dynamically at the most appropriate locations. Mobile service agents can be dynamically deployed to proxy servers and end-user systems where the services are needed.

The provision of services can be performed very flexibly by sending the service agent dynamically and on demand to the domain that the user has moved to. By performing local interactions at that domain server or user end system instead of remote procedure calls, MA can enable network traffic to be reduced. By moving the required logic and data to end systems, a reduction in dependence regarding network availability (robustness) is also achieved.

# Chapter 6
# Scenarios

We proposed the Conceptual MA Architecture for SIP IP telephony in the previous chapter, and now we describe service provision scenarios to demonstrate how this architecture works. Section 6.1 describes subscription phrase scenarios, including scenarios for service agent creation and service agent update. Section 6.2 illustrates service mobility scenarios when user registers within domain or between domains. Finally, section 6.3 describes execution phrase scenarios and concludes this chapter.

## 6.1 Subscription Phrase Scenarios

The subscription phrase scenarios are similar for all subscriber services, no matter what services are triggered at originating end – AIN services, or terminating end – BIN services.

SMU is an agent provider responsible for agent creation and update according to user's subscription, so SMU has the responsibility of discovering LUS and the subscribed service components in SIRs. To simplify our scenarios, we skip the service discovery part scenarios, which has described in Carleton architecture [CARL99], and assume that the SMU has already discovered all service components in a desired SIR, so it can fetch those components directly from this SIR to compose mobile service agents.

The subscription phrase scenarios are slightly different for service agent creation and service agent update, so we describe these two scenarios separately in section 6.1.1 and section 6.1.2.

### 6.1.1 Service Agent Creation

SMU is an agent provider, so it has a register file to record all the agents it created before. SMU can be accessed through WWW. In the SMU web site, the names of two agents, ASA and BSA, are posted for subscription. Subscriber can click on either of them, then services available for ASA or BSA will appear for subscription. If you click on

ASA, then originating services like CCA, BNT, PNP, OCA and OCR will be available for subscription. If you click on BSA, then terminating services like SCA, SCR and SCF will be available for subscription. You can subscribe to one or more services for each mobile service agent by selecting that service name and confirming it.

1. The end user accesses SMU agency via the WWW, and clicks on an agent name: ASA or BSA. The SMU agency checks its register file and finds that there is no service agent created for that user. The user subscribes to one or more services for that new agent by clicking on the service names under it.
2. According to the end user's subscription, SMU send a service subscription request to SIR, listing the service components required for constructing the new service agent.
3. SIR checks the service request it received, and sends the required service components to SMU according to the subscription list.
4. SMU creates and constructs the new mobile service agent by assembling service components received from SIR. The services in the mobile service agent are deactivated. Then SMU sends the new mobile service agent to the end user agency at a terminal in the subscriber's home domain.
   a) If the subscribed services are AIN services for outgoing calls, ASA is sent to the end user agency.
   b) If the subscribed services are BIN services for incoming calls, BSA is sent to the end user agency.
5. The end user configures and customizes the service agent with individual service data. For AIN services, the end user activates and customizes part or all of the CCA, BNT, PNP, OCA and OCR script in ASA. For BIN services, the end user activates and customizes part or all of the SCA, SCR and SCF script in BSA.
6. Before the agent executes its designed task, it automatically migrates back to the SMU agency in order to allow security checks, e.g. the determination of service activation and code modifications. Therefore, ASA or BSA will migrate back to the SMU agency for security checks.
7. If the security checks have been successful, the mobile agent will move back to the end user's home domain, and ASA or BSA will move to the service agency co-located with the proxy server in the subscriber's home domain.

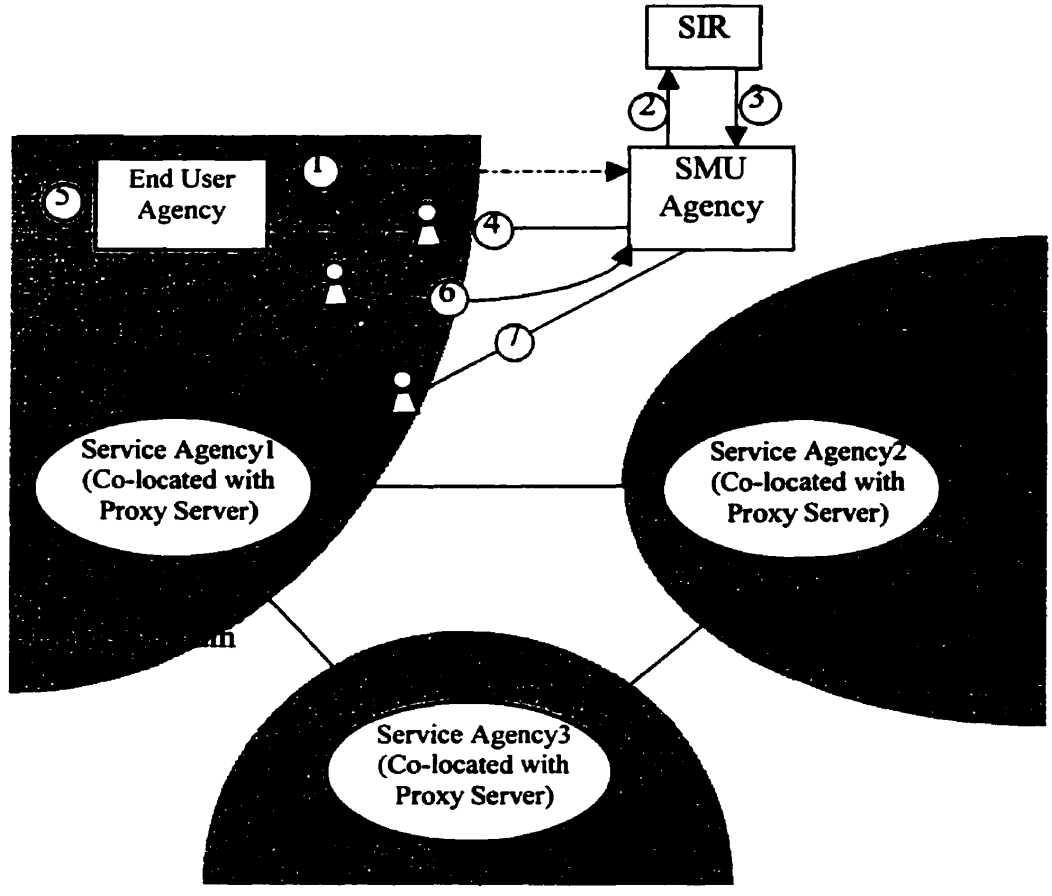



Figure 19: MA Based Subscription Phase Scenarios for Service Agent Creation

The upper figure illustrates the subscription phase scenarios when a user subscribes to AIN or BIN services for the first time, and describes how mobile service agents are created. After the agent creation, an end user may find that he wants to subscribe to more AIN or BIN services, and add them to the ASA or BSA that he has already subscribed to. An end user may also want to delete some services that he no longer uses from the mobile service agent, or to deactivate some services temporarily. We will describe service agent update scenarios in next section.

### 6.1.2 Service Agent Update

We describe the service agent update scenarios as follows:

1. The end user accesses the SMU agency via the WWW, and wants to subscribe to some more services or to delete some existing services. The user selects the agent he wants to subscribe to by clicking on the agent name: ASA or BSA, then selects one or

more services under that agent name. The SMU checks its register file and finds that an ASA or BSA already exists for that user.

2. The SMU sends a request to the service agency where the mobile service agent is currently located, and tells the service agency to send the required service agent back.
3. Receiving the request from SMU, the mobile service agent will migrate back to SMU for update.
4. SMU checks the existing service profile of the returned mobile service agent, and compares it with the new subscription. If the new subscription requires some new services that are not included in the old service agent, SMU will send a service subscription request to SIR, listing the services components that are not available in the old service agent for service update.
5. SIR checks the service request it receives, and sends the required service components to SMU according to the subscription list.
6. SMU updates the old service agent with a new one by deleting services no longer subscribed to from the old service agent, and then adding new subscribed service components to it. The new services in the mobile service agent are deactivated now. Then SMU sends the new mobile service agent to the end user agency at a terminal in the subscriber's home domain.
7. The end user configures and customizes the service agent with individual service data. He can activate or deactivate services in this service agent, and he can also provide personal data to this service agent.
8. Before the agent executes its designed task, it automatically migrates back to the SMU agency in order to allow security checks, e.g. the determination of service activation and code modifications. Therefore, ASA or BSA will migrate back to SMU agency for security checks.
9. If the security checks have been successful, the mobile agent will move back to the end user's home domain, so ASA or BSA will move to the service agency co-located with the proxy server in the subscriber's home domain.

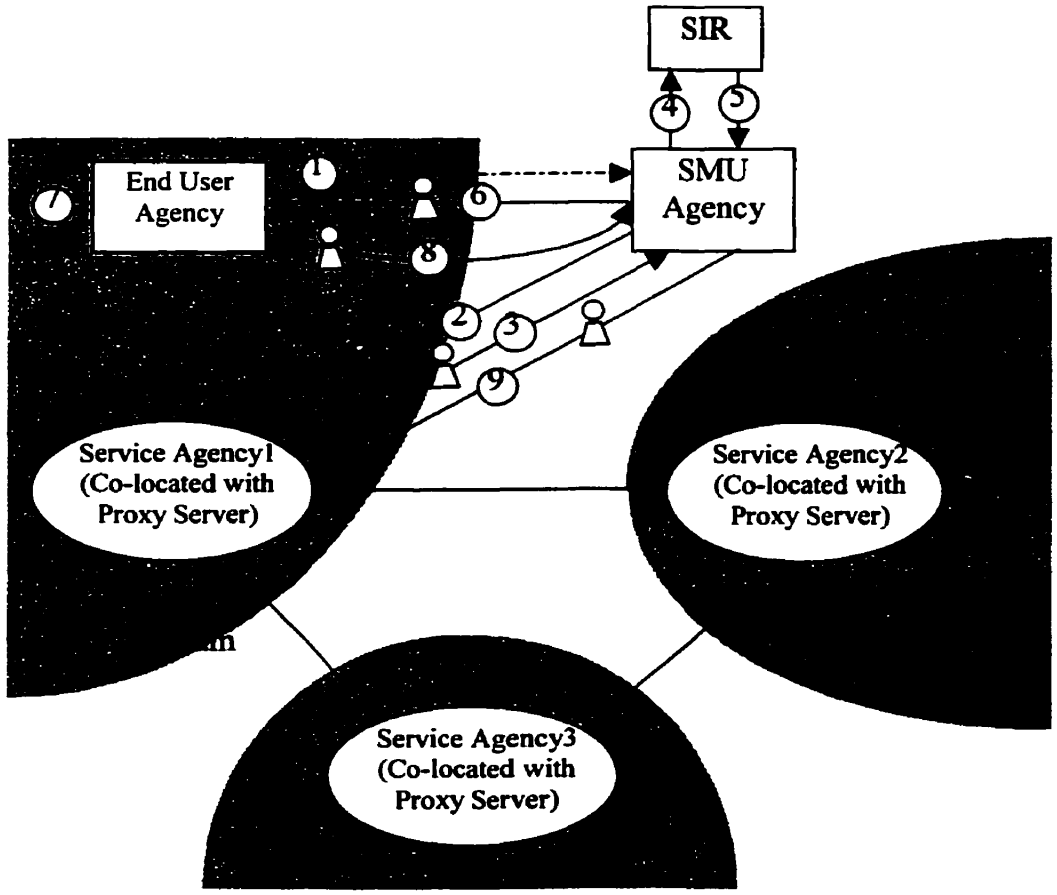



Figure 20: MA Based Subscription Phase Scenarios for Service Agent Update

The subscription phase scenario for service agent update is similar to the subscription phase scenario for service agent creation. Instead of using components completely from SIR, the new service agent is created by adding new components from SIR to the old service agent or by deleting unused components from it. After the mobile service agent is created, the end user can customize this service agent by activating some services for this agent, deactivating some services for this agent, or providing personal data for some services.

### 6.2 Service Mobility Scenarios

As mentioned earlier in this chapter, mobile agent technology provides a unified framework for the implementation of user profiles and service logic through mobile agents following the users to different domains in the network. There are two kinds of service mobility issues: mobility within a domain and mobility between domains. We will

describe these two kinds of service mobility scenarios separately in section 6.2.1 and section 6.2.2.

### 6.2.1 Service Mobility within Domain

The service mobility scenario for a user migrating within a domain is much simpler than the scenario for user migrating between domains. When an end user migrates within his home domain, AIN services will follow the user to the terminal that he is migrating to, but BIN services will stay at the home domain proxy server instead of moving.

#### 6.2.1.1 AIN Service Mobility within Domain

AIN services should allow the support of service mobility, so the moving user will be provided with his originating services at different terminals in the network. To permit service mobility and to allow the user to take his subscribed and customized AIN services with him while moving, ASA should migrate to the end user agency where the call will be initiated, and wait there for execution until the user want to initiate a call.

When a user first logs in at a terminal in his home domain, the ASA will migrate from the home domain proxy server to that terminal after the user registers from that terminal.

1. A user hwang@mcgill.ca registers for the first time from a terminal cs.mcgill.ca in the home domain to the domain Proxy Server mcgill.ca. In the SIP REGISTER request, To and From headers are hwang@mcgill.ca, Request_URI is sip:mcgill.ca, and Contact header is hwang@cs.mcgill.ca. The Proxy Server at mcgill.ca will create a registration file for hwang, and put the registration information including the current contact address at the end of the file.
2. The Proxy Server gets the current registration information and contact address from hwang's registration file, and gives it to the ASA in Service Agency 1.
3. The ASA moves from Service Agency 1 to the terminal that the user is currently at (cs.mcgill.ca) according to the current registration address.

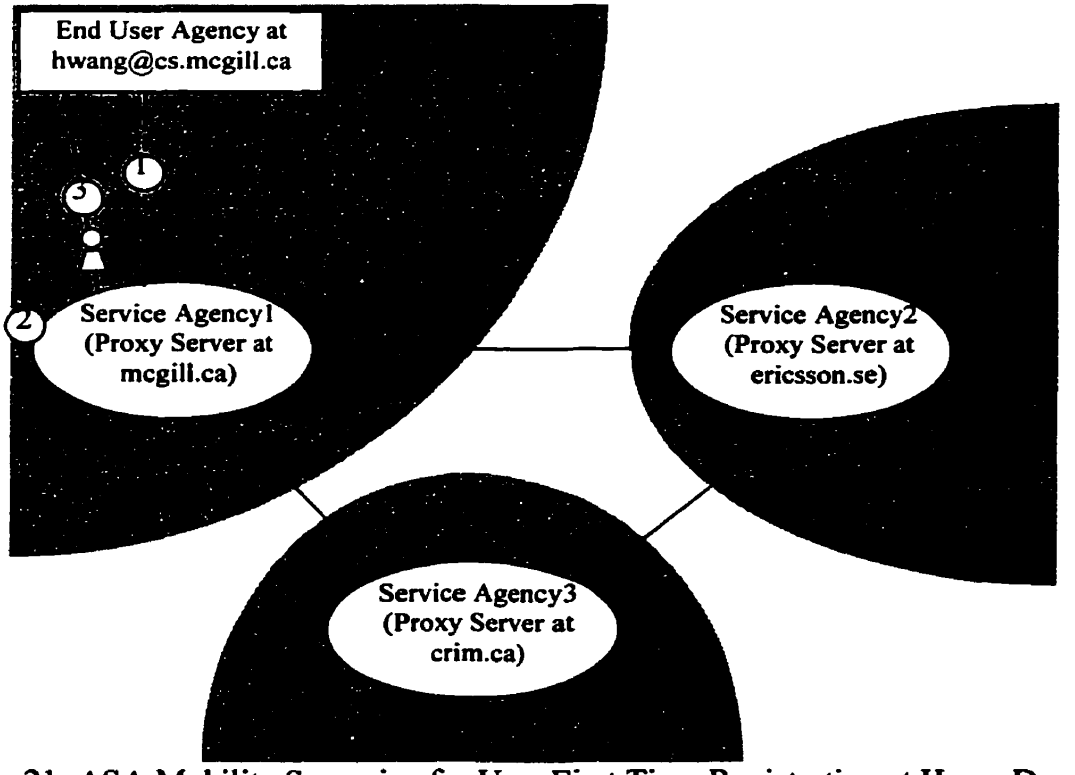



Figure 21: ASA Mobility Scenarios for User First Time Registration at Home Domain

The AIN service mobility scenario is a little bit different when a user migrates from one terminal to another terminal in the local domain and registers from the new terminal.

1. A user hwang@mcgill.ca moves from terminal cs.mcgill.ca to terminal lisa.mcgill.ca in the same domain, and registers with the domain Proxy Server mcgill.ca from the new location. In the SIP REGISTER request, To and From headers are hwang@mcgill.ca, Request_URI is sip:mcgill.ca, and Contact header is hwang@lisa.mcgill.ca. The proxy server at mcgill.ca finds the registration file of hwang in its database, and puts the new registration information including the current contact address at the end of that file.
2. Through the registration file, the domain Proxy Server gets the hwang's old registration address at hwang@cs.mcgill.ca, so it knows the ASA's current location.
3. According to information from the domain Proxy Server, service agency 1 sends the current user address of hwang@lisa.mcgill.ca to the ASA agent at hwang@cs.mcgill.ca.

4. The ASA moves to the user agency at hwang@lisa.mcgill.ca from hwang@cs.mcgill.ca according to the current registration address, and then stays there waiting for execution.

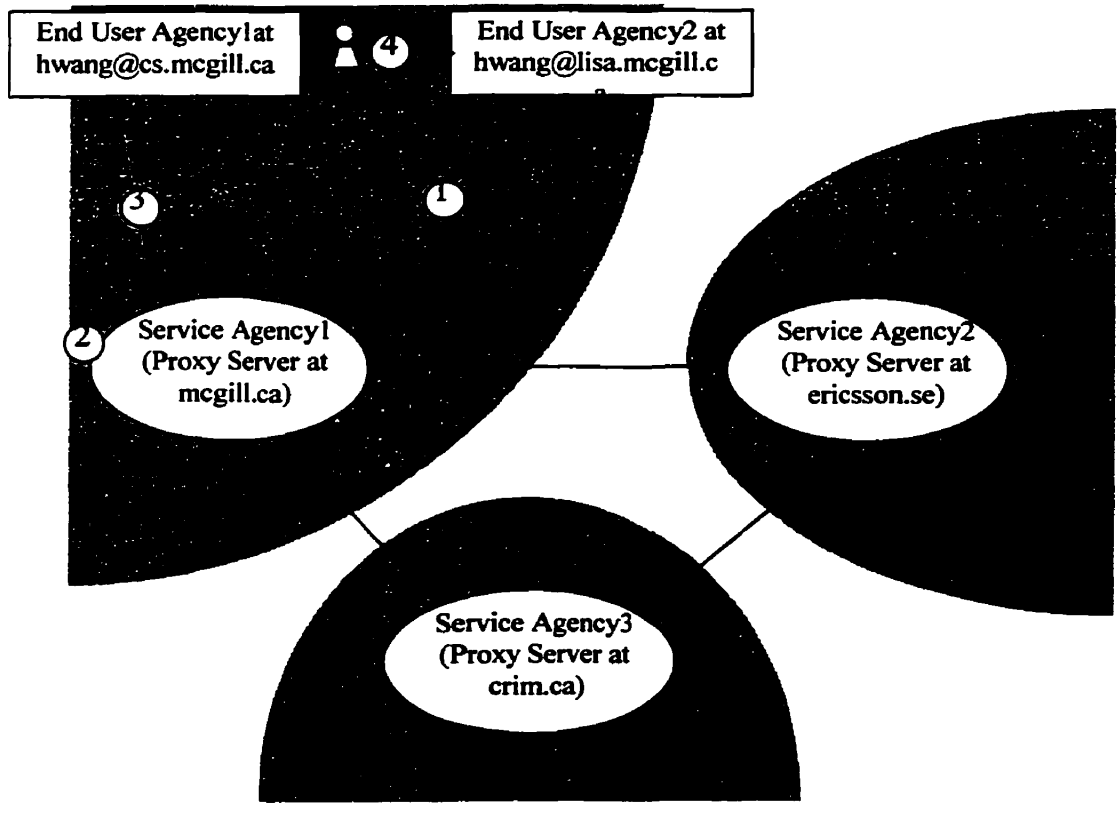



Figure 22: ASA Mobility Scenario for User Migration within Domain

### 6.2.1.2 BIN Service Mobility within Domain

BIN services should allow a user to customize his call control by configuring a set of control attributes. With the realization of the BIN services, the user can define various conditions to be applied on an incoming call - accept it, reject it, or forward it, so the user can get a powerful tool to define call-handling policies. The user can define various conditions to be applied on an incoming call, thus allowing time-dependent, service dependent and user-dependent call handling to be initiated.

To realize BIN services, BSA should follow the user and migrate to the proxy server agency at the destination domain. On the other hand, if the user just moves from one terminal to another terminal within the same domain, the BSA should stay at the domain proxy server agency, and wait for the incoming calls. Therefore, the BIN service mobility scenario for a user moving from one terminal to another terminal within domain is very simple: the BSA just stays at the local domain proxy server and waits to be executed.

### 6.2.2 Service Mobility between Domains

Just as mentioned before, when an end user migrates between domains, the ASA and BSA should follow that user and migrate to the other domain near the originating or terminating end of the call.

#### 6.2.2.1 AIN Service Mobility between Domains

AIN services are originating end services. The ASA will be triggered whenever the user wants to use a service to initiate a call. Therefore, ASA should first follow the user to the domain that he moves to, then to the terminal he log in at.

Service mobility scenario of ASA migration between domains is described as follows:

1. A user hwang@mcgill.ca moves from terminal cs.mcgill.ca in domain mcgill.ca to terminal lmc.ericsson.se in domain ericsson.se, and registers with the home domain Proxy Server mcgill.ca from the new location. In the SIP REGISTER request, To and From headers are hwang@mcgill.ca, Request_URI is sip:mcgill.ca, and Contact header is lmcadwa@lmc.ericsson.se. The home domain proxy server (sip:mcgill.ca) finds the registration file of hwang in its database, and puts the new registration information including the current contact address of hwang (lmcadwa@lmc.ericsson.se) at the end of that file.
2. The user hwang registers with the foreign domain Proxy Server ericsson.se from the new location he has moved to. In the SIP REGISTER request, To and From headers are hwang@mcgill.ca, Request_URI is sip:ericsson.se, and Contact header is lmcadwa@lmc.ericsson.se. The foreign proxy server (sip:ericsson.se) creates a new registration file, or finds the registration file of hwang in its database if hwang had

visited before, and puts the new registration information including the current contact address of hwang (lmcadwa@lmc.ericsson.se) at the end of that file.

3. Through the registration file, the home domain Proxy Server (sip:mcgill.ca) gets hwang's old registration address at hwang@cs.mcgill.ca, so it knows the location of ASA.
4. The service agency at domain 1 gets the current user address of lmcadwa@lmc.ericsson.se from registration file and sends it to the ASA agent at hwang@cs.mcgill.ca.
5. The ASA moves to user agency 2 at lmcadwa@lmc.ericsson.se from hwang@cs.mcgill.ca according to the current registration address, then stays there waiting for execution.

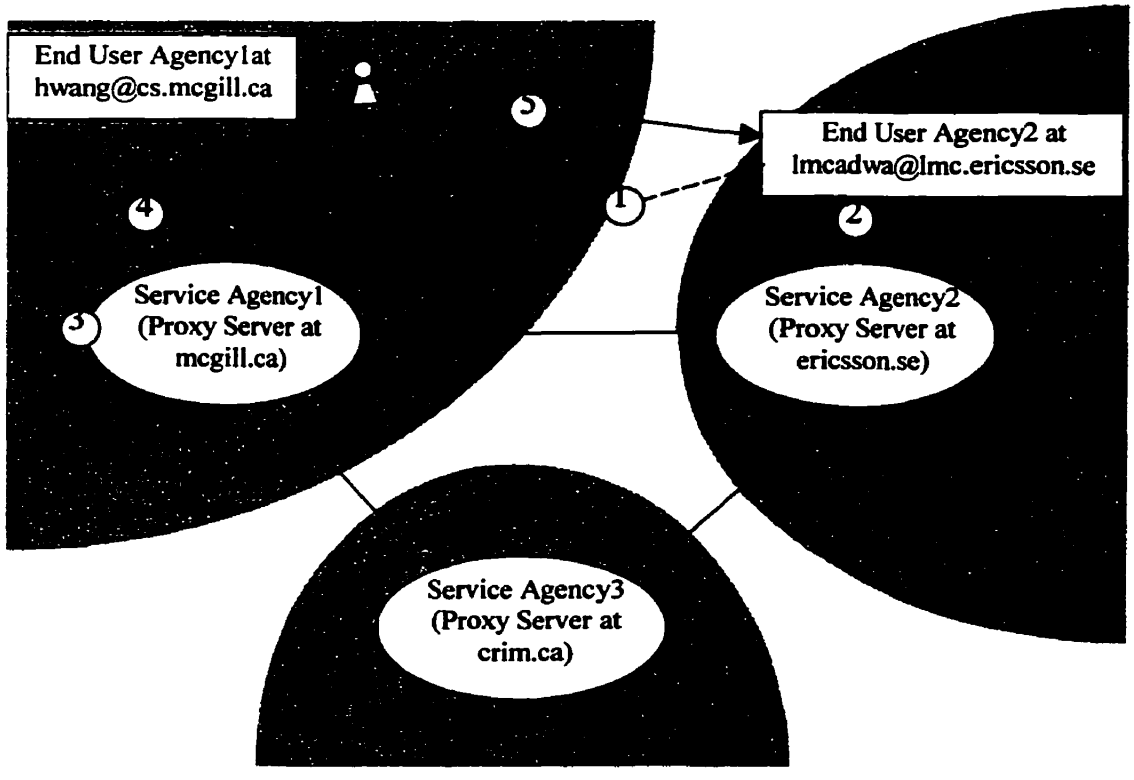



Figure 23: AIN Service Mobility Scenarios for User Registration between Domains

#### 6.2.2.2 BIN Service Mobility between Domains

BIN services should allow a user to customize his/her call handling by configuring a set of control attributes. With the realization of BIN services, the end user will get a powerful tool to define call-handling policies. The user can define various conditions to be applied on an incoming call, thus allowing time-dependent, service dependent and user-dependent call handling to be initiated. Like ASA, BSA will follow the end user to migrate to the server agency in the new destination domain. Then BSA will stay at that service agency instead of moving to an end user agency.

The following is the service mobility scenarios for BIN services: BSA follows the end user, migrates to a foreign domain agency, and stays there.

0. A user hwang@mcgill.ca moves from terminal cs.mcgill.ca in domain mcgill.ca to terminal lmc.ericsson.se in domain ericsson.se.
1. The user hwang registers with the home domain Proxy Server mcgill.ca from the new location. In the SIP REGISTER request, To and From headers are hwang@mcgill.ca, Request_URI is sip:mcgill.ca, and Contact header is lmcadwa@lmc.ericsson.se. The home proxy server (sip:mcgill.ca) finds the registration file of hwang in its database, and put the new registration information including the current contact address of hwang (lmcadwa@lmc.ericsson.se) at the end of that file.
2. The user hwang registers with the foreign domain Proxy Server ericsson.se from the new location he has moved to. In the SIP REGISTER request, To and From headers are hwang@mcgill.ca, Request_URI is sip:ericsson.se, and Contact header is lmcadwa@lmc.ericsson.se. The foreign proxy server (sip:ericsson.se) creates a new registration file for hwang, or finds the registration file of hwang in its database if hwang had visited before, and puts the new registration information including the current contact address of hwang (lmcadwa@lmc.ericsson.se) at the end of that file.
3. Through the registration file, the home domain Proxy Server (sip:mcgill.ca) gets hwang's old registration address of hwang@cs.mcgill.ca, so it knows that the BSA is in service agency 1 of the home domain (sip:mcgill.ca). Then the home domain proxy server sends the current user address to the service agency.

4. The service agency at home domain (sip:mcgill.ca) sends the BSA to the service agency at the foreign domain (sip:ericsson.se) according to the current registration address (lmcadwa@lmc.ericsson.se).

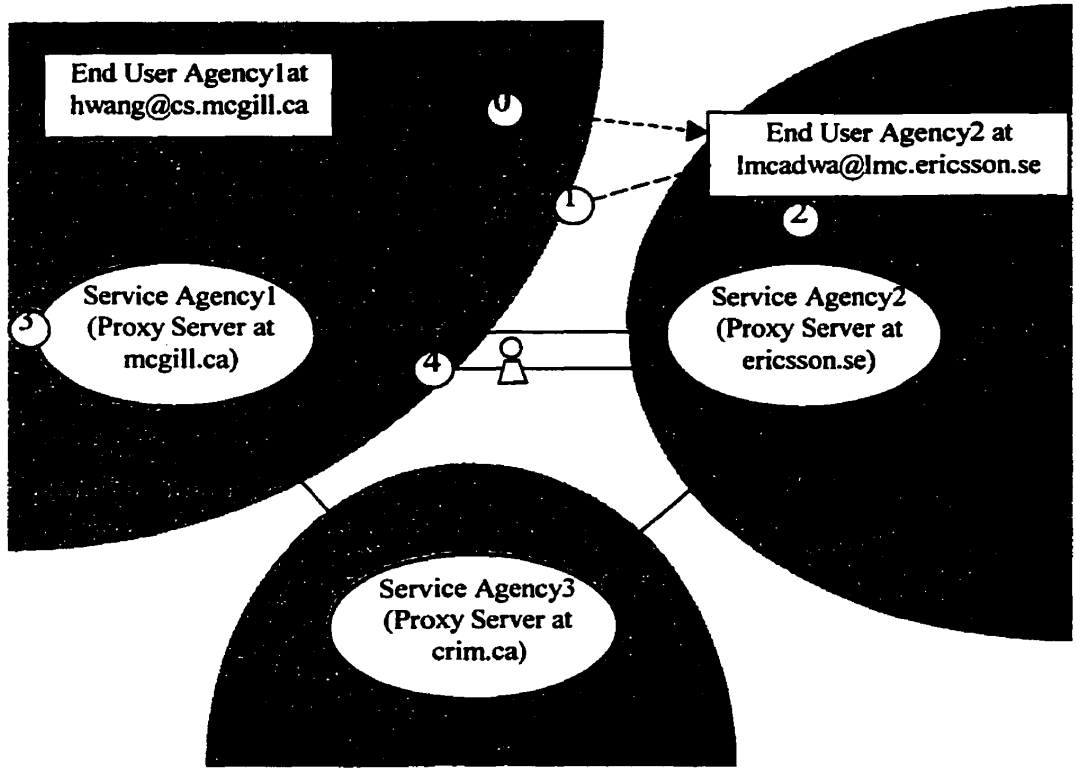



Figure 24: BIN Service Mobility Scenarios for User Registration between Domains

### 6.3 Execution Phase Scenarios

There are two kinds of execution phase scenarios. One is using MA to initiate and handle a call independently. The other is using SIP requests and responses together with MA to initiate and handle a call. We first describe the scenarios of MA handling a call independently.

Just as we mentioned before, the BSA is responsible for handling incoming calls, and the ASA is responsible for initiating outgoing calls. They are connected directly by using their own protocols (e.g. CORBA communication mechanisms).

The ASA that holds the customized originating end service information will be triggered whenever the user wants to use an AIN service (exp. Outgoing Call Allowance)

to make a call. The called party BSA receives the call request from the calling party ASA and allows the called party user to restrict (accept, forward and reject) such a request. After the incoming call signalling contacts the BSA, a user-customized policy in BSA will be evaluated to decide how the call is to be handled. This kind of direct interaction between ASA and BSA will be enabled using the DPE, which connects the agencies.

The previous scenario enables the mobile service agents to communicate through DPE. In this scenario, excepting for AIN services, ASA also handles the basic originating call processing. Excepting for BIN services, BSA also handles the basic terminating call processing. Therefore, it requires every user to have an ASA and a BSA in his current domain, even if he doesn't subscribe any AIN or BIN services. Creating an ASA and a BSA for every user, and letting them migrate with that user even if he doesn't subscribe to any service, will increase network traffic and consume network resources unnecessarily.

The following execution phrase scenario uses ASA and BSA together with SIP messages to handle calls and provide AIN and BIN services.

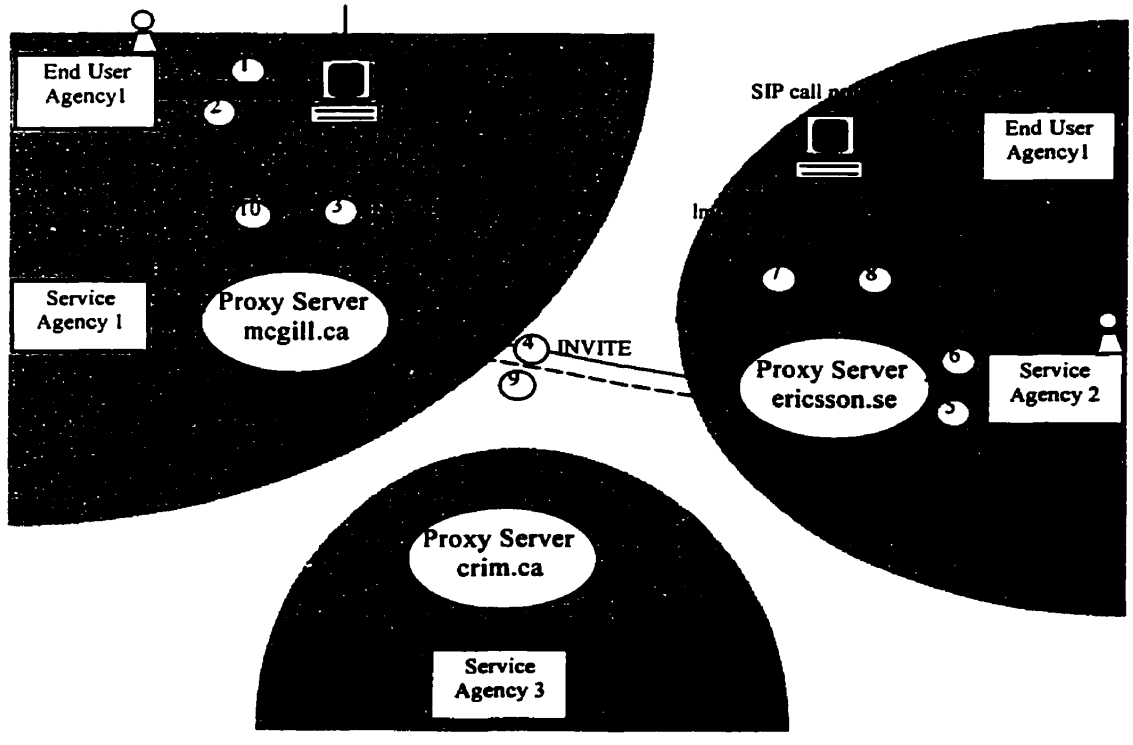



Figure 25: MA Based Service Execution Scenarios for SIP IP Telephony

Before an end user bob@mcgill.ca wants to initiate a call to hwang@mcgill.ca from terminal cs.mcgill.ca, a SIP user agent client at cs.mcgill.ca has been initiated.

1. Before initiating a call to hwang@mcgill.ca using INVITE request, the SIP call processing logic at cs.mcgill.ca will meet a detection point, which blocks the call processing and sends a request to end user agency1 to see if there is any subscribed AIN service for bob@mcgill.ca.
2. After the end user agency1 is contacted, the agency1 will check its register to see if there exists an ASA for bob. If the ASA exists, the user agency1 will trigger the ASA to execute AIN service logic in the end user agency. After executing the service logic, the result will be returned to the SIP user agent. If the ASA does not exist, a result indicating no AIN service subscription for bob will be returned.
3. Triggered by the returned value, the SIP call processing logic at cs.mcgill.ca will continue to execute. The call will continue to proceed or terminate according to the returned value. If the call should continue to proceed, an INVITE request will be sent from the SIP user agent client at cs.mcgill.ca to the home domain proxy server at mcgill.ca.
4. The SIP call processing logic at home domain proxy server receives the INVITE request, then it blocks itself and sends a request to service agency1, which checks if there exists a BSA for hwang@mcgill.ca. There is no BSA in service agency1, so an empty result will return. The call processing logic at mcgill.ca continues its execution. It checks its registration database and finds that the current contact address of hwang@mcgill.ca is lmcadwa@lmc.ericsson.se. It sends a INVITE request to ericsson.se proxy server with Request_URI field of lmcadwa@lmc.ericsson.se
5. After the proxy server at ericsson.se receives the INVITE request, the call processing logic of this proxy server will be responsible for handling this incoming call. During its execution, the call processing logic will meet some detection points, which block and send a request to the called party BSA in service agency2.
6. After the service agency2 is contacted, it will check its agency register to see if there exists a BSA for hwang@mcgill.ca. There does exist a BSA for hwang@mcgill.ca in

this agency, so service agency2 will trigger that BSA to execute BIN service scripts. After executing the BIN screening service logic: SCA, SCR, and/or SCF, a result will be returned to the ericsson.se proxy server.

7. Triggered by the returned value, the SIP call processing logic at ericsson will unblock and continue to execute. The call will continue to proceed or terminate according to the returned value. If the call should continue, an INVITE request will be sent from ericsson.se SIP proxy server to the SIP user agent server at lmc.ericsson.se.
8. The called party lmc.ericsson.se receives the INVITE request and decides to accept it. The user agent server at lmc.ericsson.se sends a 200 OK response to proxy server at ericsson.se.
9. The proxy server at ericsson.se receives the 200 OK response, and forwards the response to mcgill.ca proxy server.
10. The proxy server at mcgill.ca receives the 200 OK response, and forwards the response to cs.mcgill.ca user agent client.

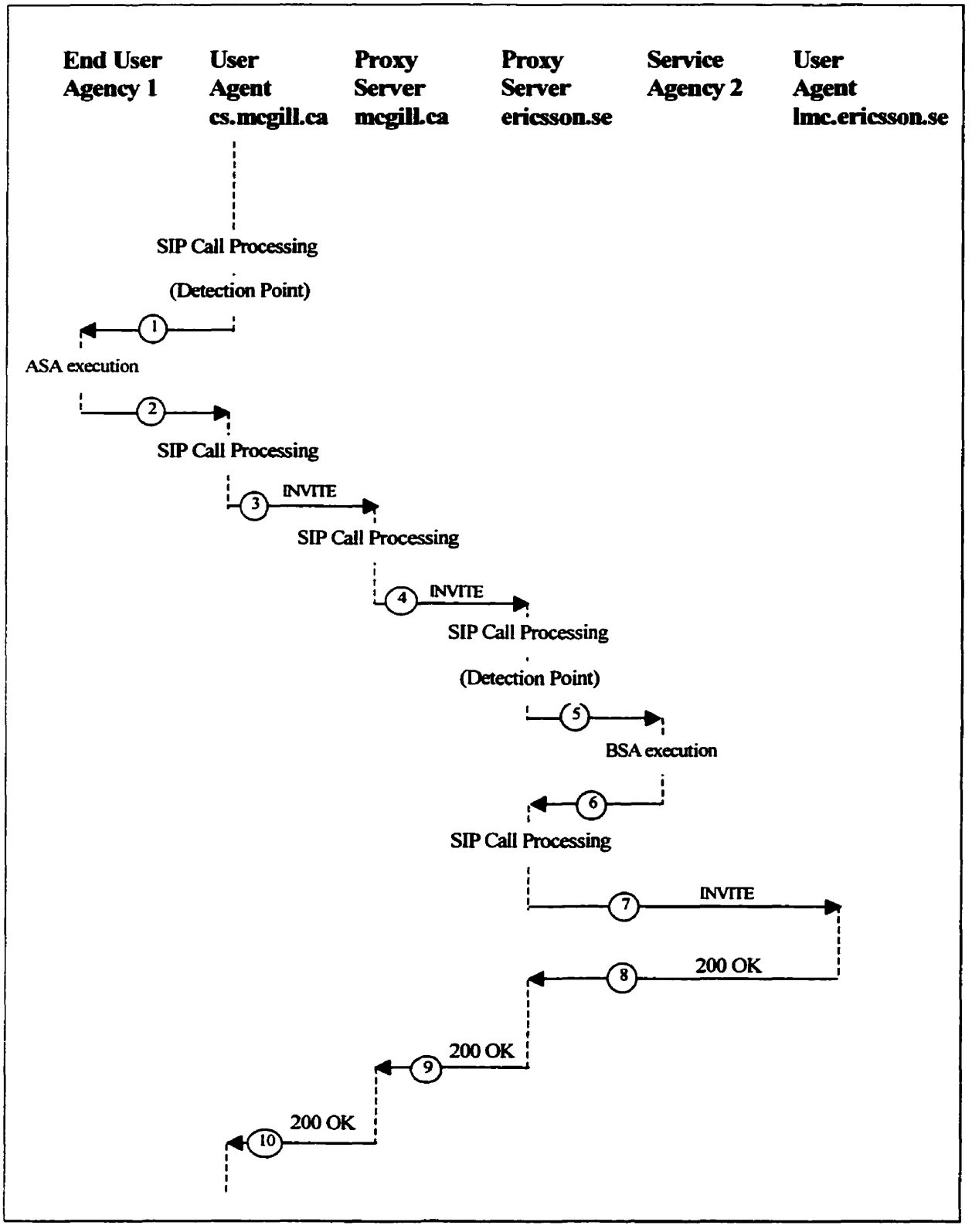



Figure 26: MA and SIP Based Service Execution Scenarios

Finally, the caller bob@cs.mcgill.ca receives the 200 OK response, and knows that the user hwang@mcgill.ca can be contacted at lmcadwa@lmc.ericsson.se. So it sends an ACK to the called party, and the session will be established. After the session initiation, voice and data can be sent directly from the caller to the callee.

# Chapter 7
# Implementation Architecture using Grasshopper Platform

After providing the conceptual MA architecture and scenarios for SIP IP telephony, we propose our implementation architecture using Grasshopper platform as follows. Section 7.1 introduces the grasshopper platform and the programming environment. Section 7.2 illustrates the mechanisms of using Grasshopper platform to realize SMU functionality. Finally, section 7.3 gives an example of how to use this architecture to realize subscription phase scenario and concludes this chapter.

## 7.1 Grasshopper Platform and Programming Environment

Grasshopper is a Distributed Agent Environment (DAE) provided by IKV++. Under the DAE, a distributed processing environment (DPE) will span all potential servers and end systems. All servers and end systems involved will provide corresponding agencies, which can be divided into places and enable the migration of mobile agents. These agents contain intelligence and can provide value-added services autonomously.

Besides Agent, Place and Agency, Region Registry is also an important software component in Grasshopper platform. A region is a logical entity, which groups a set of agencies that belong to the same authority. Associated with each region is one region registry that registers and localizes DAE components.

The functionality of Grasshopper is provided on one hand by the platform itself, i.e. by *core agencies* and *region registries*, and on the other hand by *agents* that are running within the agencies, in this way enhancing the platform with new capabilities [GBC99]. The following possibilities regarding the access of Grasshopper functionality must be distinguished:

- Agents can access the functionality of the local agency, i.e. the agency in which they are currently running, by invoking the methods of their super classes *Service*, *StationaryAgent*, and *MobileAgent*, respectively. These super classes are provided by

the platform in order to build the bridge between individual agents and agencies, and each agent has to be derived from one of the classes StationaryAgent or MobileAgent.

- Agents as well as other DAE or non-DAE components, such as user applications, are able to access the functionality of *remote agencies* and *region registries*. For this purpose, each agency and region registry offers an external interface AgentSystemP or RegionRegistrationP, which is a proxy object that can be accessed via the Grasshopper communication service. The class RegionRegistrationP enables an agent to access the region registry in order to retrieve information about registered components, i.e. agencies, places, services, and agents.
- Agencies and region registries may optionally be accessed by means of the MASIF compliant interfaces MAFAgentSystem and MAFFinder.

### 7.1.1 Agent Class Structure

In the context of Grasshopper, each agent is regarded as a service, i.e. as a software component that offers functionality to other entities within the DAE. Each agent/service can be subdivided into a common and an individual part. The common or core part is represented by classes that are part of the Grasshopper platform, namely the classes Service, MobileAgent, and StationaryAgent, whereas the individual part has to be implemented by the agent programmer [GPG99].

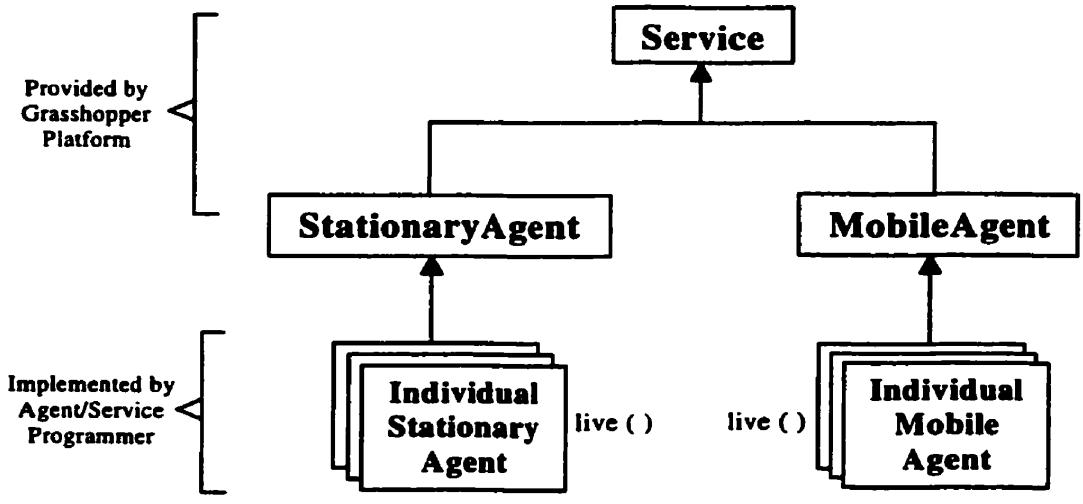
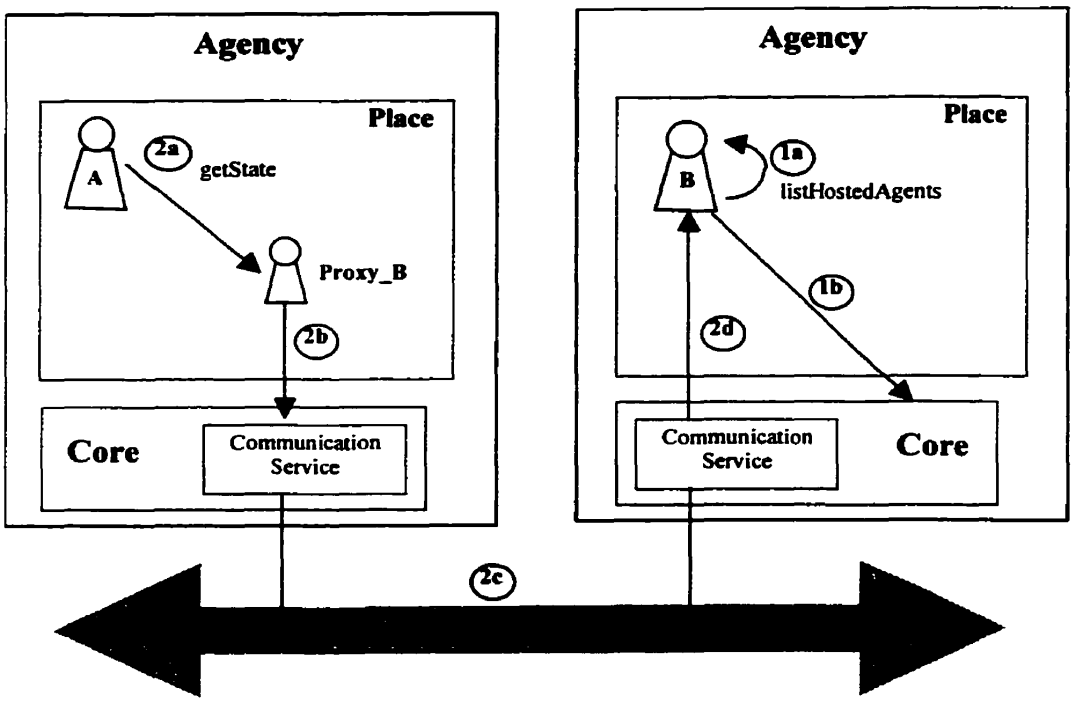



Figure 27: Agent Class Hierarchy
(Adapted from Grasshopper Programmer's Guide)

A Grasshopper agent consists of one or more Java classes. One of these classes builds the actual core of the agent and is referred to as agent class. Among others, this class has to implement the method live( ), which specifies the actual task of the agent.

The agent class must be derived either from the class StationaryAgent or from the class MobileAgent, which in turn inherit from the common super class Service. The methods of these classes represent the essential interfaces between agents and their environment. The following two ways of method usage have to be distinguished [GPS99]:

- One part of the super class methods of an agent enables the access to the local core agency. For example, an agent may invoke the method listMobileAgents( ), which it inherits from its super class Service, in order to retrieve a list of all other agents that are currently residing in the same agency.
- The remaining super class methods are defined to access individual agents. These methods are usually invoked by other agents or agencies via the communication service of Grasshopper. For instance, any agent may call the method getState() of another agent in order to retrieve information about the other agent's actual state. This way of access is not performed directly on an agent instance, but instead on an agent's proxy object.




1. Agent B invokes method on itself to access the local core agency
2. Agent A invokes method on remote agent B via B's proxy

Figure 28: Access of an Agent's Methods
(Adapted from Grasshopper Programmer's Guide)

### 7.1.2 Grasshopper Communication Service

On one hand, Grasshopper Communication Service (CS) can be used by the Grasshopper system for agent transport, or for locating entities within the DAE. On the other hand, CS can be uses by agent programmers to enable their own agents to invoke methods on remote agents, agencies, or region registries. This is done location-transparently, i.e. the agent programmer need not care about the location of the called agent. Within the agent code, remote method invocations look exactly like local method invocations on objects residing on the same Java Virtual Machine. This is achieved by means of proxy objects, which can be directly accessed by a client. The proxy object forwards the call via the ORB to the remote target object, i.e. the server.

In order to use the communication service between agents, the following steps have to be performed. Step a) and b) are performed at the server side, and step c) are performed at the client side.

a) Implementation of a server object in the server side.
   1. At first the ExampleAgent.java file has to be generated.
   2. Compilation of the server source file ExampleAgent.java with javac compiler,
      Result = *ExampleAgent.class*

b) Creation of a proxy object, which is a stub, for the server. For this purpose, Grasshopper provides a special tool called *stub generator*.
   3. Execution of stubgen with ExampleAgent.class
      Result = *ExampleAgentP.java*
   4. Compilation of the proxy object ExampleAgentP.java with javac
      Result = *ExampleAgentP.class*

c) Implementation of the client object using ExampleAgentP class. The client can now interact with the server by creating an instance of the proxy and invoking methods locally on this instance.

### 7.1.3 Communication with a Region Registry

As mentioned before, a region is a logical entity, which groups a set of agencies that belong to the same authority. Associated with each region is one region registry. The main task of the registry is to allow the localization of agencies, places, and agents/services in the scope of the region [GPG99]. In order to achieve this task, the DAE components have to be registered within the registry. During the registration, certain information associated with the respective component is delivered to the registry. For agencies and plaees, the registration procedure has to be performed only once, i.e. at their creation time. In contrast to this, mobile agents require this procedure after each migration process, i.e. a de-registration at their source agency and a registration at their destination agency.

The registration and de-registration of DAE components, including agencies, places, and agents, is performed automatically, i.e. transparent to platform users as well as to

agent programmers. Thus, the only operation that is of interest when implementing agents is the localization of DAE components.

The listing below shows how an object can establish a connection to a region registry and get the location of a specific mobile agent.

1. *Proxy of the region registry*

   RegionRegistrationP r;

2. *Identifier of the region registry*

   String id = "de.ikv.grasshopper.region.RegionRegistration";

3. *Location of the region registry*

   String l = "socket://goldfinger.ikv.de:6779/GHRegistry";

4. *Create the proxy*

   r = new RegionRegistrationP (id, l);

5. *Create id object of an agent from a string*

   agentId = new Identifier (idString.getBytes( ));

6. *Create ServiceInfo object for search constraints*

   ServiceInfo i = SearchConstraints.createServiceInfo (agentId);

7. *Communicate with region registry*

   ServiceInfo [] answer = r.lookupServices (i);

8. *Get location of the agent*

   Location agentLocation = answer[0].getLocation( );

### 7.1.4 Communication with a Remote Agency

On one hand, programmers may need to access the functionality of remote agencies, e.g. in order to create an agent by means of a CGI-script invoked via a Web site or other third-party programs. On the other hand, also agents may want to contact remote agencies, e.g. in order to retrieve information about remotely hosted agents [GPG99].

The listing below shows how an object can establish a connection to a remote agency and create a mobile agent on the remote agency via the proxy.

1. *proxy of the remote agency*

   AgentSystemP a;

2. *identifier of the agency*

String id = "de.ikv.grasshopper.agency.AgentSystem";

3. *location of the agency*

   String l = "socket://goldfinger.ikv.de:6789/AgentSystem";

4. *create the proxy of the remote agency*

   a = new AgentSystemP (id, l);

5. *create an agent on the remote agency via the proxy*

   String class = "de.ikv.grasshopper.example.MoveMeAgent";

   String codebase = "http://fiesta.ikv.de/grasshopper";

   String place = "InformationDesk";

   String initializers [] = null;

   ServiceInfo i = a.createService (class, codebase, place, initializers);

### 7.2 Using Grasshopper to Realize SMU Functionalities

SMU is a stationary piece of code, which can use Grasshopper to create agents (ASA and BSA), dispatch them, call them back and so on. SMU can be realized as stationary agent or pure user application, but only agent can access the functionality of the local agency, so we implement SMU as SMUAgent class derived from the class StationaryAgent to enhance the functionality of SMU agency. The main functionality of SMU is realized by overridden the live ( ) method of StationaryAgent.

To create SMU agent, agent provider should access SMU Agency Console and click on Agent->creat Agent, then specify agent Classname, Codebase, and Place in which the agent shall be created.

ASA and BSA are mobile agents implemented by specific agent class derived from the class MobileAgent. The main functionality of ASA and BSA for originating and terminating services are implemented by overridden the live ( ) method of MobileAgent.

ASA and BSA are created by SMU. The SMU agent can access the functionality of the SMU Agency by invoking the methods of its super classes Service, StationaryAgent, and MobileAgent. During its execution, the SMU agent invokes createService method of its super class to create a new mobile agent - ASA or BSA – according to the end users' requirements. The required parameters of createService method are the class name

associated with the agent, the code base, the name of the place in which the agent shall be created etc.

Except for creating mobile service agent, dispatch them, and call them back, the SMU also needs to access to region registries to look up a specific agent in order to distinguish agent creation and agent update. Access to region registry is possible via the Grasshopper communication service. The provided interface for region registry is RegionRegistrationP.

First SMU agent should have an action( ) method. In this method, there is only one statement: live( ), which initiates the real work of SMU. Therefore, when people double click on the icon of SMU agent, the functionality of SMU agent will resume.

The live( ) method is the most important method to provide the SMU functionality. We implement the live( ) method by programming several support method in the same class and each support method realizing one functionality. The lookup( ) method is used to search for a specific agent in the region register; the create( ) method is used to create mobile service agent for a specific user by retrieving service components from SIR, and move( ) method is used for mobile service agent dispatch and call back.

### 7.2.1 Using lookup ( ) Method to Localize Mobile Agent

The lookup (agentname) method shows how an object can establish a connection to a region registry and get the location of a specific mobile service agent. The lookup method returns the location of the specific agent. If the returned location is empty, then the agent specified by the agent name does not exist in this region.

```
Location lookup (String agentname)
{
1. Create the proxy of the region registry through its identifier and location parameter
   RegionRegistrationP r = new RegionRegistrationP
                              ("de.ikv.grasshopper.region.RegionRegistration",
                               "socket://142.133.23.154:7000/GHRegistry");
2. Create id object of an agent from its name
```

agentId = new Identifier (agentname.getBytes( ));

3. *Create ServiceInfo object for search constraints*

   ServiceInfo i = SearchConstraints.createServiceInfo (agentId);

4. *Communicate with region registry and localize the specific agent*

   ServiceInfo [] answer = r.lookupServices (i);

5. *Get location of the agent*

   Location agentLocation = answer[0].getLocation( );

6. *Return the specific agent location*

   return agentLocation

}

### 7.2.2 Using create( ) Method to Create Mobile Agent

Before calling the create( ) method, the SMU will prompt the user to choose which mobile service agent (ASA or BSA) they want to create. If the user chooses ASA, a window will appear to ask them to select one or more AIN services (CCA, BNT, PNP, OCA, OCR) for the ASA. If the user chooses BSA, a window will appear to ask them to choose one or more BIN services (SCA, SCR, and SCF) for the BSA.

After the user selects the agent type and services, the following steps will be executed to create a user service agent. We assume user1 want to create an ASA with CCA, BNT and OCA services.

ServiceInfo create (String username)

{

1. *SMU sends a request to the lookup service to download the proxy code of the services that the user has requested. The proxy code contains the interfaces that a gatekeeper needs to construct a ASAgent class, and the location of the SIR where to find the actual service code.*
2. *According to the user name and service proxy code, SMU retrieves the service components from SIR and constructs a ASA agent class file specific to that user and puts it in the same directory as SMUAgent class file. Therefore, user1_ASA.class is the class file that SMU constructed to realize AIN services for user1.*

3. *Get the location of the ASA class file*

   String codebase = this.getInfo( ).getCodebase( );

4. *Create the ASA for user1*

   ServiceInfo info = this.createService ("user1_ASAgent", codebase, "SMUplace", null);

5. *Return ServiceInfo of the created ASA*

   return info

}

### 7.2.3 Using move( ) Method to Dispatch and Call Back Mobile Agent

The move( ) method receives ServiceInfo and a string of remote address as parameter, and moves a specific mobile service agent to a specific address through Grasshopper communication service. This method is useful for mobile service agent dispatching and calling back. The procedure of move method for a specific ASA is as follows:

void move (ServiceInfo info, String remoteAddress)

{

1. *Get the id of the specific ASA agent*

   String ASAid = info.getIdentifier( ).toString( );

2. *Create proxy of ASA service agent*

   ASAgentP asa = new ASAgentP (ASAid);

3. *Transfer the String parameter of the remote address into a Location parameter*

   Location remote = new Location (remoteAddress);

4. *Move the mobile service agent through its proxy object*

   asa.move(remote);

5. *Exit and return*

   Return;

}

Using the functionality described above together with functionality directly provided by Grasshopper agency, SMU could realize subscription phase scenarios for agent creation and update.

### 7.3 Using Grasshopper to Realize Subscription Phase Scenarios

Whenever an end user wants to subscribe some service agent, the SMU first checks if an agent for that user already exists using lookup( ) method. If the agent does not exist, it creates a new mobile service agent with user subscribed services immediately. If an old agent already exists for that user, but the user's new subscribed services are partly different from the services in the old one, SMU calls back the old mobile service agent, destroys it, and then creates a new service agent with new subscribed services. The service creation is realized by using create( ) method. Finally, the SMU dispatches the service agent by calling the move( ) method. If the subscribed services are same as the services in the old agent, then SMU informs the end user that there already exists a mobile service agent for him.

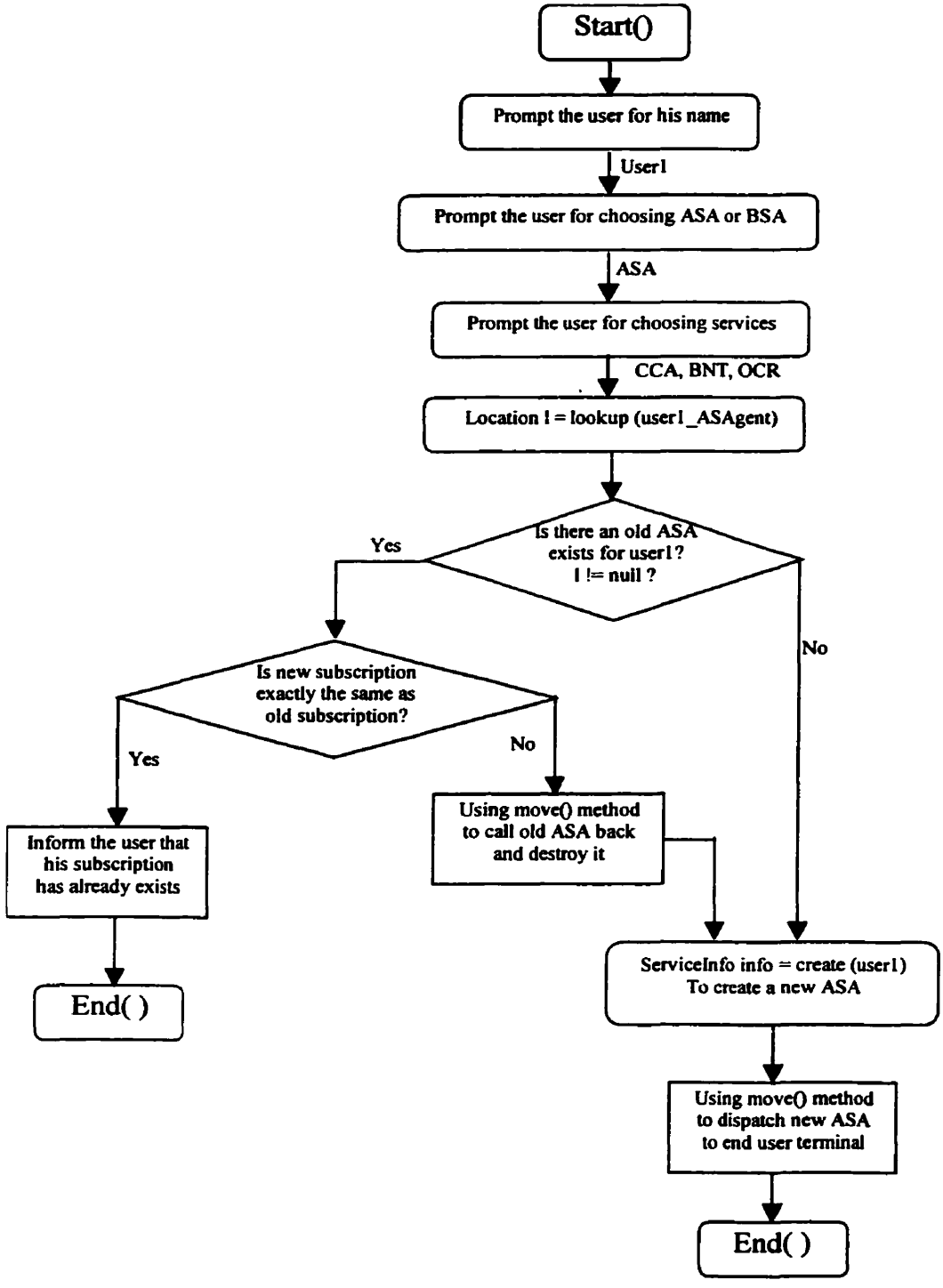



Figure 29: Subscription Phase Scenarios of ASA Creation and Update

The following is the SMU subscription phase scenario for agent update using Grasshopper platform:

1. The end user accesses SMU stationary agent and wants to subscribe an ASA for originating services. The SMU stationary agent will pop up a window, prompt the user for his name, and ask him to choose one service agent (ASA or BSA). The user enters his name and selects the agent he wants to subscribe by clicking on the agent name: ASA, then selects one or more services under that agent name.
2. The SMU checks with the region registry by calling lookup (agentname) method described in section 7.2.1. The lookup( ) method returns the location of the agent if the agent exists. If the agent doesn't exist, the return value is null.
3. If the return value is not null, which means that a service agent already exists for that user. If the new subscribed services are partly different from the services in the old agent, the old service agent will be called back by the move( ) method described in section 7.2.3. The move( ) method worked though Grasshopper communication service, so it is location transparent. It can work on an agent in the same region no matter where the agent is.
4. The old agent will be destroyed in the agency by call the remove method on that agent, and a new agent with the same name will be created by create( ) method described in section 7.2.2.
5. Finally, SMU sends the new mobile service agent to the end user agency at a terminal in the subscriber's home domain by calling the move ( ) method described in section 7.2.3.

Grasshopper platform not only can support subscription phase scenarios proposed by MA concept architecture in chapter 5, but it can also support execution phase scenarios and service mobility scenarios in a similar way.

# Chapter 8

# Conclusion and Future Work

In this thesis, first we proposed a novel conceptual MA architecture for SIP based IP telephony. Then we provided scenarios to show how this MA architecture works. Finally, we implemented this architecture using Grasshopper platform.

In the conceptual architecture, we use two kinds of mobile service agents, ASA and BSA, to provide originating services and terminating services separately. Originating services are normally provided to the user agent client where the user is currently at, and terminating services should be provided to the proxy server of the user's current domain. Because originating and terminating services should be provided to different locations in the network, defining two mobile service agents, ASA and BSA, can facilitate the service provision and service mobility.

Service mobility is an important issue in IP telephony environment. An Internet user may migrate within domain or between domains, and uniform services should be provided to that user no matter where he migrates. Therefore, there are two kinds of service mobility: service mobility within domain, and service mobility between domains. Because of the advantages of mobile agent, service logic and data can follow the end user to the new location, and fulfill both kinds of service mobilities.

Our mobile service agent (ASA or BSA) consists of Service Interaction Manager and Service Logic & Data. The Service Interaction Manager is the main method that defines which service class should be executed first, which service class should be executed next, and which one should be executed last. Service Logic & Data contains many classes, and each class defines one kind of service contained in the mobile service agent.

In our MA service architecture, there is a Service Management Unit (SMU) to manage service subscription and update. It may co-locate with a proxy server, or completely separated from the proxy server. If a service would be dynamically upgraded, the SMU would be involved.

After proposing the conceptual MA architecture, we demonstrated the scenarios in chapter 6 to show how this MA architecture works. MA service scenarios include subscription phase scenarios, execution phase scenarios, and service mobility scenarios. Subscription phase scenarios include scenarios for service agent creation and scenarios for service agent update. MA service mobility scenarios include service mobility within domain and service mobility between domains. Through those service mobility scenarios, we can see that mobile agents have great advantages in flexible and on demand service provision.

After proposing the MA conceptual architecture and scenarios for SIP based IP telephony, we introduced the Grasshopper platform to implement the MA service architecture and realize those scenarios.

Concerning the implementation architecture, because of time limitations, we only described in detail the SMU functionality, and the mechanisms it used to realize subscription phase scenarios. More time and effort are required to look into the implementation architecture for realizing execution phase scenarios and service mobility scenarios in the future.

**References**


[AAP99] H. Schulzrinne, and J. Rosenberg. Internet Telephony: Architecture and Protocols: an IETF perspective. *Computer Networks and ISDN systems, vol.31, Feb. 1999.*

[ASA99] Roch H. Glitho. Advanced Services Architectures for Internet Telephony: State of the Art and Prospects. *Submitted for eventual publication in IEEE Network (Special issue on Internet Telephony).*

[ATS98] H. Schulzrinne, and J. Rosenberg. The Session Initiation Protocol: Providing Advanced Telephony Services across the Internet. *Bell Labs Technical Journal, vol. 3, October-December 1998.*

[AVC96] H. Schulzrinne. RTP profile for audio and video conferences with minimal control. *RFC 1890, IETF, Jan. 1996.*

[CARL99] Jingrong Tang, Tony White, and Bernard Pagurek. Advanced Service Architectures for H.323 Internet Protocol Telephony. *A report to Ericsson from Systems and Computer Engineering, Carleton University.*

[CCS98] H. Schulzrinne, and J. Rosenberg. SIP call Control Services. *Internet Draft, IETF, March 1998.*

[CMS88] CMS 8800 Wireless Intelligent Network. *Product Line Description, Ericsson.*

[GRA98] M. Breugst, I. Busse, S. Covaci, and T. Magedanz. Grasshopper: A Mobile Agent Platform for IN Based Service Environments. *IEEE IN Workshop, Bordeaux, France, May 10-13, 1998.*

[GBC99] IKV++/GMD. Grasshopper Development System: Basics and Concepts. *http://www.ikv.de/download/index.html*

[GPG99] IKV++/GMD. Grasshopper Development System: Programmer's Guide. *http://www.ikv.de/download/index.html*

[IA96] T. Magedanz, K. Rothermel, and S. Krause. Intelligent Agent: An Emerging Technology for Next Generation Telecommunications. *IEEE INFOCOM, 1996, San Francisco, CA, Mar. 24-28, 1996.*

[IINS99] J. Lennox, H. Schulzrinne, and T. F. La Prota. Implementing Intelligent Network Services with the Session Initiation Protocol. *Columbia University Computer Science Technical Report, January 1999.*

[IMA98] Makrus Breugst, Lars Hagen, and Thomas Magedanz. Impacts of Mobile Agent Technology on Mobile Communications system Evolution. *IEEE Personal Communication Magazine, August 1998.*

[IN96] Thomas Magedanz and Radu Popescu-Zeletin. Intelligent Networks. *International Thomson Computer Press.*

[Index] Intelligent Network. http://www.webproforum.com/bell-atlantic/index.html. *Bell Atlantic.*

[INO93] James J. Garrahan, Peter A. Russo, Kenichi Kitami, and Roberto Kung. Intelligent Network Overview. *IEEE Communications Magazine, March 1993.*

[IOD96] S. Krause, T. Magedanz. Mobile Service Agents enabling "Intelligence on Demand" in Telecommunications. *Proceedings of the IEEE Global Telecommunications Conference, p.78-85, IEEE Press, 1996.*

[IW98] Lorenzo Faglia, Gennaro Marino, and Fabrizio Aizza. The Interworking of IN and B-ISDN and the Introduction of Mobile Agents Technology. *http://www.italtel.it/drsc/marine/marine.htm*

[MAS98] Markus Breugst, and Thomas Magedanz. Mobile Agents – Enabling Technology for Active Intelligent Network Implementation. *IEEE Network Magazine, Special Issue on Active and Programmable Networks, May/June 1998 Vol 12 No.3.*

[MSA98] Vu Anh Pham, and Ahmed Karmouch. Mobile Software Agents: An Overview. *IEEE communications Magazine, July 1998.*

[NM98] A. Bieszcazad. Mobile Agents for Network Management. *IEEE Communications Survey, Third Quarter.*

[RSVP97] B. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource reSerVation Protocol (RSVP). *RFC 2205, IETF, Oct. 1997.*

[RTP96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. *RFC 1889, IETF, Jan. 1996.*

[RTSP98] H. Schulzrinne, R. Lanphier, and A. Rao. Real time streaming protocol (RTSP). *RFC 2326, IETF, Apr. 1998.*

[SFIT98] H. Schulzrinne, and J. Rosenberg. Signaling for Internet Telephony. *In Proc. Of 6th IEEE International Coference on Network Protocols, Oct. 1998.*

[SIP99] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg. SIP: Session Initiation Protocol. *Internet Draft, IETF, Jan. 1999.*

[SNA98] Ch. Z. Patrikakis, S. E. Polykalas, and I. S. Venieris. Service Node Architectures Incorporating Mobile Agent Technology and CORBA in Broadband IN. *http://www.italtel.it/drsc/marine/marine.htm*

[SUA98] H. Schulzrinne. Requirements for SIP servers and User Agents. *Internet Draft, IETF, Nov. 1998.*

[TIOD96] T. Magedanz, and R. Popescu-Zeletin. Towards "Intelligence on Demand" – On the Impacts of Intelligent Agents on IN. *4th International conference on Intelligence Networks, Bordeaux, France, November 25-28, 1996.*

[WINS] Wireless Intelligent Network Services. *Product Line Description, Ericsson.*