Handling Feature Interactions In the SIP Servlet Context

Lu Yang

A Thesis

In

The Department

Of

Electrical and Computer Engineering

Presented as Part of the Requirements

For the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

June 2005

Your file Votre référence
ISBN: 0-494-10257-8
Our file Notre référence
ISBN: 0-494-10257-8

ABSTRACT

Handling Feature Interactions in the SIP Servlet Context

Lu Yang

Session Initiation Protocol (SIP) and Servlet technology introduce new ways of delivering telephony services over IP networks. The richness and flexibility of these new protocols make it faster and easier for service providers to develop and deploy new services. This flexibility, however, is offset by the challenge of managing the feature interaction problem, which can prove to be quite severe.

This thesis proposes a modified SIP Servlet architecture and introduces a logical entity, the Feature Interaction Handler (FIH), to address the feature interaction problem for telephony services. The approach addresses *offline* and *online* feature interaction detection, the former occurring when the user registers to a feature and the latter occurring during feature runtime execution. For offline feature interaction detection, a behaviour mapping approach is introduced to reduce the interaction matrix table. For online feature interaction detection, two mechanisms are proposed – "forward detection" and "backward detection".

Forward detection extends the originating side user profile when sending the message, such that the terminating side can use it for detection. In contrast, backward detection correlates the SIP session "request" message with the "response" message belonging to the same session in order to determine if the resulting service behaviour is acceptable.

To validate the new feature interaction detection approach, an offline feature interaction detection tool and online feature interaction detection unit FIH have been implemented. The feature interaction benchmark is applied on both the tool and the FIH, the result proves to be successful.

The feature interaction detection approach proposed in this thesis proves to be a viable solution in the context of SIP servlet service environment.

Acknowledgements

I would like to take this opportunity to express my great respect and sincere thanks to my supervisor, Dr. Ferhat Khendek, without his professional guidance and encouragement this research would not have been possible.

I also wish to thank all my colleagues at Ericsson and friends in the Concordia University for the help they have provided during the course of this project.

Most important of all, my deepest gratitude to my dear wife Yvonne for her support and advice. This thesis is dedicated to our son Patrick.

TABLE OF CONTENTS

– LIST OF FIGURES.......ix

– LIST OF TABLES.......x

– LIST OF ACRONYMS.......xi

1 INTRODUCTION.......1

1.1 Evolution Of Telecommunication Networks.......1
1.2 Feature Interaction Problem.......4
1.3 Contributions Of This Thesis.......6
1.4 Organization Of The Thesis.......7

2 FEATURE INTERACTIONS IN IN AND IP NETWORKS.......8

2.1 Introduction.......8
2.2 Categories Of Feature Interaction.......10
2.3 Feature Interaction In Intelligent Networks.......12
2.3.1 Detection.......16
2.3.2 Resolution.......18
2.3.3 Prevention.......20
2.3.4 Management.......20
2.4 Feature Interaction In IP Networks.......21
2.4.1 Architectural Model.......22
2.4.2 Difference Between PSTN And IN.......24
2.4.3 Application Of Existing Feature Interaction Techniques.......27
2.5 Chapter Summary.......28

3 SIP AND SIP SERVLET ARCHITECTURE.......29

3.1 Introduction.......29
3.2 Session Initiation Protocol.......30
3.2.1 SIP Session.......31
3.2.2 SIP Message Format And Parameter.......33
3.2.3 Request Message.......35
3.2.4 Response Message.......37
3.2.5 Header Field.......39
3.3 SIP Servlets.......44
3.3.1 SIP Servlet Overview.......46
3.3.2 SIP Servlet API Architecture.......47

3.3.3 Servlet Engine *49*
3.3.4 User Agent *49*
3.4 Chapter Summary 50

4 FEATURE INTERACTION DETECTION IN A SIP SERVLET ENVIRONMENT 51

4.1 Introduction 51
4.2 General Service Architecture 52
4.3 Offline Detection Technique 55
4.3.1 Mapping *57*
4.3.2 Analysis *58*
4.3.2.1 Forwarding and Delay 59
4.3.2.2 Delay and Multi-party 59
4.3.2.3 Feature Category Interactions Matrix 60
4.3.3 Resolution *61*
4.4 Applying Feature Category Interactions Matrix To Two Users 61
4.4.1 Forwarding And Authentication *62*
4.4.2 Modified Feature Category Interactions Matrix *63*
4.5 Online Detection Technique 63
4.5.1 Enhanced SIP Servlet Architecture *64*
4.5.2 Backward Detection *66*
4.5.2.1 Overview of Backward Detection Technique 66
4.5.2.2 Algorithm for Backward Detection Technique 68
4.5.2.3 Example of Authentication vs. Forwarding 70
4.5.2.4 Example of Delay vs. Delay 74
4.5.3 Forward Detection *76*
4.5.3.1 Overview of Forward Detection Technique 76
4.5.3.2 Forward Detection Header Info Only 77
4.5.3.3 Example of Delay vs. Forwarding 81
4.5.3.4 Example of Multi-party vs. Forwarding 82
4.5.3.5 Forward Detection Header Info and Incoming Leg 83
4.5.3.6 Example of Forwarding vs. Authentication 86
4.6 Chapter Summary 87

5 IMPLEMENTATION 88

5.1 Introduction 88
5.2 Implementation Architecture 88
5.3 Online Feature Interaction Detection Architecture 92
5.3.1 Interface Description *93*
5.3.2 Function Description *94*

5.3.3 FIH Structure ... *95*
5.3.3.1 Transaction Manager ... 97
5.3.3.2 Call Leg ... 99
5.3.3.3 User Profile ... 100
5.3.3.4 Feature Category Interactions Matrix ... 100
5.3.3.5 FIH Class Structure ... 101
5.3.4 Feature Servlet ... *102*
5.4 Offline Tool Architecture ... 104
5.4.1 Offline Tool Class Structure ... *104*
5.5 Programming Environment ... 105
5.6 Application ... 106
5.6.1 Bellcore Benchmark ... *106*
5.6.2 European Benchmark ... *107*
5.6.3 FIH Application ... *107*
5.6.4 Offline Tool Demonstration ... *112*
5.5 Chapter Summary ... 115
6 CONCLUSION ... 116
6.1 Main Contributions ... 116
6.2 Main Advantages Of Our Approach ... 117
6.3 Potential Extensions And Future Work ... 118
REFERENCES ... 120

LIST OF FIGURES

Figure 1 Categorization of approaches to address FI........................14
Figure 2 Classification of Approaches 16
Figure 3 SIP Servlet API Architecture...48
Figure 4 Service Architecture Without FIH 53
Figure 5 Service Architecture With FIH...54
Figure 6 Enhanced SIP Servlet Architecture....................................65
Figure 7 Backward Detection Algorithm 69
Figure 8 SIP Call Forwarding Signal Flow 71
Figure 9 SIP ARC And ACB Flow 75
Figure 10 Forward Detection Header Info Only 80
Figure 11 Forward Detection – Camp On And CFOB........................81
Figure 12 Forward Detection – Forking And Voice Mail 83
Figure 13 Forward Detection Header Info And Incoming Leg...........85
Figure 14 PROXY Stack...89
Figure 15 End-to-End Call Flow Stack View....................................90
Figure 16 PROXY Stack With FIH 91
Figure 17 End-to-End Call Flow Stack View With FIH 92
Figure 18 FIH Structure..96
Figure 19 FIH Class Structure...101
Figure 20 Servlet Structure...103
Figure 21 Offline Tool Class Structure..105
Figure 22 Offline Tool 113
Figure 23 Offline Tool Dialogue 114
Figure 24 Offline Tool Usage Example...114

LIST OF TABLES

Table 1 SIP Message Header Field 42
Table 2 Feature Interaction Category Matrix 60
Table 3 Modified Feature Interactions Category Matrix 63

LIST OF ACRONYMS

3GPP	3rd Generation Partnership Project
API	Application Programming Interface
CW	Call Waiting
CFOB	Call Forwarding On Busy
CFON	Call Forwarding On No-reply
CFU	Call Forwarding Unconditional
CGI	Common Gateway Interface
CPL	Call Processing Language
CW	Call Waiting
FI	Feature Interaction
FIH	Feature Interaction Handler
GUI	Graphical User Interface
HTML	Hyper Text Marking Language
HTTP	Hyper Text Transfer Protocol
IETF	Internet Engineering Task Force
IN	Intelligent Network
IP	Internet Protocol
ISDN	Integrated Service Digital Network
ISO	International Organisation for Standardization
ITU-T	International Telecom Union Telecommunication Sector
JVM	JAVA Virtual Machine

NNI	Network Network Interface
OCS	Outgoing Call Screening
PSTN	Public Switch Telephone Network
RTP	Real-Time Transport Protocol
SCTP	Stream Control Transmission Protocol
SDP	Session Description Protocol
SIP	Session Initiation Protocol
TCP	Transport Control Protocol
TCS	Terminating Call Screening
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecom System
UNI	User Network Interface
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
UTF	UCS Transformation Format

Chapter 1

INTRODUCTION

1.1 Evolution Of Telecommunication Networks

With the development of modern telecommunications and the introduction of stored program control exchanges, value added services and features have become increasingly important to both individuals and business users. Some typical examples, such as call forwarding or call waiting, are widely used features today.

Features offered by the Public Switched Telephone Network (PSTN) have traditionally been implemented using vendor-proprietary solutions in a closed environment. This has resulted in a very long lead-time for feature deployment and has also sometimes resulted in similar features having very different behaviours. The Intelligent Network (IN) concept of service and feature development evolved as a means to solve this problem.

An IN is a service-independent telecommunications network. Its basic premise involves removing the intelligence from the switch and placing it instead in computer nodes distributed throughout the network. Because the functional behaviour and interfaces are standardized, new capabilities can be rapidly introduced into the network, providing the network operator with an efficient means of developing and controlling services. Once introduced, services can also easily be customized to meet individual customers' needs [1].

The key benefits of Intelligent Network solutions include [2]:

- A wide range of feature-rich services;
- Fast implementation of revenue-generating solutions;
- An easy-to-use Service Design Environment (SDE); and
- High capacity.

In the late 90's, with the deployment of World Wide Web and data services, many service providers started considering the use of their IP networks for transporting voice as well as data. Internet Telephony emerged. For service providers, one of the main objectives was to ensure that all the features supported by modern telephony systems could also be supported using Internet Telephony.

The architectural model of Internet Telephony is rather different than that of the traditional telephone network. The basic premise is that all signalling and media flow over an IP-based network, which makes use of the public Internet or various intranets. This provides a drastic change in the ability of nodes in the network to communicate. In traditional telephone architectures, nodes can generally communicate with only the 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, unless the network specifically restricts it from doing so through means such as firewalls [3].

This change necessitates a major transformation in the architectural assumptions underlying traditional telephone networks. In particular, in a traditional network, a large amount of administrative control such as call-volume limitation implicitly resides at every switch, and additional controls can easily be added there without much architectural impact. In an Internet environment, in contrast, an administrative point of control such as a firewall must be explicitly engineered into the network to prevent end systems from simply bypassing any device, which attempts to constrain their behaviour.

In addition, the Internet model transforms the location where many services can be implemented and executed. As a general principle, end systems are assumed to be much more intelligent than in the traditional telephony model, with the result that services which had resided within the network are moved out to its edges without requiring any explicit support within the network itself. Other services can be supported by widely separated specialized servers, which results in call setup information traversing paths which might be indirect when compared with the actual network's physical topology [4].

A set of Internet Telephony protocols, including Session Initiation Protocol (SIP) [5] and H.323 [6], has been defined by different standardization organizations. H.323 is defined by the International Telecommunications Union (ITU) and SIP has been defined by the Internet Engineering Task Force (IETF). SIP in particular is gaining momentum in the industry and has been selected by the 3GPP third generation wireless core network standard forum [7].

SIP is a client/server protocol for multimedia call control and services. It is highly scalable and fairly simple, such that applications and features can be readily built on top of it [8]. Most of the telephone features and service features of Intelligent Network standards can be provided by the SIP.

In 1999, an IETF Internet draft proposed a new JAVA Application Programming Interface (API) for use with SIP servers and user agents. It defined the abstractions necessary to allow a SIP protocol stack to defer some of its decision-making for the handling of SIP requests and responses to SIP "Servlets", Java classes that implement the SipServlet interface. SIP Servlets can inspect and set message headers and bodies, and can proxy and respond to requests as well as initiate their own requests and forward responses upstream. They can reside on top of a SIP protocol stack in network elements such as proxies, registrars, redirect servers, and user agents.

SIP Servlets provide a new way of implementing services or features in IP networks. Each individual service or feature can be implemented by a Servlet, which can be developed by any service operator by following the standard API. This facilitates the compatibility issue and speeds up the feature deployment [9].

1.2 Feature Interaction Problem

So far, telecommunications service providers have been able to maintain extremely tight control over the types of equipment and services attached to their networks. Even within this closed environment, however, up to 90% of the deployment cost of a specific service can be attributed to management aspect. Despite this fact, Intelligent Networks fail to address the problem of service management concretely, and fail to provide a definition for a management interface [10].

A key issue is the Feature Interaction problem. By definition, feature interaction is a situation in which a feature or a feature set modifies or influences another feature's behaviour. Feature interaction is inevitable in a feature-oriented specification, because little can be accomplished when features are developed completely independently from each other.

There are many contributing factors:

- In telecommunications, new features are often stimulated by new technology. New technology eliminates obstacles and, in doing so, invalidates deeply entrenched assumptions underlying the features and their attributes [11].
- Most telecommunication devices are simple and standardized. All features, no matter how many of them there are, must share the same small vocabulary of input and output signals, which increases the likelihood of usage interaction [11].
- Telecommunication systems last for a long time, and grow to have hundreds or even thousands of features the volume of which increases the likelihood of undesired interactions[11].
- All telecommunication systems interoperate with features and services that have been developed by other parties, which are unknown and cannot be trusted to abide by the same rules of behaviour [11].

A lot of work has been done to deal with the feature interaction problem in PSTN and IN [12] [13]. A special network entity called "Feature and Service Interaction Management" (FIM) entity was defined in the network reference model when the Intelligent Network architecture was introduced [14]. However, feature interaction handling in the context of Internet Telephony is still new and not much work has been done to address the problem. In fact, thus far, feature interaction related to SIP call control and feature implementation is not even yet on the standards development agenda. With the growing trend of using SIP Servlets for feature implementation, it is therefore imperative to have a new feature interaction handling mechanism in place.

1.3 Contributions Of This Thesis

This thesis explores the possibilities in handling feature interaction in the context of the SIP Servlet environment. It addresses the issue of detecting feature interaction when SIP Servlets are used for feature implementation and provides relevant management solutions to detect, resolve and manage the feature interaction. An offline feature interaction detection method based on the mapping between individual feature and general feature category is developed. A new online detection technique, based entirely on the "mandatory" SIP parameters associated with the SIP messages used in basic call handling, is also introduced.

1.4 Organization Of The Thesis

This thesis is organized as follows:

Chapter 2 provides an overview on the feature interaction handling technique. It puts different detection and management mechanisms into perspective. It also outlines the feature interaction differences between IN and IP networks.

Chapter 3 gives an overview of SIP and SIP Servlets. The feature handling in the context of SIP and SIP servlet is detailed. The feature's signalling diagram in a SIP and SIP Servlet environment is also reviewed.

Chapter 4 presents our detection techniques as well as an architectural view of feature interaction handling in the context of SIP Servlet environment. It includes both offline and online feature interaction detection. Within online feature interaction detection, two methods, backward and forward are explored. For offline feature interaction detection, a feature to "feature category" mapping approach is introduced.

Chapter 5 discuss the implementation of offline feature interaction detection tool and online feature interaction detection unit FIH. Both the offline tool and the online FIH unit are validated using feature interaction benchmark.

Chapter 6 summarizes the contributions proposed in this thesis as well as the advantages of our methods. It also provides some thoughts and ideas for future work.

Chapter 2

FEATURE INTERACTIONS in IN AND IP NETWORKS

2.1 Introduction

The feature interaction problem is a major obstacle to the mass deployment of some telecommunications services. With the introduction of IP networking and the distribution of service control, the severity of the problem grows. This chapter provides an overview of different feature interaction techniques, and discusses some of the differences between IN and IP networks that are relevant to feature interaction.

The term "Feature Interaction Problem" in telecommunications systems was coined in the early 1980's by Bellcore. *Bowen et al* made the first efforts to address this large problem by providing a framework in 1988 [15]. Since then, researchers from academia, research centers, and industry have been working towards a viable solution [12]. The work carried out in this area has already grown enormously, with many new aspects of the problem discovered in relation to PSTN, ISDN and IN services. In the late 1990's, when IP telephony started to emerge, a new question came to light: would the existing mechanisms be capable of handling the new network paradigm?

Because an imprecise definition of the terms used in this field is sometimes a source of confusion, a few important terms are defined and summarized here [12]:

- ***Supplementary Service*** - In ITU-T terminology, a *supplementary service* “modifies or supplements a basic telecommunication service. It must be offered together with or in association with a basic telecommunication service. The same supplementary service may be common to a number of telecommunications services." Well-known examples of supplementary services are Call Forwarding Unconditional (CFU), Call Waiting (CW), and Camp on Busy (CAMP-ON).
- ***Service Feature or Feature*** - The term *feature* is defined as a "unit of one or more telecommunications or telecommunications management-based capabilities a network provides to a user”. ITU-T defines the term *service feature* as “the smallest part of a service that can be perceived by the service user”. In that context, the term service refers to a telecommunication service.
- ***Feature Interaction*** - Using the definition above, the term *feature interaction* refers to situations where different service features or instances of the same service feature affect each other. This term is used in most publications as the most general term describing the problem.

Feature interaction handling in traditional PSTN/IN networks and in new IP environments can be very different. The following section explores how feature interaction is handled in the PSTN/IN environment, and highlights the key differences in an IP environment. The analysis first categorizes the feature interaction, then suggests methods of handling it.

2.2 *Categories Of Feature Interaction*

Feature interactions are categorized along three dimensions, in accordance to a benchmark document in this field of study [11]. The dimensions include a consideration of several factors as listed below:

- The kinds of features involved in the interaction - This dimension distinguishes interactions that involve only *customer* features from interactions that also involve *system* features.
- The number of users involved in the interaction - This dimension distinguishes *single-user* interactions from *multiple-user* interactions. Single-user interactions arise when different features simultaneously activated by a single user interfere with each other, whereas multiple-user interactions arise when features activated by one user interfere with those activated by another user.
- The number of network components in the interaction - This dimension makes a distinction between *single-component* interactions, which arise when only one network component is involved in the feature processing, and *multiple-component* interactions, which arise when features supported in one network component interfere with the operation of those supported in another network component.

The following major categories of feature interaction result [11]:

- Single-User-Single-Component (SUSC) interactions between customer features;
- Single-User-Multiple-Component (SUMC) interactions between customer features;
- Multiple-User-Single-Component (MUSC) interactions between customer features;
- Multiple-User-Multiple-Component (MUMC) interactions between customer features; and
- Customer-System (CUSY) interactions between customer features and system features.

The causes of the above-listed interaction categories vary from case to case. The major ones are the following [16]:

- **Feature assumption** – Features in a telecommunications network need to operate under a set of assumptions, including a particular call processing mode, architectural support, and / or other factors. As the network evolves, the addition of new features or changes in architecture may violate the assumptions underlying existing features, which can result in feature interaction.
- **Limited network support** - Network components and existing telecom protocols have limited capabilities of communicating with other network components or in processing calls. As a result, two seemingly independent features may come into conflict over the reception of the

same signal or the usage of the same component functionality, causing feature interaction.

- **Problem in distributed systems** - Telecom network are real-time distributed systems. Many of the difficulties seen in dealing with large distributed systems are also present in managing feature interactions in telecom networks. One obvious case is the problem of resource contention where features come into conflict because they attempt to use the same network resources. Another is that the distribution of feature support in the network and the customization of features by each individual can create interactions that require coordination.

The following sections discuss the handling of feature interaction in both traditional PSTN/IN environments and in the new Internet Telephony environment.

2.3 Feature Interaction In Intelligent Networks

The increasing demand for telecommunications services has led to a rapidly growing number of new services, as well as to the enhancement of existing services with new service features. This trend is still evident today. In order to facilitate the procurement of new services, the architectural concept of the Intelligent Network has been developed. It separates the functions of basic call processing from the execution of supplementary services and presents a well-defined interface for the realization of new services [1]. This simplifies and speeds up feature development, but increases the risk of interactions due to the larger number of new services, the lack of tool support for automatic interaction treatment, and the typical lack of substantive experience held by the service developers in this environment.

As the number of features and services grows, the amount of effort invested into the treatment of interactions explodes, because the number of possible combinations of features and services grows exponentially with their number. The interaction problem consequently becomes a dominant issue during the software development period. For the same reasons, exhaustive testing of the service and feature combinations becomes almost unmanageable. The end result is that, even with IN, efforts to avoid interaction problems are a major obstacle to the timely and cost-efficient introduction of new services - and this obstacle also increases the risk and cost to the service provider [17].

The consequences of failing to manage interactions in a network are significant. Even though feature interaction may not cause call failures in the Public Switched Telephone Network, it can bring confusion and annoyance to the end user, and can ultimately cause financial losses. However, the costs of dealing with the feature interaction during the development of a new service may be very high, perhaps even to the extent that the service does not ultimately reach the marketplace at all.

Handling the feature interaction problem adequately, given its complexity, requires both a general framework and a consistent approach. The generally accepted categorization of approaches is based on the terminology of *avoidance*, *detection*, and *resolution* as listed in Figure 1 below [18].

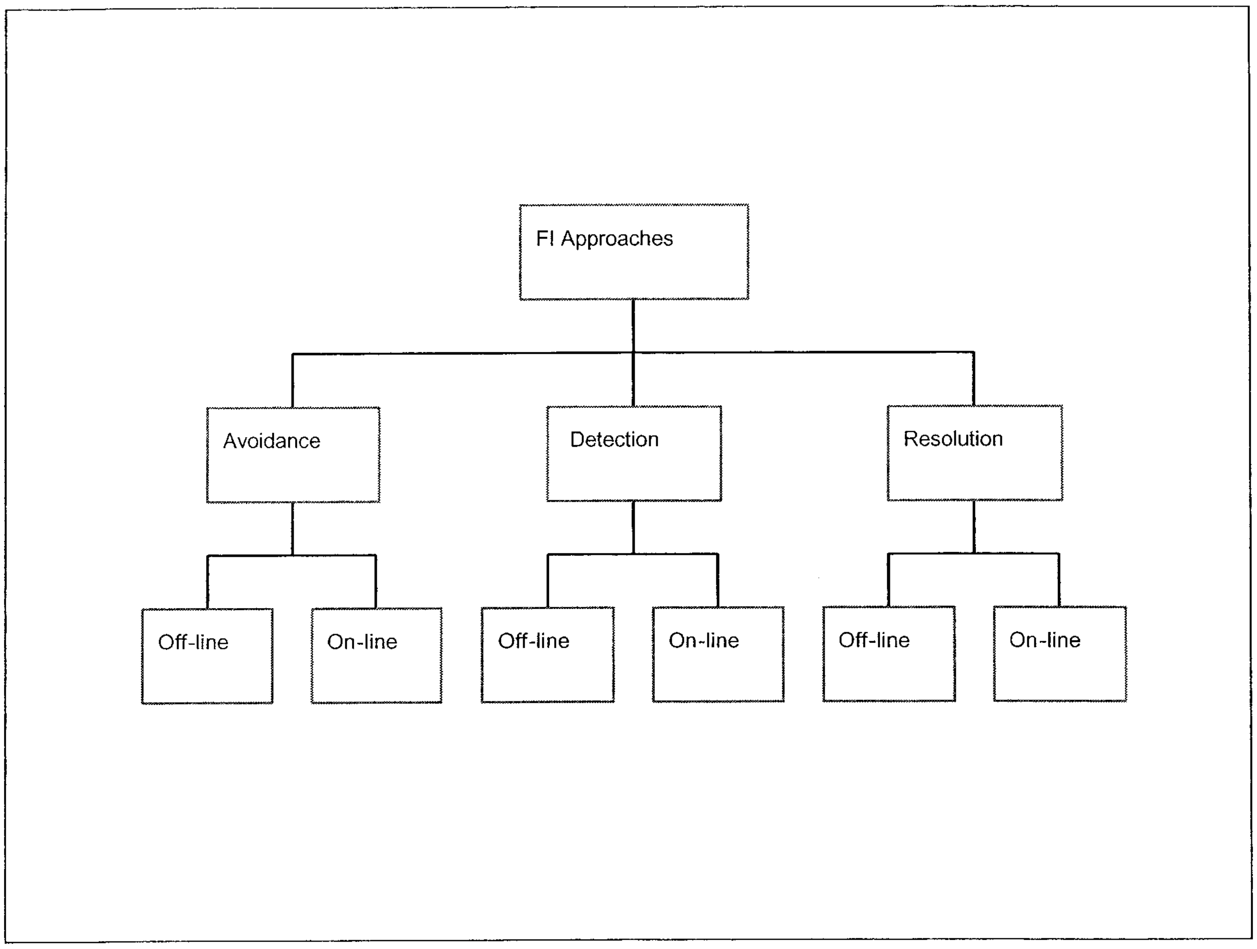


Figure 1 Categorization of approaches to address FI

The scheme is supplemented by considering the lifecycle state during which a given method applies, whether this is "off-line" or "on-line". The off-line part includes the feature's specification, implementation, and test phases. The *specification* phase can be further subdivided into more granular levels:

- The *network level specification* and the *detailed specification* corresponding to the abstract architecture in terms of the emergence level view.
- The *implementation* phase concludes the sequence of off-line service development. In the IN approach, the implementation step is fulfilled by software tools that translate a detailed specification into software code.

After an implementation is available in a test bed, careful and methodical *testing* takes place. If the tests are successful, the feature is deployed in the network, and is made available to network users who may register to use it. This concludes the off-line phase. When the feature is invoked, then the on-line phase of feature interaction begins.

Although Figure 1 has proven useful, the huge amount of research carried out in this field and the broad spectrum of methods applied suggest refinements and extensions that are summarized in Figure 2 below.

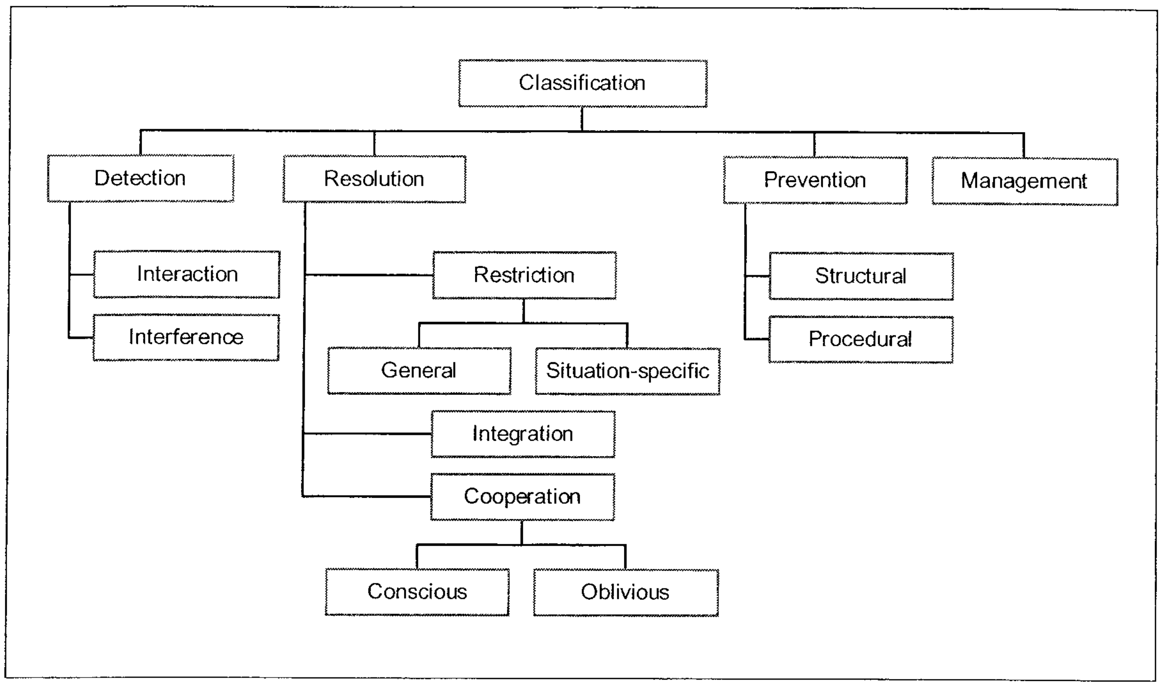


Figure 2 Classification of Approaches

2.3.1 Detection

Detection denotes the identification of the existence (presence) of interactions. Because it is practically impossible to prove the absence of mistakes, and because a given interaction is not necessarily undesired, it cannot be expected that a network is free of any interactions. The detection technique has two main tracks: formal techniques and informal techniques.

Formal techniques aim to ensure the correctness of a system by first creating a functional specification of the system and its components and then validating the specifications using verification techniques, mostly model checking. An important precondition for the successful application of formal techniques is the availability of a suitable model of the system.

Formal techniques provide suitable means to address the development of distributed systems, as they allow the choice of an appropriate abstraction level and, with proper tool support, the verification of a system's correctness, at least with respect to a number of criteria. A large amount of work in addressing feature interactions relies on the application of formal description techniques for the specification of a system and its features, in combination with formal verification. One example is "verification using general criteria". In this method, interactions are detected checking general correctness criteria such as the presence of deadlock and life lock, transitions to invalid states, ambiguities, etc. This kind of approach is termed the general property approach [19].

Methods of object-oriented software engineering, such as the application of use cases, provide another means to cope with the interaction problem. These are generally termed informal techniques. Network-level use cases are elaborated, and their analysis leads to the examination of possible feature combinations [20]. This method uses the simple but effective criterion that two features are interaction-prone if they access the same service or call specific data. The interaction-prone scenarios are then analyzed manually to determine if an interaction problem actually exists [18].

2.3.2 Resolution

Resolution is the desired consequence of feature interaction *detection.* If an interaction between two service features is detected, resolution may follow one of several different routes, namely restriction, integration, precedence or cooperation.

The simplest approach is *restriction,* which seeks to avoid situations that will cause both features to interact, without requiring either feature to modify or adapt its normal behaviour. This stands in contrast to a *precedence* approach, which holds that the behaviour of one of the feature be changed.

Two sub-approaches of restriction can be identified [18]:

- *General restriction* is characterized by a static set of rules, which disallows the activation of one feature if another feature is present.
- *Situation-specific restriction* is characterized by a set of rules that takes into account the current situation in which the second feature is activated.

General restriction is a rather rigid method, but is very often chosen by practical approaches. *Situation-dependent restriction* is more flexible, in that it allows the decision of whether or not to carry out a specific operation to be dependent on the situation encountered by the conflicting services. In contrast to general restriction, situation-dependent restriction does not affect the user's ability to activate and invoke the conflicting features.

Integration of both features into one with larger functionality is another strategy for handling interactions. On the specification level, however, it is not always easy to decide if an approach belongs to the restriction or to the integration category, as special treatment in the integration case often has integrative as well as situation-specific restrictive characteristics. The criterion used to decide this (if the method is restrictive or integrative) is whether or not it allows the possibility to clearly assign parts of the specification to one or both features.

Cooperation of the interacting features is the most sophisticated approach of resolving interactions. Beyond the simple decision used by restriction approaches to carry out or prohibit an operation, this approach tries to find a compromise, which satisfies the needs of all participating features. In the cooperation approach, the features change or adapt their behaviour in order to resolve the problem. *Conscious cooperation* is achieved if each of the features possesses explicit knowledge of the other, and if one or both are equipped with mechanisms to resolve their specific interactions. With *oblivious cooperation*, there exist general mechanisms that require no service-specific knowledge in order to resolve interactions with unknown counterparts. In this method, features or services cooperate even though they do not have any special knowledge of each other, thereby achieving one of the goals of architectures like the IN - the independence of new services development.

2.3.3 Prevention

Prevention complements the detection and resolution of feature interactions. It is used for approaches where feature interaction cannot occur. These approaches can be divided into two groups. *Structural* approaches avoid the occurrence of interactions by imposing a suitable system structure, including, for example, architecture and protocols. *Procedural* approaches attack from another direction, by introducing new strategies or restricting existing processes of service creation, for example, or by giving sets of design guidelines. This distinction has already been identified in some research work, although this has not been formulated generally.

2.3.4 Management

As the feature interaction problem is so complex, and as the number of feature combinations grows with their numbers, it is inevitable to add management issues to the list. In describing the methods that deal directly with the interaction problem, it becomes clear that no single method can cover all aspects of the problem at once. To master the problem as a whole, a sophisticated coordination of a number of different methods and means must be employed.

The research in this area is divided into two groups. The first group is concerned with the management of combinatorial complexity of the interaction problem, including tools for context generation and information acquisition. The second group comprises approaches concerning the whole interaction handling process, and involves tool support for the overall process of dealing with interactions, such as an integrated toolset for service creation, feature interaction analysis, and management [21].

2.4 Feature Interaction In IP Networks

Internet Telephony is defined as the provision of telephone-like service over the Internet. Some consider it the next stage of development of the telephone network and the first incarnation of the long held goal of an "integrated services" network. The growth of the Internet as a platform for data delivery and its rapidly increasing bandwidth make it desirable to enable creation of a telephone service that can run entirely over Internet protocols. Internet Telephony service offers the possibilities of multimedia communications, integration with other Internet services, and simplified development and operation.

In telephone networks, feature interaction occurs when several features or services, operating simultaneously, interact in such a way as to interfere with the desired operation of some of the features. This problem of feature interaction also exists in Internet telephony, and becomes an increasingly pressing problem as more and more sophisticated services are created and deployed in this environment. The large amount of work spent in understanding and resolving feature interactions in traditional telephone networks can also be put to use to understand and control interactions in Internet Telephony [22].

Internet Telephony, however, is different in many ways from the PSTN. Some of the differences help prevent or resolve feature interaction problems, as the design of new protocols and the characteristics of the underlying network eliminate some of the problems associated with legacy networks and systems. However, some of the differences in Internet Telephony introduce new types of interactions, which make legacy techniques more difficult and sometimes impossible to employ.

The following section discusses Internet telephony in terms of the Internet Engineering Task Force's (IETF's) architecture, centered on the SIP. This will be compared with the handling of feature interaction with the classical method used in the PSTN and IN networks.

2.4.1 Architectural Model

While the architecture of Internet telephony is similar to traditional telephone networks in many ways, it also has some significant differences. Most fundamentally, Internet telephony is different in that it (naturally) runs over the Internet, or more generally over IP networks. The most significant consequence of using this underlying network is that it provides transparent connectivity between any two devices on the network. Whereas devices in traditional networks are restricted to communicating with those devices to which they are directly connected, and whereas the telephony protocols themselves must handle all location and routing features, Internet telephony can rely on an underlying infrastructure to obtain all these capabilities automatically.

Within the Internet telephony network, there are usually three types of devices [23]: end systems, gateways, and signalling servers. *End systems* are the devices on which users make and receive calls. These devices initiate and respond to signalling, and transmit and receive media. They are "smart" in that they are aware of Call State, and keep track of the status of each call in which they are involved. They may provide a number of services based on this call state information; for example, *Call Waiting* is generally handled entirely in the end systems in the Internet.

Gateways are devices, which allow calls to be routed to and from other telephone networks. To other Internet telephony devices, they are not conceptually different from end systems. Like end systems, they initiate and respond to signalling, and transmit and receive media. Other devices need not be aware of the existence of the other network "behind" the gateway.

Signalling servers handle the application level control of the routing of signalling messages. They are typically used to perform user location services. A signalling server maintains information about a user's current location, and can forward or redirect call setup requests appropriately. From the point of view of feature creation, signalling servers are the devices, which most resemble service control or service switching points in the circuit-switched network. These can be programmed to direct, block, or alter call signalling messages based on their own internal logic.

2.4.2 Difference Between PSTN And IN

Key differences between IP Telephony and PSTN/IN networks revolve around the signalling protocol, networking aspects, and features [24].

On the signalling protocol side, Internet telephony is different from traditional telephone networks due to the effects of the Internet environment. Many of the differences affect the kinds of features that may exist, how these features are created, and how their interactions are managed.

Internet telephony signalling protocols are significantly more expressive than those of the PSTN, particularly when compared to the suite of signalling tones and hook signals available to two-wire analog telephones. Rich signalling in Internet telephony eliminates many legacy limitations on feature development. As an example, an end system need no longer indicate its desire to transfer a call using an elaborate sequence of switch hook and DTMF tones but can instead explicitly indicate to its partner the party to which the call should be transferred.

Another characteristic of Internet telephony signalling is that it can be extended while maintaining compatibility. When new signalling properties or events are created and added to the existing protocol, they interoperate cleanly with existing implementations, either by providing richer information about the signalling information or by allowing fine-grained control over the features required to understand a signalling message successfully. Internet telephony devices can also query each other to determine what properties and parameters they support. As new signalling elements and capabilities are developed, the network is able to evolve gracefully to support advanced features without the need for painful upgrades over an entire system [24].

One major difference between the Internet telephony protocols and those of the PSTN or ISDN is that the protocols employed by the user's device to communicate with the network (over the user-network interface, or UNI) and the protocols used by network devices to communicate with each other (over the network-network interface, or NNI) are identical. In fact, Internet telephony does not make a strong distinction between user devices and network devices. A device sending a request is typically not aware of whether it is communicating with a signalling server or an end system, nor does it need to be aware of this distinction. Because of this unification, Internet telephony deployments can readily scale up from a few individuals running end systems to a giant organization providing elaborate services and user location features - and both these organizations can co-exist and interoperate without problems. What's more, this means that a customer of a large provider can even choose to bypass the provider if his current needs don't require its services. For reasons of simplicity, flexibility, reliability or privacy, users can choose to communicate with each other directly end-to-end rather than through intermediate servers, without any incumbent need to modify their end systems.

<u>On the network side</u>, because IP is entirely packet-based, media communications are not limited to a single fixed rate channel as they are in the

circuited-switched network. Internet telephony can use very low bit rate for speech encoding and very high bandwidth for video. Multiple media sessions can also be used in a single call.

IP also supports network-level multicast protocols, which do not require application-level devices such as bridges. This enables a number of features both at the signalling and media level. More interestingly, media can also be multicast, which allows multi-party conferences to be established, in a bandwidth-efficient way, without the need for a conference bridge.

On the features side, however, the features of the Internet itself introduce a significant number of additional complications to the creation and deployment of features and to the resolution of their interactions. Most of these problems have occurred concurrently with the new possibilities enabled by the Internet. These issues also increase the complexity of creating features. The most significant of these new complications is the *distributed nature* of the Internet itself. Features can be implemented and deployed on numerous network devices, both end systems and signalling servers. What's more, these systems may well be controlled by entirely distinct organizations, which may be unaware of each other or may even competing with each other, and thus may not be inclined to co-operate to resolve feature interactions [25].

Another related complication is the fact that end systems have control of the call state. While this introduces many new possibilities for general feature creation and deployment, it also complicates things when the network wants to impose control in a manner opposed to the expressed desires of an end system. For example, in traditional telephone networks, emergency calls are usually subjected to special handling, such that the emergency operator must be the one to release the call before the line is cleared. If the end system controls its own states, however, it is impossible for the network to enforce such special treatment without the end system's cooperation.

2.4.3 Application Of Existing Feature Interaction Techniques

Several varieties of new feature interactions appear in Internet telephony, which are either not present or not as prominent in traditional telephone networks. Most feature interaction solutions used in the PSTN or in IN networks should also be applicable to the new IP network. The special character of completely distributed architecture in IP network, however, make it imperative to supplement this with new solutions to deal with feature interaction problems.

These new solutions can be categorized into two types of interactions. *Cooperative* interactions are based on the premise that all parties who implement features consider the other parties' actions to be reasonable, and would therefore prefer to avoid an interaction if at all possible. *Adversarial* interactions, by contrast, are those where the parties involved in the call have conflicting desires, and each is trying to subvert the other's features. Roughly speaking, cooperative interactions correspond with single-user multiple-component (SUMC) interactions whereas adversarial interactions are more commonly multiple-user multiple-component (MUMC) or customer-system (CUSY) interactions as discussed in the previous sections [26].

2.5 Chapter Summary

This chapter discusses the feature interaction issue in general. It covers the general feature interaction category based on the benchmark [11]; describes the feature interaction handling approaches, which include detection, resolution and management; lastly it addresses the handling difference between PSTN and IP network from three different perspectives, which are signalling protocol side, network side and feature side.

Chapter 3

SIP AND SIP SERVLET ARCHITECTURE

3.1 Introduction

With the new technology development and Internet is getting more ubiquitous, nowadays telephony can be provided through either traditionally switched network or Internet, which is so-called Internet telephony. The two employ different ways to establish, manage and terminate a call. In the second case, the same IP network can be used for voice, video and data. One of the most popular protocols in this area is SIP.

On top of SIP signaling for the basic call handling, SIP servlet [27] provide a new way of implementing different services. The SIP servlet is very much similar to HTTP servlet in terms of functions and implementation [28]. This chapter focuses on the protocols, including messages, parameters and their formats.

3.2 Session Initiation Protocol

The Session Initiation Protocol (SIP) signaling is an application-layer control (signaling) protocol for creating, modifying and terminating sessions with one or more participants. These sessions include Internet multimedia conferences, Internet telephone calls and multimedia distribution. Members in a session can communicate via multicast or via a mesh of unicast relations, or a combination of these [29].

SIP invitations used to create sessions carry session descriptions, which allow participants to agree on a set of compatible media types. SIP supports user mobility by proxying and redirecting requests to the user's current location. Users can register their current location. SIP is not tied to any particular conference control protocol. SIP is designed to be independent of the lower-layer transport protocol and can be extended with additional capabilities.

In SIP protocol participants are identified by SIP URLs. SIP is a request-response protocol, with requests sent by clients and received by servers. A single implementation typically combines both client and server functionality. SIP requests can be sent using any reliable or unreliable protocol, including UDP, SCTP and TCP. Protocol operation is largely independent of the lower-layer transport protocol.

The SIP specification [29] defines six SIP request methods:

- INVITE initiates sessions
- ACK confirms session establishment

- OPTIONS requests information about capabilities
- BYE terminates a sessions
- CANCEL cancels a pending session
- REGISTER allows a client to bind a permanent SIP URL to a temporary SIP URL reflecting the current network location.

SIP requests and responses consist of a request (or status) line, a number of header lines and a message body. SIP requests can be sent directly from a user agent client to a user agent server, or they can traverse one or more proxy servers along the way.

User agents send requests either directly to the address indicated in the SIP URI or to a designated proxy ("outbound proxy"), independent of the destination address. The current destination address is carried in the Request-URI. Each proxy can forward the request based on local policy and information contained in the SIP request. The proxy may rewrite the request URI. A proxy may also forwards the request to another designated proxy regardless of the request URI. For example, a departmental proxy could forward all authorized requests to a corporate-wide proxy, which then forwards it to the proxy operated by the Internet service provider, which finally routes the request based on the request URI.

3.2.1 SIP Session

A SIP session is considered an exchange of data between associations of participants. As defined, a callee can be invited several times, by different calls, to the same session. If SDP is used, a session is defined by the concatenation of the user name, session id, network type, address type and address elements in the origin field [29].

A SIP session is initiated with the “INVITE” request message. A successful SIP invitation consists of two request messages, “INVITE” followed by “ACK”. The “INVITE” request message asks the callee to join a particular conference or establish a two-party conversation. After the callee has agreed to participate in the call, the caller confirms that it has received that response by sending an “ACK” request message.

The “INVITE” request message typically contains a session description, for example written in SDP format that provides the called party with enough information to join the session. For a multicast session, the session description enumerates the media types and formats that are allowed to be distributed to that session. For a unicast session, the session description enumerates the media types and formats that the caller is willing to use and where it wishes the media data to be sent. In either case, if the callee wishes to accept the call, it responds to the invitation by returning a similar description listing the media it wishes to use. For a multicast session, the callee should only return a session description if it is unable to receive the media indicated in the caller's description or wants to receive data via unicast.

3.2.2 SIP Message Format And Parameter

SIP is a text-based protocol and uses the ISO 10646 character set in UTF-8 encoding [30]. A SIP message is either a request from a client to a server, or a response from a server to a client.

SIP message = Request or Response

Both Request and Response messages use the generic-message format of transferring entities (the body of the message). Both types of messages consist of a start-line, one or more header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the carriage-return line-feed (CRLF)) indicating the end of the header fields, and an optional message-body.

```
generic-message  =  start-line *message-header
                    [ message-body ]
start-line       =  Request-Line | Status-Line
message-header   =  ( general-header | request-header | response-header
                    | entity-header )
```

The SIP header format is shown below:

```
general-header  =  Accept
                   | Accept-Encoding
                   | Accept-Language
                   | Call-ID
                   | Call-Info
                   | Contact
```

```
                  | CSeq
                  | Date
                  | Encryption
                  | From
                  | MIME-Version
                  | Organization
                  | Record-Route
                  | Require
                  | Supported
                  | Timestamp
                  | To
                  | User-Agent
                  | Via
entity-header   = Allow
                  | Content-Disposition
                  | Content-Encoding
                  | Content-Language
                  | Content-Length
                  | Content-Type
                  | Expires
request-header  = Alert-Info
                  | Authorization
                  | In-Reply-To
```

```
                        | Max-Forwards
                        | Priority
                        | Proxy-Authorization
                        | Proxy-Require
                        | Route
                        | Response-Key
                        | Subject
response-header  =  Error-Info
                        | Proxy-Authenticate
                        | Retry-After
                        | Server
                        | Unsupported
                        | Warning
                        | WWW-Authenticate
```

3.2.3 Request Message

The request message format is shown below:

```
Request  = Request-Line *(general-header | request-header | entity-
header )
                  [ message-body ]
```

The Request-Line begins with a method token, followed by the Request-URI and the protocol version as shown below:

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

Request-URI = SIP-URL | absoluteURI

SIP-Version = "SIP/2.0"

The methods are defined below:

Method = "INVITE" | "ACK" | "OPTIONS" | "BYE"| "CANCEL" | "REGISTER" | extension-method

We will only discuss "INVITE" and "ACK" messages format and its related parameters in this thesis since it is closely related to the detection mechanism. All other SIP request messages are not relevant for the topic of this thesis.

The "INVITE" request message indicates that the user or service is being invited to participate in a session. The message body may contain a description of the session to which the callee is being invited.

The "ACK" request message confirms that the client has received a final response to an "INVITE" request message. ("ACK" is used only with "INVITE" requests.) 2xx responses are acknowledged by client user agents, all other final responses by the first proxy or client user agent to receive the response. The "Via" header field is always initialized to the host that originates the "ACK" request message, i.e., the client user agent after a 2xx response or the first proxy to receive a non-2xx final response. The "ACK" request message is forwarded as the corresponding "INVITE" request message, based on its Request-URI and thus may take a different path than the original "INVITE" request message, and may even cause a new transport connection to be opened in order to send it.

It should be highlighted that the session is not established and SDP connection is not available before "ACK" message is sent. This is one of the key issues for feature interaction detection, which will be discussed later on.

3.2.4 Response Message

After receiving and interpreting a SIP request message, the recipient responds with a SIP response message. The response message format is shown below:

*Response = Status-Line *(general-header | response-header | entity-header)*

[message-body]

The first line of a response message is the Status-Line, consisting of the protocol version followed by a numeric Status-Code and its associated textual phrase.

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

The Status-Code is a 3-digit integer result code that indicates the outcome of the attempt to understand and satisfy the request. The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata, whereas the Reason-Phrase is intended for the human user. The client is not required to examine or display the Reason-Phrase.

Status-Code = Informational | Success | Redirection | Client-Error | Server-Error

| Global-Failure | extension-code

extension-code = 3DIGIT

Reason-Phrase = TEXT-UTF8

The overview of the Status-Code is shown below. The first digit of the Status-Code defines the class of response. The last two digits do not have any categorization role. SIP/2.0 allows 6 values for the first digit:

1xx: Informational -- request received, continuing to process the request

2xx: Success -- the action was successfully received, understood, and accepted

3xx: Redirection -- further action needs to be taken in order to complete the request

4xx: Client Error -- the request contains bad syntax or cannot be fulfilled at this server

5xx: Server Error -- the server failed to fulfill an apparently valid request

6xx: Global Failure -- the request cannot be fulfilled at any server.

The Informational and success status codes is shown below:

```
Informational = "100" ; Trying
              | "180" ; Ringing
              | "181" ; Call Is Being Forwarded
              | "182" ; Queued
              | "183" ; Session Progress
Success       = "200" ; OK
```

We will only focus on the 1xx and 2xx response messages in this thesis. The main reason for that is the feature interaction wouldn't be an issue if the call or feature activation were failed.

3.2.5 Header Field

SIP header fields are similar to HTTP header fields in both syntax and semantics. In particular, SIP header fields follow the syntax for message-header. The header fields required, optional and not applicable for each method are listed in table below. The table uses "o" to indicate optional, "m" mandatory and "-" for not applicable.

"Optional" means that a UA may include the header field in a request or response message, and UA may ignore the header field if present in the request or response message. A "mandatory" request header field must be present in a request message, and must be understood by the UAS receiving the request message. A mandatory response header field must be present in the response message, and the header field must be understood by the UAC processing the response message. "Not applicable" means for request header fields that the header field must not be present in a request message. If one is placed in a request by mistake, it MUST be ignored by the UAS receiving the request. Similarly, a header field labeled "not applicable" for a response message means that the UAS must not places the header in the response message, and the UAC must ignore the header in the response message.

Header field	where	ACK	BYE	CAN	INV	OPT	REG
Accept	R		o	o	o	o	o
Accept	415		o	o	o	o	o
Accept	r						
Accept-Encoding	R	-	o	o	o	o	o
Accept-Encoding	415	-	o	o	o	o	o
Accept-Language	R	-	o	o	o	o	o
Accept-Language	R	-	o	o	o	o	o
Alert-Info	R	-	-	-	o	-	-
Allow	R	o	o	o	o	o	o
Allow	200	-	-	-	o	o	o
Allow	405	m	m	m	m	m	m
Also	R	-	o	-	-	-	-
Authorization	R	o	o	o	o	o	o
Authorization	R	o	o	o	o	o	o
Call-ID	gc	m	m	m	m	m	m
Call-Info	g	-	-	-	o	o	o
Contact	R	o	-	-	m	o	o
Contact	1xx	-	-	-	o	o	-
Contact	2xx	-	-	-	m	o	o
Contact	3xx	-	o	-	o	o	o
Contact	485	-	o	-	o	o	o
Content-Disposition	e	o	o	-	o	o	o

Content-Encoding	e	o	o	-	o	o	o
Content-Language	e	o	o	o	o	o	o
Content-Length	e	m	m	m	m	m	m
Content-Type	e	*	*	-	*	*	*
Cseq	gc	m	m	m	m	m	m
Date	gc	o	o	o	o	o	o
Encryption	g	o	o	o	o	o	o
Error-Info	g	o	o	o	o	o	o
Expires	gc	-	-	-	o	-	o
From	R	m	m	m	m	m	m
In-Reply-To	R	-	-	-	o	-	-
Max-Forwards	R	o	o	o	o	o	o
MIME-Version	g	o	o	o	o	o	o
Organization	g	-	-	-	o	o	o
Priority	R	-	-	-	o	-	-
Proxy-Authenticate	401,407	o	o	o	o	o	o
Proxy-Authorization	R	o	o	o	o	o	o
Proxy-Require	R	o	o	o	o	o	o
Record-Route	R	o	o	o	o	o	o
Record-Route	2xx,401,484	o	o	o	o	o	o
Require	g	o	o	o	o	o	o
Response-Key	R	-	o	o	o	o	o
Retry-After	R	-	-	-	-	-	o

Retry-After	404,413,480 ,486 500,503 600,603	o	o	o	o	o	o
Route	R	o	o	o	o	o	o
Server	r	o	o	o	o	o	o
Subject	R	-	-	-	o	-	-
Supported	g	-	o	o	o	o	o
Timestamp	g	o	o	o	o	o	o
To	gc(1)	m	m	m	m	m	
Unsupported	R	o	o	o	o	o	o
Unsupported	420	o	o	o	o	o	o
User-Agent	g	o	o	o	o	o	o
Via	gc(1)	m	m	m	m	m	m
Warning	r	o	o	o	o	o	o
WWW-Authenticate	R	o	o	o	o	o	o
WWW-Authenticate	401	o	o	o	o	o	o

Table 1 SIP Message Header Field

We will now use "Contact" header field as an example, to analyze the information provided by this field.

Contact:

The “Contact” general-header field can appear in “INVITE”, “OPTIONS”, “ACK”, and “REGISTER” request messages, and in 1xx, 2xx, 3xx, and 485 response messages. In general, it provides a URL where the user can be reached for further communications.

In some of the cases below, the client uses information from the “Contact” header field in Request-URI of future request messages. In these cases, the client copies all but the "method-param" and "header" elements of the addr-spec part of the “Contact” header field into the Request-URI of the request message. It uses the "header" parameter to create headers for the request messages, replacing any default headers normally used. Unless the client is configured to use a default proxy for all outgoing request messages, it then directs the request to the address and port specified by the "maddr" and "port" parameters, using the transport protocol given in the "transport" parameter. If "maddr" is a multicast address, the value of "ttl" is used as the time-to-live value.

For “INVITE”, “OPTIONS” and “ACK” request messages: “INVITE” request message must and “ACK” request message may contain “Contact” headers indicating from which location the request is originating. The URL in the “Contact” header field is then used by subsequent requests from the callee.

This allows the callee to send future requests, such as “BYE” message, directly to the caller instead of through a series of proxies. The “Via” header is not sufficient since the desired address may be that of a proxy.

"INVITE" 1xx response messages: A UAS sending a provisional response (1xx) may insert a Contact response header. It has the same semantics in a 1xx response message as a 2xx response message.

"INVITE" and "OPTIONS" 2xx response messages: A user agent server sending a definitive, positive response (2xx) MUST insert a "Contact" response header field indicating the SIP address under which it is reachable most directly for future SIP requests, such as "ACK" message, within the same Call-ID. The "Contact" header field contains the address of the server itself or that of a proxy, e.g., if the host is behind a firewall. The value of this "Contact" header is copied into the Request-URI of subsequent requests for this call if the response did not also contain a Record-Route header. If the response message also contains a Record-Route header field, the address in the Contact header field is added as the last item in the Route header field.

3.3 SIP Servlets

One of the requirements for Internet telephony is to support at least the same applications/service/feature as traditional telephony.

The main reason that the World Wide Web is so successful is that it is (relatively) easily programmable. This means that Web server can host a variety of applications, which can be updated and managed independently of Web browsers. Client software needs only to know how to access services using HTTP protocol and how to render HTML and doesn't usually participate directly in executing service logic [31].

In the traditional switched network, services such as personal mobility, Call Forwarding, Call Screening, etc., are also introduced via programmability of network servers. For a number of reasons, this has traditionally been done in a closed, proprietary manner, which translates directly into slower, more expensive, less creative, but also more reliable and secure services. The service is mostly provided together with the call control machine.

By using SIP servlets to execute services, a number of advantages can be identified. Some of these advantages are directly derived from the fact that SIP servlets are Java programs. Just to name a few [28]:

- SIP servlets are platform independent, as the actual code is hidden by the Java Virtual Machine.
- Memory access violations and strong typing violations are not possible, so that faulty servlets will not crash the.
- As SIP servlets are platform independent, the service production cost is dramatically reduced.

- SIP servlets provide strong security policy support, as all the Java environments provide a Security Manager, which can be used to control whether actions such as network or file access are to be permitted.

What is not addressed so far with this approach is the issue of "feature interaction". Feature interaction is a general problem even in the switched network, when more than one feature is activated together; there is always a possibility that the interaction will be occurring. This cannot be avoided in the new IP telephony environment. This issue while in switched network can mostly be resolved by proprietary solution since call control and service control used to be integrated together. However, in a distributed environment such as SIP, new mechanism is required to detect, manage and resolve the feature interaction. This thesis focuses on the solution of detecting and resolving feature interaction in the context of SIP servlet environment.

3.3.1 SIP Servlet Overview

The architecture used in this thesis is based on the SIP Servlet API [32]. There are other solutions available, such as CPL [33] based. In this API, a servlet extension to SIP is proposed. However, the mechanism of how to handle the feature interaction is not addressed. Given the momentum of SIP in the industry, the feature interaction will be an issue in the foreseeable future. This thesis explored one of the possibilities. There are many other approaches, such as "Network-level administrative restriction or Universal authentication" [22].

The SIP servlet extension in many ways is very similar to the Http servlet. Generally speaking, the service servlet in HTTP context can only be initiated by the incoming "request" message. In the traditional http servlet environment, the service environment consists of feature/service servlet and http server. It is sufficient to run the service without any major problem since the user client can only send "request" messages, but not receiving "request" messages. The user client in this case is normally the web browser. The situation is a little bit different in the SIP environment. The User Agent (UA) can not only send but also receive request messages. It is a both way communication. By SIP definition, a UA includes User Agent Client (UAC) and User Agent Server (UAS). While sending "request" messages, UA assumes the role of UAC. While receiving "request" messages, it assumes the role of UAS. This imposed new challenge on the servlet in SIP.

3.3.2 SIP Servlet API Architecture

The Figure 3 below represents the high level SIP servlet API architecture.

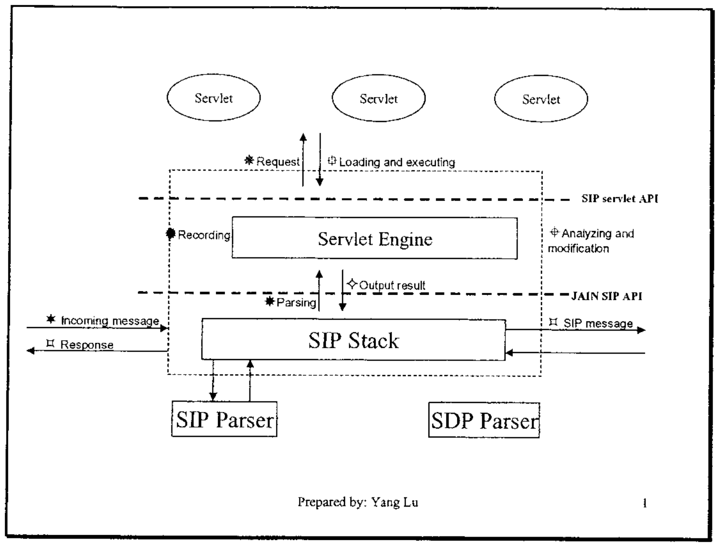


Figure 3 SIP Servlet API Architecture

The SIP servlet architecture consists the following components:

• SIP stack:	sending/receving SIP request or response messages
• Servlet Engine:	loading and executing individual feature servlet
• Service Servlet:	each servlet is an individual service

When the SIP stack receives a SIP request message, it will interpreter the message and if it contains the invocation of a service servlet, the stack will past the control to the servlet engine. The servlet engine will then look up the relevent service servlet, load it into the servlet engine and get it executed.

3.3.3 Servlet Engine

The servlet engine is the software used in conjunction with a SIP server (i.e. Redirect Server and Proxy Server) or UA (i.e. UAS and UAC) that provides support for an execution of servlets as defined by servlet API. To run a servlet, the servlet engine must instruct the server what to run when a servlet is encountered (much like CGI). Once the server passes control to the servlet engine, the servlet engine is responsible for executing the servlet.

The servlet engine is just the software specifically made to control the execution of individual SIP servlet. When the SIP server receives the incoming SIP message, the message will be passed over to servlet engine. The servlet engine will first of all interpret the message with the SIP parser stack help and then execute them in the servlet handler.

3.3.4 User Agent

The user agent by SIP definition should contain UAC (user agent client) and UAS (user agent server). The user agent server (UAS) receives SIP request messages from a client (UAC). It processes the header fields of the request message that describe the call, and the body of the request message, which describes the individual media sessions that, makes up the call. Depending on the header information and on the state of the server, decision to invoke or not a service can be taken by the server.

3.4 Chapter Summary

This chapter describes the general SIP and SIP servlet architecture and its components. For SIP, the detailed message and parameter format and some basic principles are specified. For SIP servlet, all the involved components are individually described.

It provides a foundation for the in-depth feature interaction detection analysis in later sections.

Chapter 4

FEATURE INTERACTION DETECTION IN A SIP SERVLET ENVIRONMENT

4.1 Introduction

The feature interaction handling for traditional telephony service in a SIP servlet environment will be different. In order to really understand the issue, we need to understand in details how SIP protocol and SIP servlets interact and how the feature interaction can be detected and managed. Here we will first examine some typical telephony services and their interactions by analyzing the signals and call flows implemented with SIP and SIP servlets.

A logical entity "Feature Interaction Handler (FIH)" is then proposed to specifically handle the feature interaction detection in the SIP servlet environment. The FIH usage and its interface with other components, such as servlet engine are described.

Our feature interaction detection approach covers both offline and online types. The distinction between these two types is the following.

- The offline feature interaction detection technique applies when the feature are assigned to the user by service provider.

- The online feature interaction detection technique applies when the feature is executed during a "call" session.

4.2 General Service Architecture

In this section, we discuss the service environment this thesis is based on. In the real network, there are many different configurations and setups. It is impossible to have only one generic network architecture. For instance, within a service provider domain, there may be a centralized "feature/service" server. There should be a "PROXY" server in the edge of the service provider domain to interface with other service providers. In this case, since all user profile is controlled centrally, it is relatively easy to detect and manage the feature interaction problem.

However, it is more often that two users involved in a call scenario belong to two different service provider domains. Therefore, the user profiles are distributed and feature interaction detection has to rely on the messages passed between service provider domains. For the purpose of a general feature interaction detection technique, we assumed two independent "service provider" domains in this thesis. Each domain maintains its own user data.

Figure 4 below shows the general service architecture.

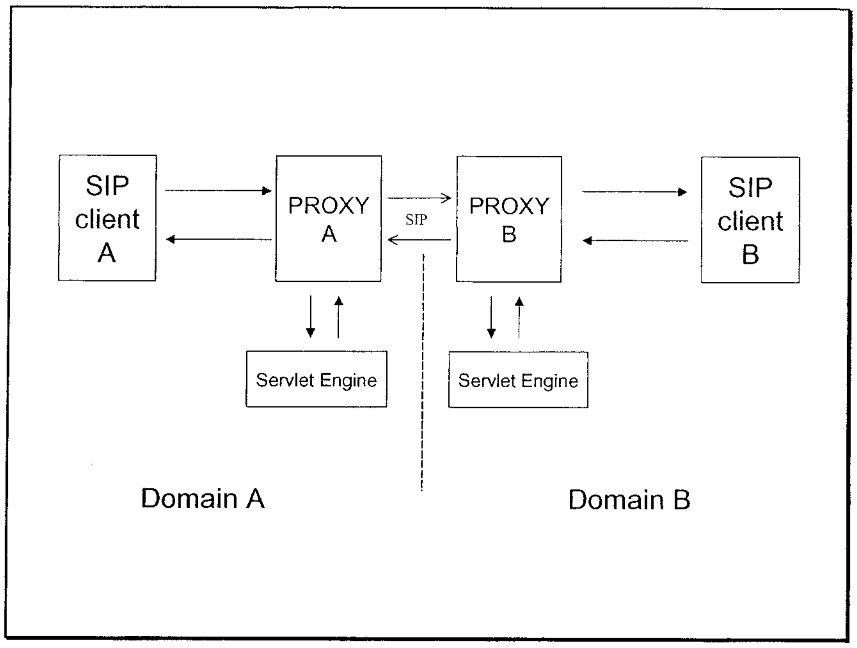


Figure 4 Service Architecture Without FIH

The problem with the architecture as shown in Figure 4 is that it does not address the feature interaction problem. We propose to add another logical entity, called “Feature Interaction Handler (FIH)”. The main function of the FIH is to detect and resolve feature interaction when it occurs. FIH is a logical entity and it interfaces with the servlet engine.

With the addition of the FIH, the service architecture is modified as shown in Figure 5.

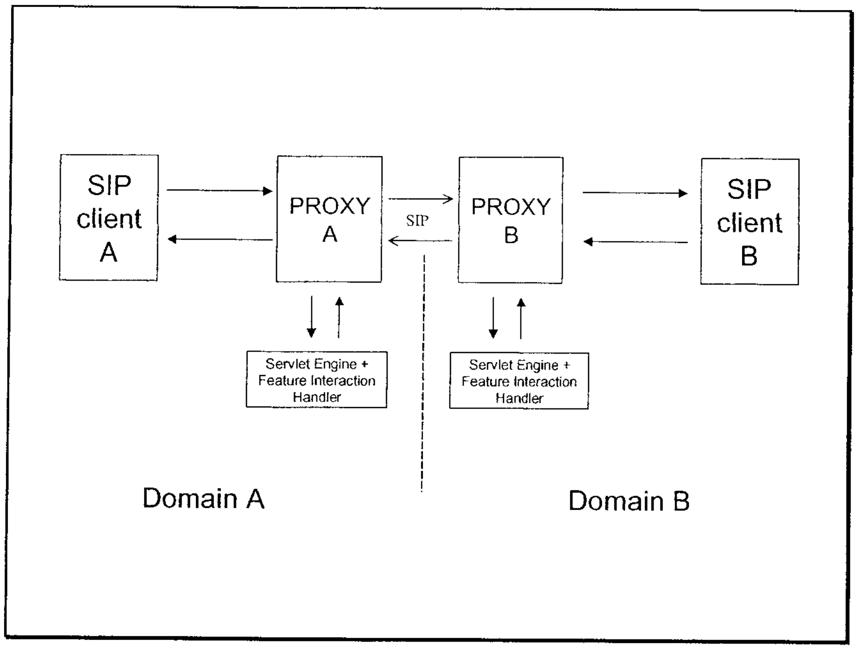

SIP client A
PROXY A
SIP
PROXY B
SIP client B
Servlet Engine + Feature Interaction Handler
Servlet Engine + Feature Interaction Handler
Domain A
Domain B

Figure 5 Service Architecture With FIH

In this architecture, there are two "service provider" domains, Domain A and Domain B. Each domain has its own PROXY and the interface between the two "service providers" is through PROXIES using SIP protocol. There are also two users, User A (UA A) and User B (UA B). The assumptions of this thesis are listed below:

- Each user's profile is only available in its own domain and managed by domain's PROXY. This includes the feature registration/activation, account management, billing, etc.
- The feature implementation in each PROXY is based on SIP servlet concept. The feature execution is initiated by servlet engine and each feature has its own servlet.
- The servlet engine is co-located with the PROXY and is available in every PROXY. A servlet engine is a logical entity (functional). To simplify the

implementation and focus on the feature interaction detection issue, we decide to co-locate it with PROXY.

- The FIH introduced in this thesis is also co-located with the servlet engine and is available in every PROXY. FIH is also a logical entity.
- The signaling between PROXYs is done using SIP protocol.

Our feature interaction detection approach consists of two techniques, offline and online.

4.3 Offline Detection Technique

Offline detection technique discussed in this thesis is applicable for one user only.

This type of feature interaction can be detected immediately while user gets registered to certain feature combination. For instance, as soon as subscriber register to feature "Call Waiting" and Call Forwarding" at the same time, we know this will definitely cause feature interaction. The feature interaction in this case can be managed easily with a pre-defined knowledge.

Since the behavior of each individual feature is known in the design phase, so by analyzing the behavior, we could determine if two features would have interactions if invoked at the same time.

The common approach of handling offline feature interaction is pairwise analysis. Before introducing a new feature into a network, the feature has to be analyzed with respect to every feature the user has subscribed to.

This thesis proposes an approach for feature interaction detection by categorising the "behavior" of each feature, and then grouping them accordingly. Let us assume the total number of features is $\underline{M}$. We group the features with similar behavior together. Each feature group is a feature category and the total number of feature categories is $\underline{N}$. Then this will create an $\underline{M} \rightarrow \underline{N}$ translation table, in which $\underline{N}$ should be much less than $\underline{M}$. By doing this, the feature interaction detection is on the category basis rather than on feature basis. This should substantially reduce the complexity on the feature interaction detection.

There are three steps in this approach as listed below:

A) Mapping: The first step is to map every feature to a pre-defined behavior category. Each category is defined according to certain behavior.

B) Analysis: The second step is to analyze the interaction between feature categories. Based on the analysis result, we create a feature category interaction matrix.

C) Resolution: The third step is to resolve the feature interactions. The solution proposed in this thesis is to inhibit the combination of features. Strictly speaking, this is not a proper resolution method in feature interaction management.

4.3.1 Mapping

The feature category is based on the actual actions of the feature. For instance, forwarding category covers all type of call forwarding features. Even though the forwarding features have various triggering conditions, the effect is the same. If the forwarding happens, the call will be terminated in a third party. Based on this principle, we have defined six categories, as follows:

- *Forwarding*: This is applicable to all type of call forwarding features, like forward on busy, forward on no reply. The trigger used to identify this behavior is the modification of "URL" or “Via” header in the original "INVITE" message.
- *Authentication:* This is applicable to all type of features that requires authentication before the session is established. The trigger used to identify this behavior is the parameter "Auth". Features in this category are terminating call screening, outgoing call screening, etc.
- *Delay*: This is applicable to features that require "delayed" session establishment. Unlike the regular terminating call scenario, the “delay” category involves “pre-condition”. It means that “pre-condition” has to be met before the call can be terminated. Features in this category are call waiting, camp on busy, or call completion to busy subscriber, etc.

- *Multi-party:* This is applicable to features that involve more than two users where session is established. Features in this category are three way calling, etc.
- *Regulation:* This is applicable to features that require special treatment, such as 911. This has to be treated separately due to the regulation requirement. Even in the normal conditions, it behaves differently from the normal sessions.
- *Display:* This is applicable to features related to either calling number display or called number display. It also applies to the screening service based on the numbers. Features in this category are calling number presentation, etc.

There is no limitation in the combination of user's features. The same principle applies to the behavior combination of above categories. In this case, a mechanism needs to be created to simulate all possible combinations.

4.3.2 Analysis

We will not analyze all different combinations of categories as introduced in the previous section. Only a few combinations are selected to explain the concepts and the result will also be used in the examples shown in later sections.

4.3.2.1 Forwarding and Delay

Between feature category "Forwarding" and category "Delay", a feature interaction is inevitable if both categories are applied to one user. Regardless the type of forwarding (forwarding on busy, forwarding unconditional, etc), it means that the call will terminate in user C instead of dialed user B. "Delay" category means "pre-condition" has to be met before the call can be terminated, but the call will be terminated in user B. With the call forwarding effect, the behavior of "forwarding" feature category violates the assumption of "Delay" feature category. We therefore can conclude that "forwarding" feature category and "Delay" feature category will cause feature interaction.

{Forwarding <> Delay} => FI-AA

4.3.2.2 Delay and Multi-party

Between category "Delay" and category "Multi-Party", the feature interaction can happen under certain conditions. Features in "Delay" category has "pre-condition" to meet before call termination could precede. For instance, "call waiting" feature requires user to put existing call leg on hold before he can accept the new incoming leg. Same as features in "multi-party" category, the same action "flash" is required in both cases in order to put existing leg on hold. However, the result is different in two cases. One is to accept the new incoming leg and the other is to connect a third party. Feature interaction will occur in this case. However, we cannot conclude that "delay" category will always cause feature interaction with "Multi-party" category. There will be no feature interaction if we put "camp on busy" (part of Delay category) together with "three-way call" (part of Multi-party category).

{Delay <> Multi-Party} => or =/>FI-AA

4.3.2.3 Feature Category Interactions Matrix

The Table 2 below shows the feature interaction summary between the different categories defined in the offline detection method. The summary serves as a general detection matrix. The feature categories shown in the table represent features assigned to one user.

In the table:

- FI means feature interaction will occur.
- "<>" means no feature interaction.

	Forwarding	Authentication	Delay	Multi-Party	Regulation	Display
Forwarding	<>	<>	FI	<>	<>	<>
Authentication	<>	-	<>	<>	<>	<>
Delay	FI	<>	-	FI	<>	<>
Multi-party	<>	<>	FI	-	<>	<>
Regulation	<>	<>	<>	<>	-	<>
Display	<>	<>	<>	<>	<>	<>

Table 2 Feature Category Interaction Matrix

4.3.3 Resolution

The common resolution method is to inhibit the combination of conflicting features. This is also the method used in this thesis for offline feature interaction handling. It works well for the FI-AA type of feature interaction defined in this thesis. However, this method will not work in distributed service environment architecture as shown in Figure 5. It doesn't address the FI-AB type of feature interaction. There is no way to coordinate the user profile between two different domains. The approach proposed in this thesis is to drop the call whenever feature interaction is detected. It simplifies the implementation and focus only on the feature interaction detection issue.

4.4 Applying Feature Category Interactions Matrix To Two Users

As discussed earlier, the offline detection is for one user only. However, the concepts for detecting interaction between feature categories are also applicable to multiple users. Therefore, in this section, we analyze the feature categories interaction between two users by showing one example. The analysis result for two users will be used later on for online detection analysis.

To avoid confusion, we also introduce our notation for the type of feature interaction. "**FI-AA**" represents feature interaction between two features of one user.

"**FI-AB**" represents feature interaction between features belonging to two users. "**AB**" also represents the direction of the call.

4.4.1 Forwarding And Authentication

Between category "Forwarding" and category "Authentication", an interaction may occur if these two features are applied to two different users. Regardless of the type of forwarding (forwarding on busy, forwarding unconditional, etc), the call will terminate in user C instead of dialed user B. "Authentication" category means either user B for an outgoing call or user A for an incoming call need to be checked before a session is established. With the call forwarding effect, the behavior "forwarding" feature category violates the assumption of "Authentication" feature category. The feature interaction condition is that user A has "authentication" feature category and user B has "forwarding" feature category. Therefore we can conclude the following:

{Authentication <> Forwarding} => FI-AB

It should be noted that there will be no feature interaction if user A has "forwarding" feature category and user B has "Authentication" feature category. There is also no feature interaction if both above features are applied to one user.

4.4.2 Modified Feature Category Interactions Matrix

The feature category interaction matrix is modified with the inclusion of "two users" capability and it is shown in Table 3. The feature categories shown in the table represent features assigned to user A and User B respectively.

In the table:

- FI-AA means two features are applied to one user. This is applicable to offline detection.
- FI-AB means two features are applied to two different users. This is applicable to online detection.
- "<>" means no feature interaction.

	Forwarding	Authentication	Delay	Multi-Party	Regulation	Display
Forwarding	FI-AB	FI-AB	FI-AA	<>	Conditional FI-AB	<>
Authentication	<>	-	<>	<>	<>	<>
Delay	FI-AA	<>	-	FI-AA	<>	<>
Multi-party	<>	<>	FI-AA	-	<>	<>
Regulation	<>	<>	<>	<>	-	<>
Display	<>	<>	<>	<>	<>	FI-AB

Table 3 Modified Feature Category Interactions Matrix

4.5 Online Detection Technique

The initial SIP servlet concept as proposed in the IETF Internet draft [32] did not address the feature interaction issue. The specified API structure in the draft needs to be enhanced in order to support feature interaction detection. In this section, we propose an "online feature interaction detection" mechanism to address this requirement. It is used for **FI-AB** type of feature interaction detection defined in the previous section.

Two methods are introduced. One is called "forward detection" and the other is called "backward detection".

- The "forward detection" is to detect feature interaction on the terminating side of the call as soon as SIP "INVITE" message is received;
- The "backward detection" is to detect feature interaction on the originating side of the call when SIP response message is received;

In order to handle feature interaction, the current SIP servlet architecture needs to be enhanced to incorporate the feature interaction detection and management function. As stated in the previous chapter, SIP servlet is very similar to HTTP servlet. It is however not enough just to apply the current HTTP servlet mechanism to the SIP servlet environment, especially when the feature interaction is brought into the picture.

We need to introduce a new logical entity called Feature Interaction Handler (FIH) as an enhancement to the general SIP servlet architecture. The main function of this entity is to detect and resolve the feature interaction.

4.5.1 Enhanced SIP Servlet Architecture

As mentioned earlier, we have introduced a new logical entity FIH into the SIP servlet architecture. The enhanced SIP servlet architecture is shown in Figure 6.

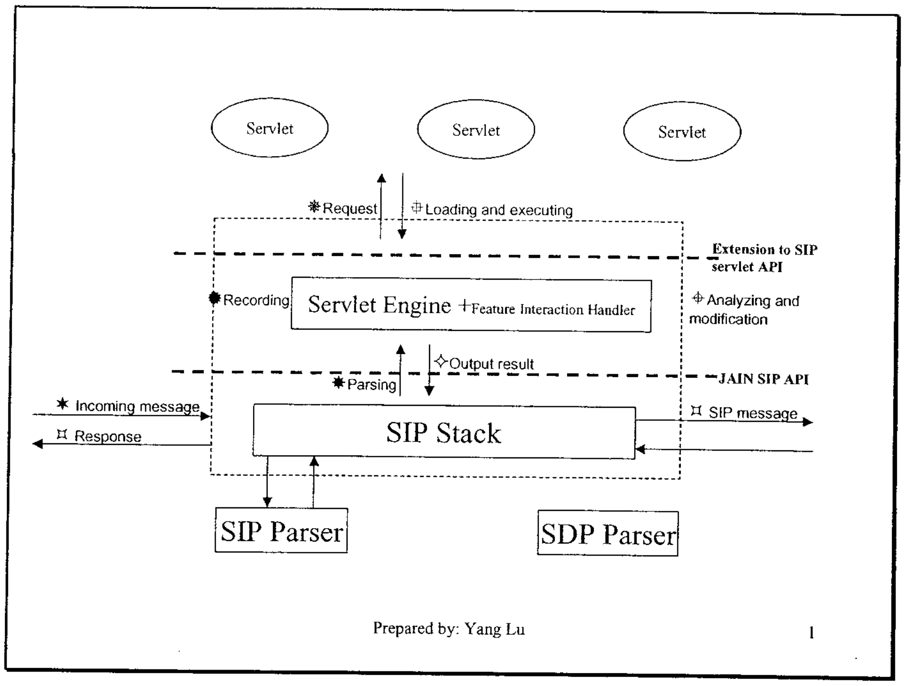


Figure 6 Enhanced SIP Servlet Architecture

As we can see from Figure 6, the FIH is an integral part of the servlet engine. For every incoming SIP message, it will go through the servlet engine if the feature servlet needs to be invoked. Since the FIH sits between the SIP stack and the servlet engine, the received SIP message will go through the FIH first, and then get forwarded to the servlet engine after checking for feature interactions.

Depending on the combination of features involved in a particular call scenario, the FIH may or may not take action. If there is no feature interaction detected by the FIH, the FIH will pass all the received information transparently to servlet engine. In this case, the FIH will take no action on the call, and servlet engine will execute feature as defined. If feature interaction is detected in the FIH, the FIH will construct a SIP release message to disconnect the call.

The FIH can check both SIP request messages and SIP response messages. It can also correlate between SIP request messages and SIP response messages for the same session.

In this thesis, the JAIN SIP API is used between FIH and SIP parser; SIP servlet API is used between the servlet engine and the individual servlets.

4.5.2 Backward Detection

Backward detection, as its name implies, is dependent on the SIP response message (received in backward direction). The detection point is at the call origination side of the call session. The FIH in domain A (as shown in Figure 5) is responsible for the feature interaction detection. The FIH needs to correlate the SIP request message (such as "INVITE") with the SIP response message (such as "2xx") in order to detect feature interaction.

4.5.2.1 Overview of Backward Detection Technique

According to RFC 3261 [29], a SIP session will not be established until "200 OK" response message is returned to the originating side. The SIP response message will provide additional information related to the terminating side. This information can be used by the originating side to determine if an ongoing session is allowed to proceed or not based on feature interaction analysis.

The detection procedure is as follows:

- Store and extract the received SIP response message and its headers. Only 1xx, 2xx and 4xx response message header are covered in this thesis. All other response messages can be potentially introduced.
- Correlate and compare the session information between initial request message "INVITE" with the received response message and its header. This is to detect feature effect on terminating side. For instance, if user B has call forwarding service, the returned response "200 OK" message "contact info" header will be changed. By comparing the request message and response message, the forwarding can be detected.
- Look up originating user A's profile to get feature list.
- Both originating user's feature list and SIP response message are crosschecked in the feature interaction category matrix as defined in the previous section. This is to determine if feature interaction will occur or not.
- If feature interaction is detected, the FIH will not forward any message to servlet engine. FIH will construct a SIP "BYE" message to abort the ongoing session.

4.5.2.2 Algorithm for Backward Detection Technique

The backward detection algorithm is shown in Figure 7. It has four key components as listed below:

- Response message header information extraction
- Response message header analysis
- Originating user profile reading
- Feature interaction category matrix table lookup

Each component in the backward detection algorithm is for one specific task.

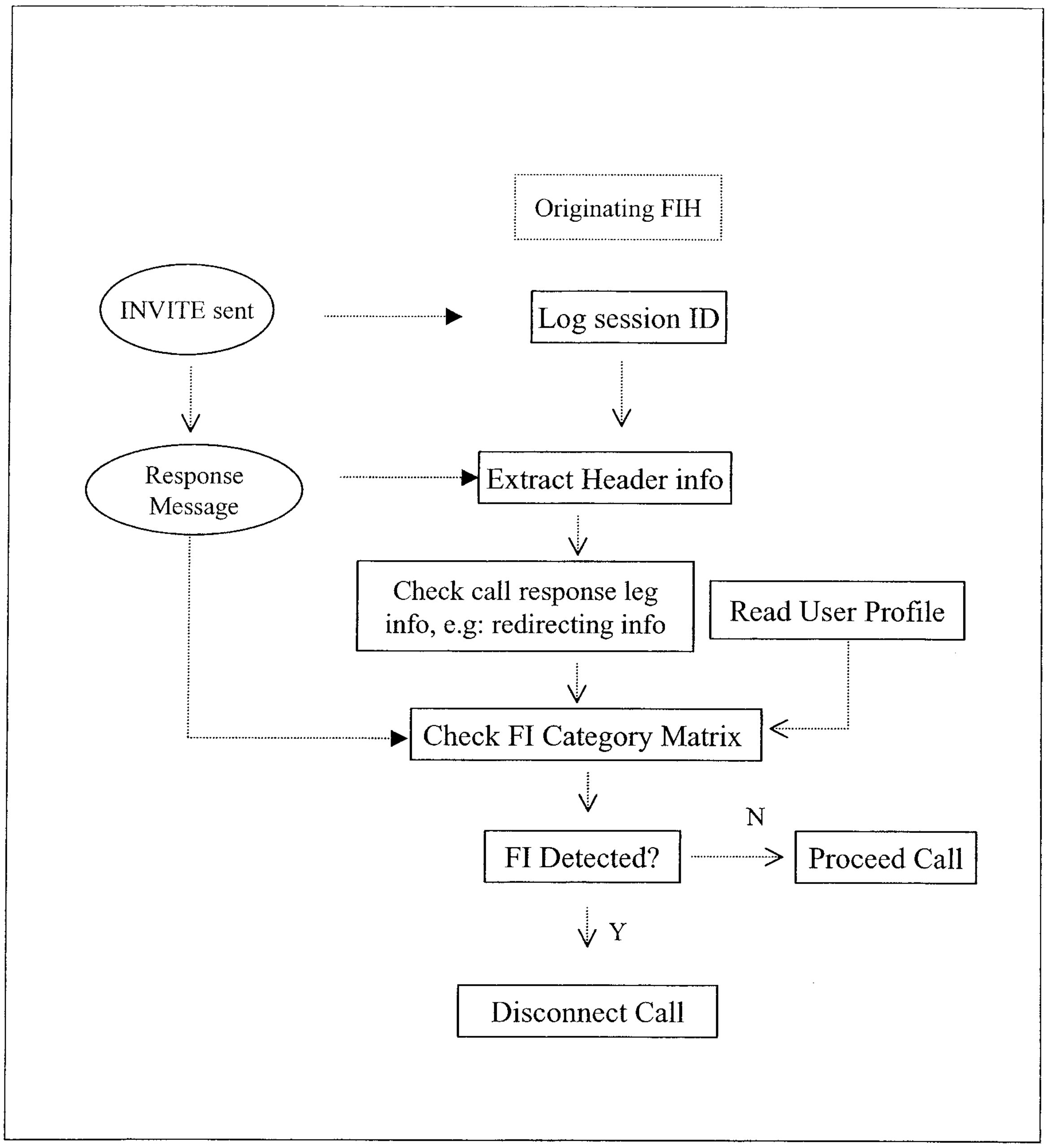


Figure 7 Backward Detection Algorithm

4.5.2.3 Example of Authentication vs. Forwarding

In general, the FIH on the originating side will not be aware of what has happened in the terminating side unless the received SIP response message can provide additional information [34].

If terminating user B has feature “call forwarding” [Category - Forwarding] and originating user A has feature "outgoing call screening” [Category - Authentication], feature interaction will occur on originating side. The question is how to detect if a call has been forwarded or not during the call setup stage. In order to get the additional information, we have to look at the SIP 1xx and 2xx response messages and their contents.

The "contact" header field in the SIP 1xx and 2xx response messages would be the place to look for the "forwarding" information. By comparing the "contact" header field in the request "INVITE" message and 1xx or 2xx response messages, we will find if the call has been forwarded or not.

Let us look at the scenario shown in Figure 8. This is an unconditional call forwarding signaling flow chart [35].

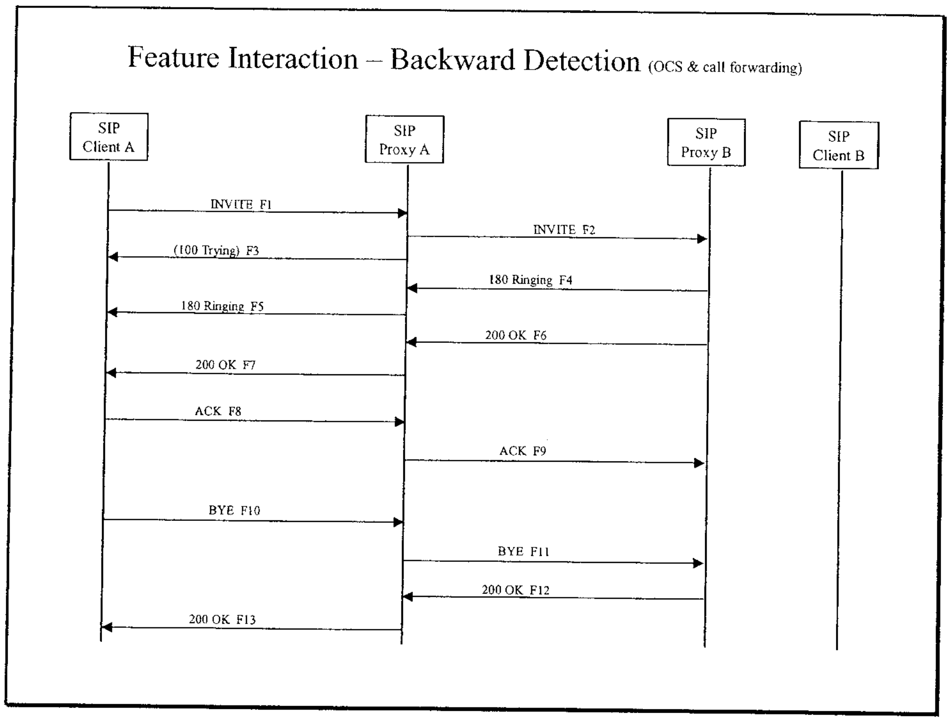


Figure 8 SIP Call Forwarding Signal Flow

User B wants all calls forwarded to the PSTN. User A calls User B. Since we only care about the feature interaction detection, we will only look at message step 1 & 6 & 7. The messages behind PROXY B are not shown [35].

For step 1, the message detail is as below:

INVITE sip:UserB@there.com SIP/2.0

Via: SIP/2.0/UDP here.com:5060

From: BigGuy <sip:UserA@here.com>

To: LittleGuy <sip:UserB@there.com>

Call-ID: 12345600@here.com

CSeq: 1 INVITE

Contact: BigGuy <sip:UserA@here.com>

```
Content-Type: application/sdp

Content-Length: ...

v=0

o=UserA 2890844526 2890844526 IN IP4 user.here.com

s=Session SDP

c=IN IP4 100.101.102.103

t=3034423619 0

m=audio 49170 RTP/AVP 0

a=rtpmap:0 PCMU/8000
```

For step 6 & 7, the message detail is as below:

F6:

```
200 OK Gateway -> Proxy

SIP/2.0 200 OK

Via: SIP/2.0/UDP ss1.wcom.com:5060;branch=83749.1

Via: SIP/2.0/UDP here.com:5060

Record-Route: <sip:UserB@there.com;maddr=ss1.wcom.com>

From: BigGuy <sip:UserA@here.com>

To: LittleGuy <sip:UserB@there.com>;tag=314159

Call-ID: 12345600@here.com

CSeq: 1 INVITE

Contact: <sip:+19727293660@gw1.wcom.com;user=phone>

Content-Type: application/sdp
```

Content-Length: ...

v=0

o=GATEWAY1 2890844527 2890844527 IN IP4 gatewayone.wcom.com

s=Session SDP

c=IN IP4 gatewayone.wcom.com

t=0 0

m=audio 3456 RTP/AVP 0

a=rtpmap:0 PCMU/8000

F7:

200 OK Proxy -> A

SIP/2.0 200 OK

Via: SIP/2.0/UDP here.com:5060

Record-Route: <sip:UserB@there.com;maddr=ss1.wcom.com>

From: BigGuy <sip:UserA@here.com>

To: LittleGuy <sip:UserB@there.com>;tag=314159

Call-ID: 12345600@here.com

CSeq: 1 INVITE

Contact: LittleGuy <sip:7773660,phone-

context=p1234@gw1.wcom.com;user=phone>

Content-Type: application/sdp

Content-Length: ...

v=0

o=GATEWAY1 2890844527 2890844527 IN IP4 gatewayone.wcom.com

```
s=Session SDP
c=IN IP4 gatewayone.wcom.com
t=0 0
m=audio 3456 RTP/AVP 0
a=rtpmap:0 PCMU/8000
```

For this particular feature interaction detection, by examining the "contact" header field in both “INVITE” message and “200 OK” response message of the same session, we detect if the interaction can occur or not. If the “contact” header remains unchanged for the whole session, we can conclude that the call is not forwarded during the setup stage and there will be no potential feature interaction. This correlation and detection can be well handled by FIH.

This approach also applies to any other forwarding services, including CFU, CFON and CFOB.

4.5.2.4 Example of Delay vs. Delay

Feature “automatic recall (ARC)” [Category – Delay]] automatically repeats last outgoing call when destination line is no longer busy. Feature “automatic call back (ACB)” [Category – Delay] automatically returns last incoming call when user line is no longer busy.

The signalling flow of ARC and ACB is shown in Figure 9. In this example, user A has feature ARC and user B has ACB.

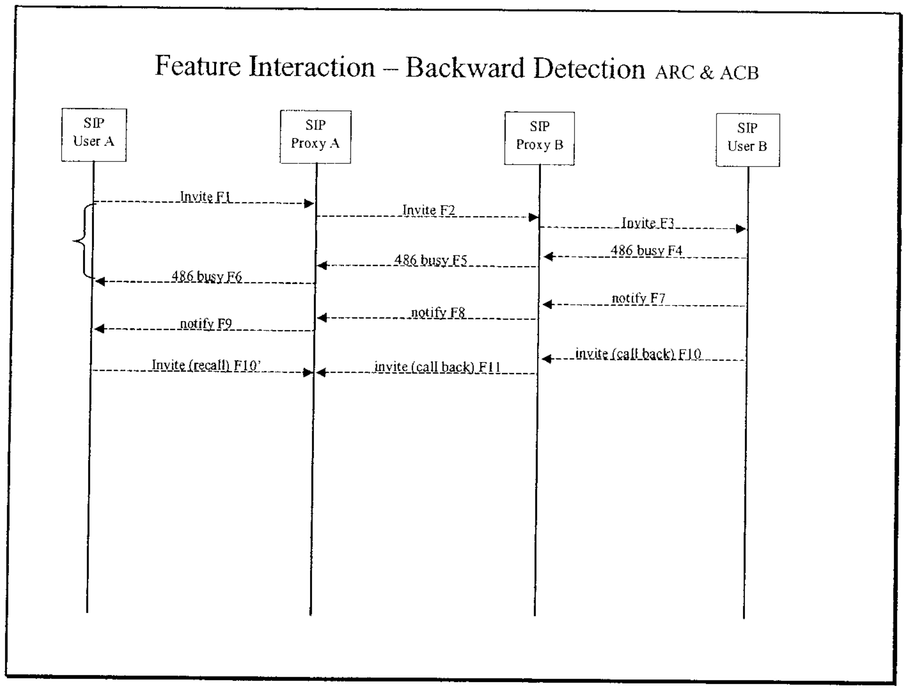


Figure 9 SIP ARC And ACB Flow

When user A makes a call to user B and user B is busy, SIP response message 486 is returned from terminating side. Once user A receives SIP 486 response message, it will not kill the session since feature ARC is triggered.

In step F7, terminating user B sends out SIP message "notify" [36] to originating side. This is to inform originating side that user B is now in "idle" state. At the same time, user B initiates automatic call back to user A. Upon receiving SIP "notify" message, user A initiates automatic recall to user B.

On originating side, the new "INVITE" message belongs to the same session as initial "INVITE" request, therefore the "Call-ID" header in both "INVITE" should be the same. On terminating side, the "INVITE" message generated by ACB feature should also use the same "Call-ID" as previous received "INVITE" message from user A.

The originating FIH can detect feature interaction by correlating user A's profile with received 486 response message and the "Call-ID" header of the new "INVITE" message.

4.5.3 Forward Detection

Forward detection, is dependent on the SIP request message (sent in forward direction). The detection point is on the call termination side. The FIH in domain B (as shown in Figure 5) is responsible for the feature interaction detection. The terminating FIH (located in domain B) needs to correlate the SIP request message, such as "INVITE", with the terminating user's profile (user B) to detect potential feature interaction.

4.5.3.1 Overview of Forward Detection Technique

Two scenarios are handled in this case. One is called "forward detection header info only" and the other is called "forward detection header info and incoming leg info".

The detection procedure is the following:

- Store and extract the received SIP "INVITE" request message and its headers.
- Look up user B's profile to get user's feature list.

- Both request message information from originating user and feature list from terminating user are crosschecked in the feature interaction category matrix as defined in the previous section, to determine if feature interaction will occur or not.

If feature interaction is detected, the FIH will not forward any message to servlet engine. FIH will construct a SIP "BYE" message to abort the ongoing session.

4.5.3.2 Forward Detection Header Info Only

"Forward detection header info only" discuss the scenario that the terminating FIH only checks the SIP "INVITE" message "Request-Disposition" header and terminating user's profile.

The forward detection is to detect feature interaction in the terminating side after receiving SIP request "INVITE" message. Due to the distributed architecture, the user A in the originating side might have features that will cause interaction with the terminating user B's features (Feature interaction type FI-AB as defined in early section).

There is no way for the terminating side to know if originating user A has any features that might cause feature interaction unless this information is made available to the terminating side. The current SIP protocol [29] doesn't support the user preferences. A set of extensions to SIP has been defined in Internet Draft " SIP Caller Preferences and Callee Capabilities " [34]. It allows a caller to express preferences about request handling in servers. These preferences include the ability to select which URIs a request gets routed to, and to specify certain request handling directives in proxies and redirect servers.

It does so by defining three new request headers, Accept-Contact, Reject-Contact and Request-Disposition. The extension also defines new parameters for the "Contact" header that describe the characteristics of a User Agent.

The Request-Disposition header field specifies caller preferences on how a proxy or redirect server should process a request. User agents do not process it. Its value is a list of tokens. Each value specifies a particular feature. When the caller specifies a feature, the server should treat it as a requirement and will process the feature request.

The header field has the following syntax:

```
Request-Disposition  = ( "Request-Disposition" | "d" ) ":"
                         1# (proxy-feature | cancel-feature |
                          fork-feature | recurse-feature |
                         parallel-feature | queue-feature)

proxy-feature        = "proxy" | "redirect"
cancel-feature       = "cancel" | "no-cancel"
fork-feature         = "fork" | "no-fork"
recurse-feature      = "recurse" | "no-recurse"
parallel-feature     = "parallel" | "sequential"
queue-feature        = "queue" | "no-queue"
extension-feature    = token
```

The “Request-Disposition” header provides relevant information regarding originating user’s service profile. This will help terminating FIH to determine if there will be any feature interaction.

The “forward detection header info only” algorithm is shown in Figure 10.

This algorithm has three key components as listed below:

- Request-disposition header information extraction
- Terminating user profile reading
- Feature interaction category matrix table lookup

Each component in this algorithm is for one specific task.

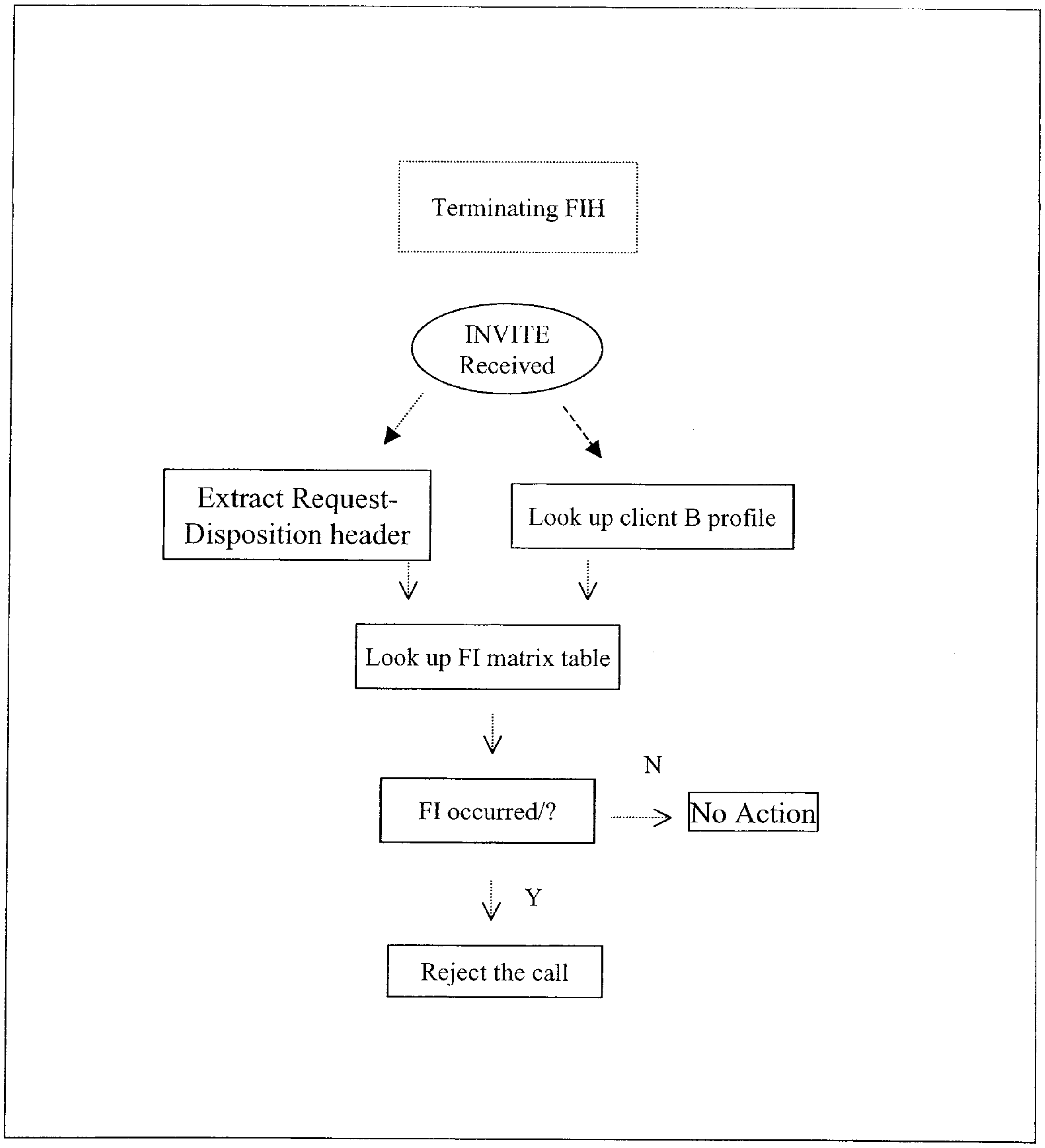


Figure 10 Forward Detection Header Info Only

4.5.3.3 Example of Delay vs. Forwarding

The typical example is where the originating user has "Camp-on busy" [Category – Delay] feature and terminating user has "Call Forwarding On Busy" [Category – Forwarding] feature.

The Figure 11 below shows the high-level signaling diagram.

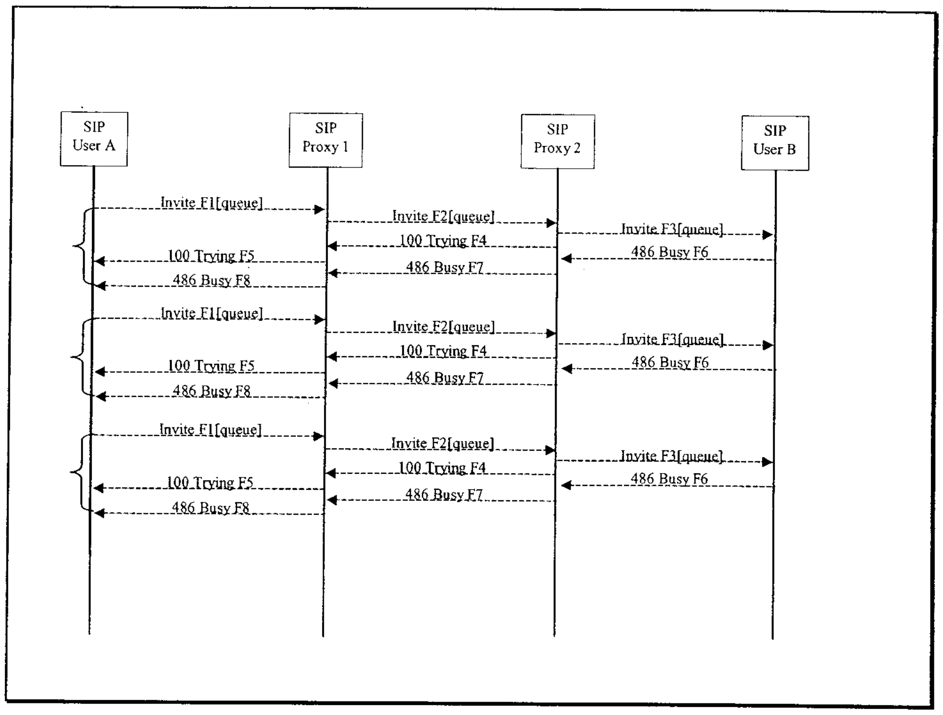


Figure 11 Forward Detection – Camp On And CFOB

In this scenario, the detection FIH will be the one in the terminating side.

Here we assume SIP user A initiates a call to SIP user B and user B is already engaged in a call. When the "INVITE" message arrives in domain B from user A, feature interaction will occur since user A has the capability of "camp-on" while busy, however user B would like to have the call transferred.

When user A sends the SIP "INVITE" message, the parameter “queue-feature” in the “request-disposition” header of the “INVITE" message is set to "queue". The FIH in domain B will correlate user A’s “INVITE” message header “Request-Disposition” (has “queue” indicator) with user B’s profile (has feature “CFOB”). Then the correlation is analyzed in the feature interaction category matrix, and feature interaction is detected.

4.5.3.4 Example of Multi-party vs. Forwarding

Another example as shown in Figure 12 is that User A has feature “Forking” [Category – Multi-party] and one of the defined forking destinations has “forwarding to voice mail” [Category – Forwarding]. Feature “forking” allows PROXY B to attempt to locate a user by forwarding a request to multiple destinations, for instance user B and user C. The call will be connected to the first destination to pick up, and the call attempt to the others will be cancelled. The feature interaction arises when one of the users to be reached, for instance user C, is currently located at domain B has had its calls forwarded to a voicemail system. The call to C will always be picked up first, as it is an automated system, and thus PROXY B will connect the call to user C and cancel the call to user B. The caller will never be able to reach the actual human.

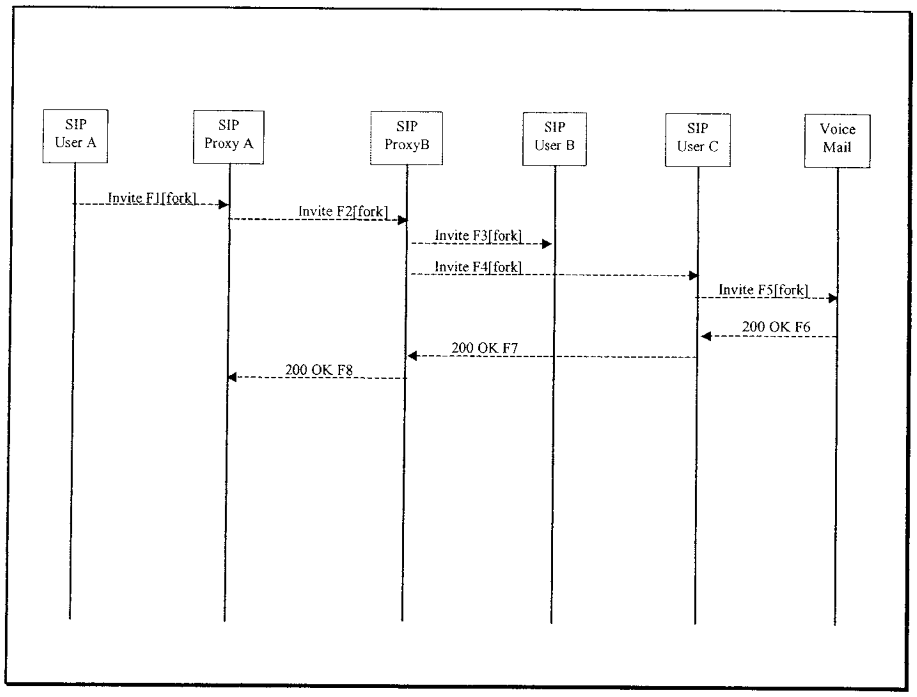


Figure 12 Forward Detection – Forking And Voice Mail

When user A sends the SIP "INVITE" message, the parameter “fork-feature” in the “request-disposition” header of the “INVITE" message is set to "fork". The FIH in domain B will correlate user A’s “fork” indicator (“INVITE” message header “Request-Disposition”) with user C’s profile, which has “forwarding to voice mail”. Then the correlation is analyzed in the feature interaction category matrix and feature interaction will be detected.

4.5.3.5 Forward Detection Header Info and Incoming Leg

“Forward detection header info and incoming leg” algorithm discuss the scenario that the terminating FIH checks not only the incoming “INVITE” message and terminating user's profile, but also the incoming call leg "history". The intension is to check what has happened to the incoming call leg before it terminates.

The algorithm is shown in Figure 13.

The algorithm has four key components as listed below:

- Request-disposition header info extraction
- General Header info extraction
- Terminating user profile reading
- Feature interaction category matrix table lookup

Each component in this algorithm is for one specific task. Comparing with "forward detection header info only" algorithm, the "forward Detection header info and incoming leg" algorithm has one more component, general header info extraction. The purpose of this additional component is to extract information related to the call leg "history". It provides additional information, such as if the call has been forwarded or not before it terminates. This information will help terminating FIH to detect potential feature interaction.

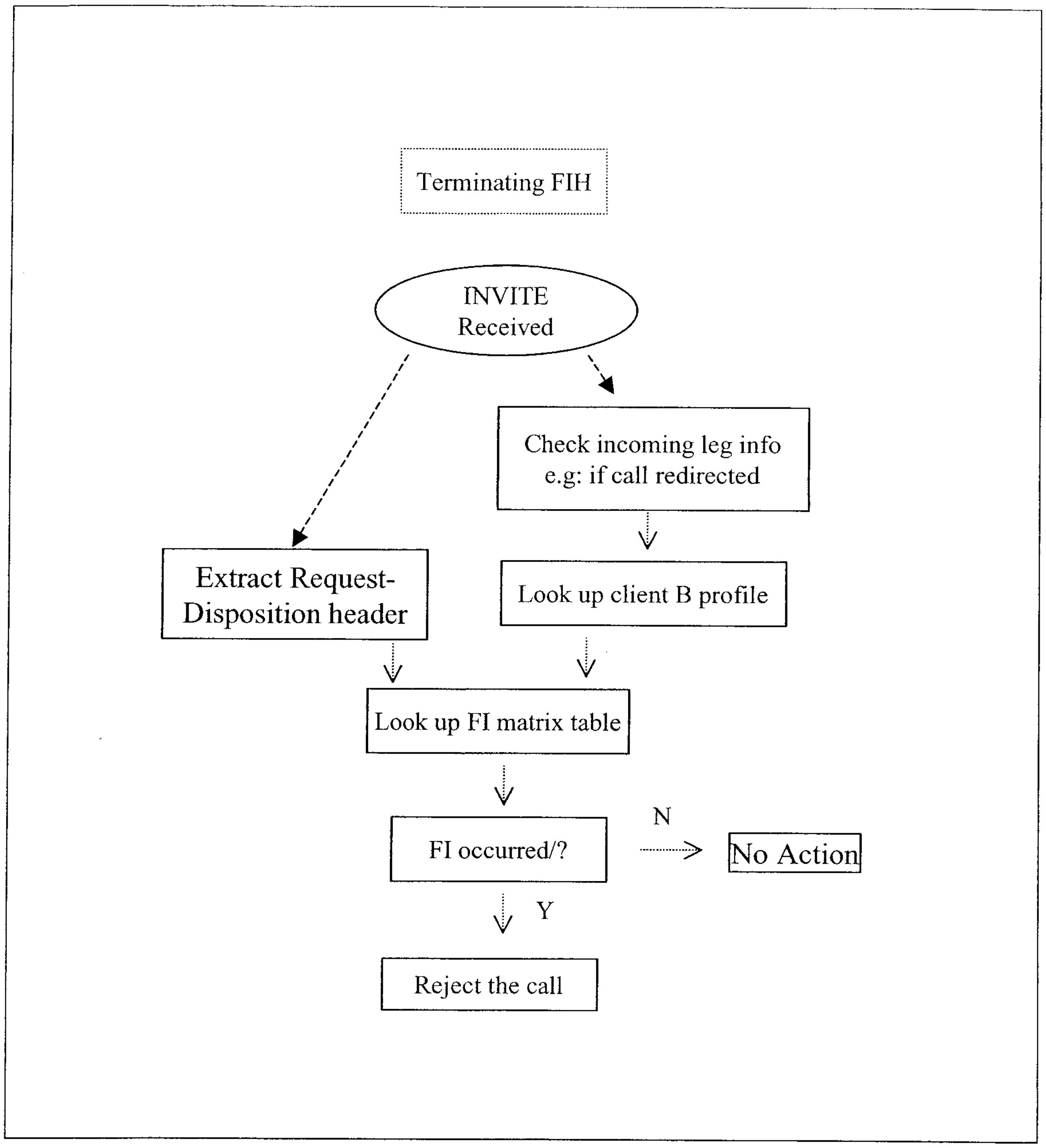


Figure 13 Forward Detection Header Info And Incoming Leg

4.5.3.6 Example of Forwarding vs. Authentication

Let's discuss another scenario where terminating user B has "Incoming Call Screening" [Category – Authentication] service activated. In the normal situation, every incoming call terminates at user B will be screened first before the termination process can actually happen. If the originating user A's number is on the user B's "forbidden" list, then the call will be rejected.

If however the call was redirected once during the call path, the originating user's ID in the "INVITE" message "from" will be the redirection point:

From: Redirection_Point <sip:User Redirection_Point@here.com>

If original originating user A is in the user B's "forbidden" list and redirecton_point user is not in the "forbidden" list. Then the call will be accepted and terminated in the user B. Thus it causes the feature interaction type FI-AB.

The detection algorithm here is similar to the "backward detection" algorithm, but the correlation is between the SIP request "INVITE" message and the terminating user B's profile. The general header "Via" of the "INVITE" message will provide the redirection information. If call forwarding occurred in the call path before terminating in PROXY B, the information will be recorded in the "Via" header of SIP "INVITE" message. The FIH in domain B will be able to detect the feature interaction and the call will be released.

4.6 Chapter Summary

In this chapter, we have discussed two types of feature interaction detection techniques, the offline detection technique and the online detection technique.

For the offline feature interaction detection, we introduced the concept of mapping features into feature categories. It groups features with similar behavior into a feature category. Instead of detecting feature interaction on a feature basis, we can detect interaction at the feature categories level. It significantly reduces the complexity of the task and number of cases to consider.

For online feature interaction detection, two methods are introduced, which are "forward detection" and "backward detection". "Forward detection" is to detect feature interaction on terminating side of the call session. It correlates the SIP "INVITE" request message with terminating user's profile to determine if feature interaction could occur. "Backward detection" is to detect feature interaction on originating side of the call session. It correlates the originating user's profile with the SIP response message to determine if feature interaction could occur.

Chapter 5

IMPLEMENTATION

5.1 Introduction

To validate the feature interaction detection techniques proposed in this thesis, we implemented the FIH for online detection and a tool for offline detection as discussed in previous chapter.

For the FIH, its interfaces and its architecture are described. For the offline tool, its functions and architecture are described. The system has been implemented using JAVA [37].

After the FIH and the offline detection tool have been implemented, we tested them with the “Bellcore benchmark” [11]and “European benchmark” [38].

5.2 Implementation Architecture

The general service architecture in Figure 4 shows the end-to-end call flow components. The architecture includes SIP client and SIP PROXY server. For SIP client, it has a SIP parser, an SDP parser and a SIP stack. For PROXY server, it has a SIP parser, an SDP parser, a SIP stack and a servlet engine. The

PROXY server is implemented using SIP client with "SIP user agent server" function.

The architecture of the PROXY server stack is shown in Figure 14.

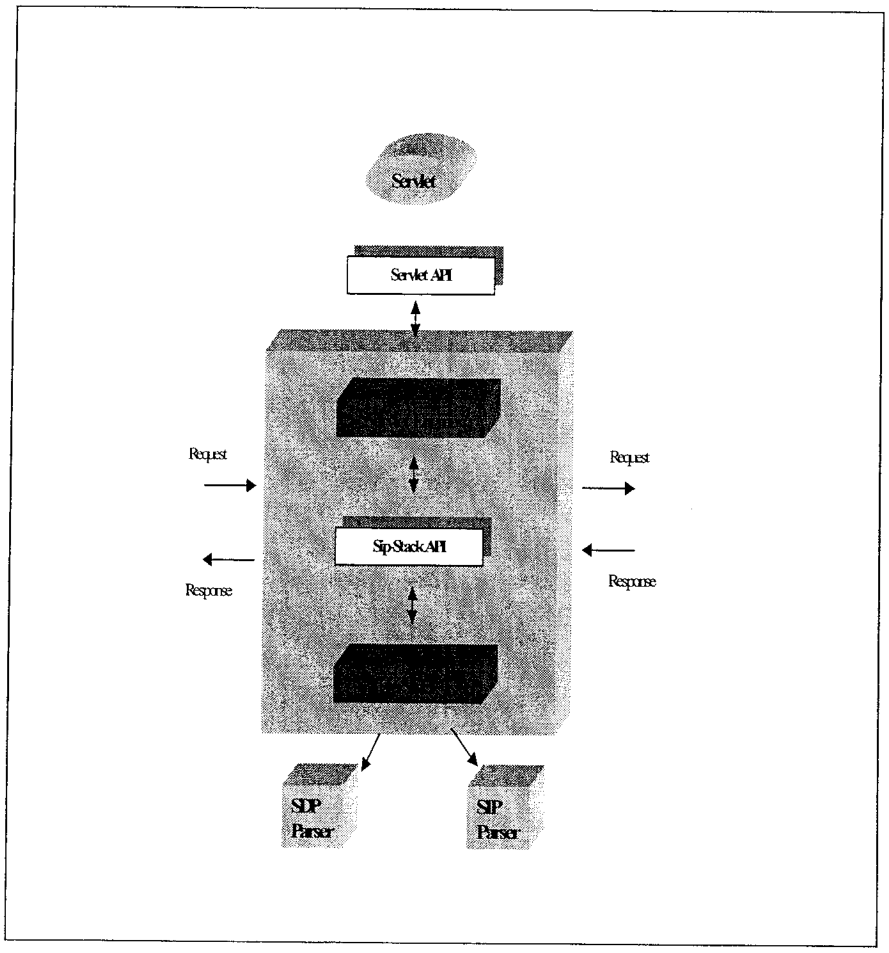


Figure 14 PROXY Stack

Then the end-to-end call flow is shown in Figure 15.

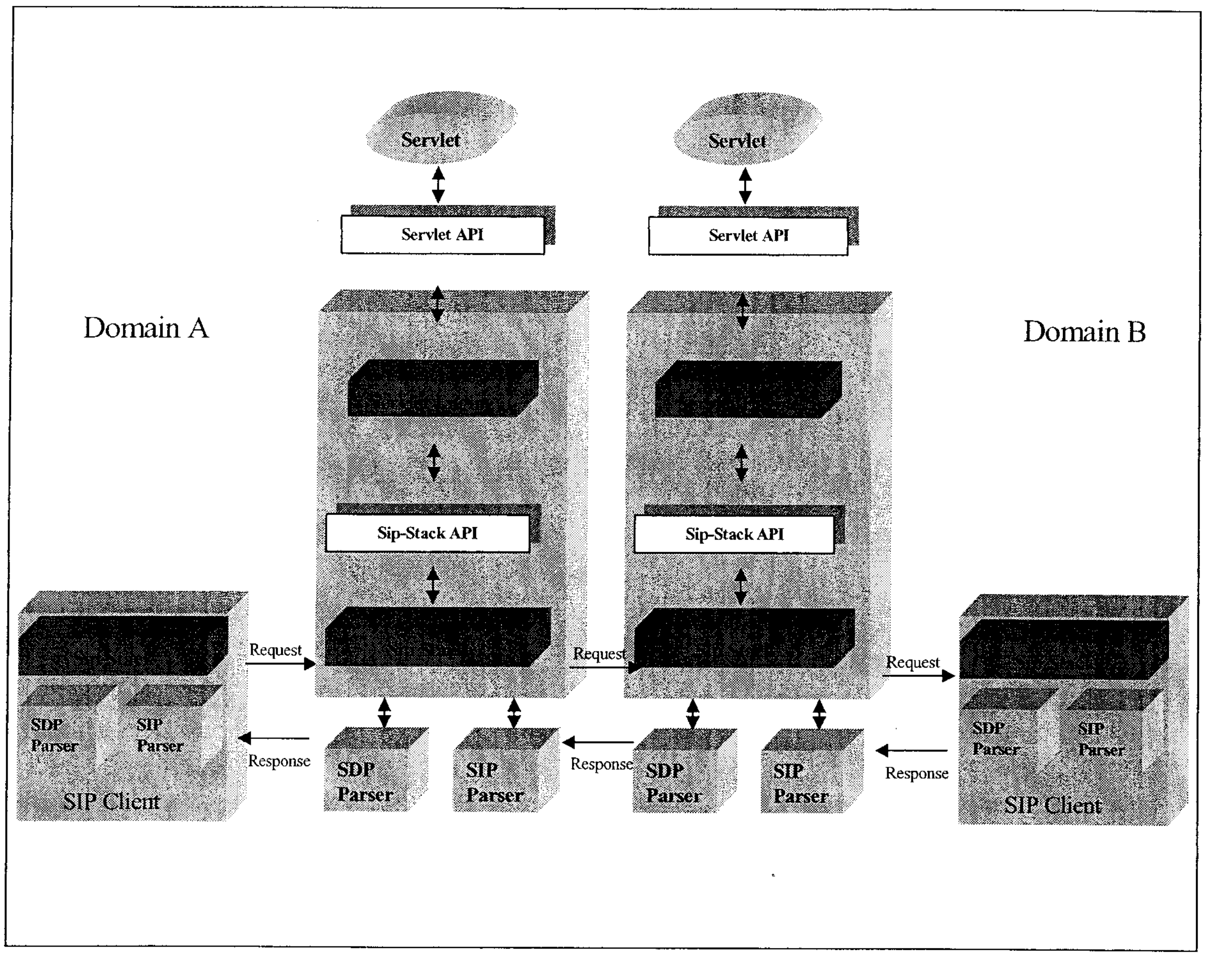


Figure 15 End-to-End Call Flow Stack View

As discussed in the previous chapter, FIH is an integral part of the servlet engine. It detects and manages feature interactions. The modified PROXY stack (with FIH included) is given in Figure 16.

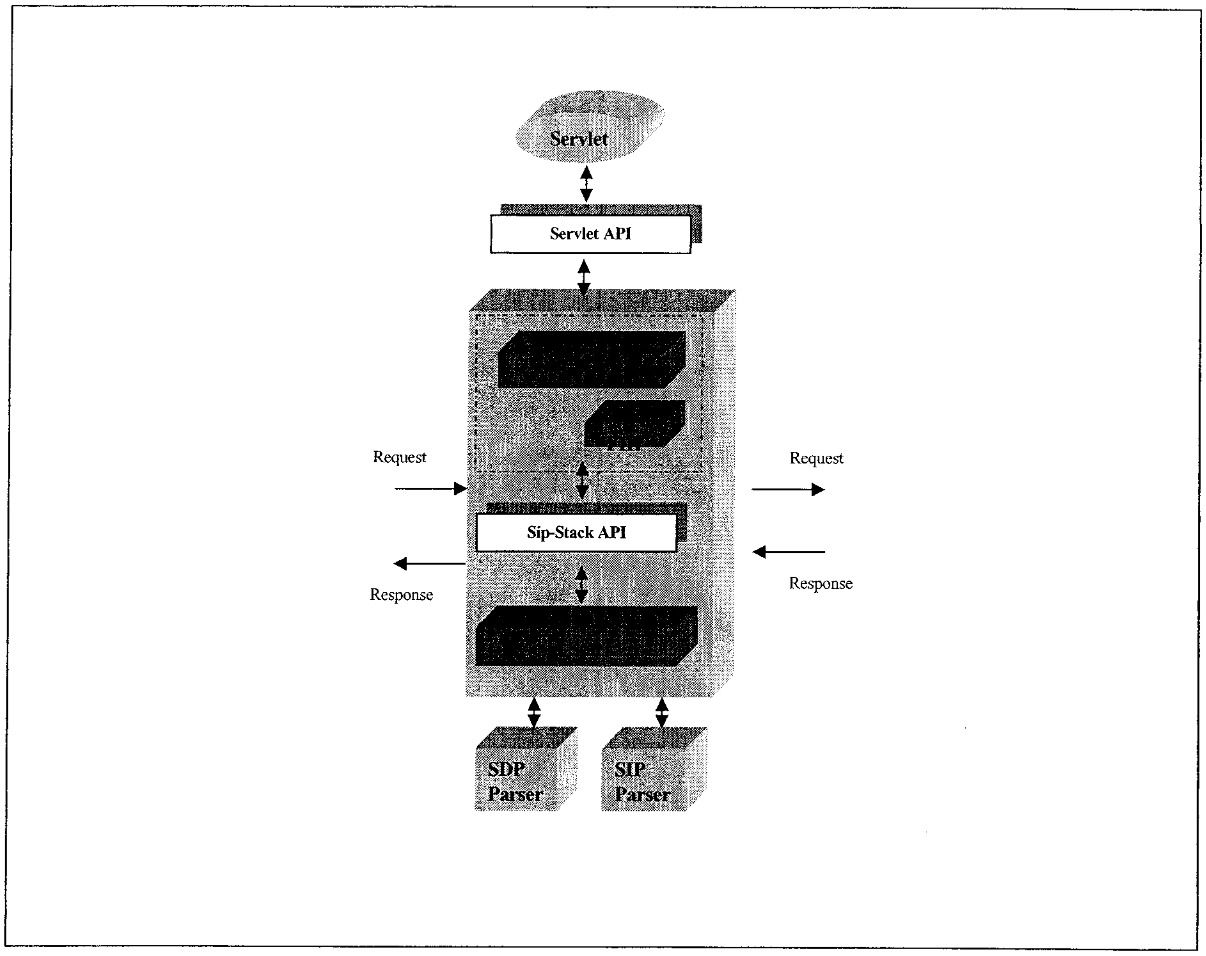


Figure 16 PROXY Stack With FIH

As proposed in Figure 16, FIH resides between the servlet engine and the SIP stack. This way, it allows FIH to check every incoming/outgoing SIP request/response message whenever feature servlet is to be invoked.

At the same time, FIH does not interfere with servlet engine if there is no feature interaction detected. If there is no interaction detected, the servlet engine will execute individual servlet as usual. If interaction is detected, the FIH will stop the servlet engine process and sends a SIP "4xx" response message to disconnect the ongoing "call" session.

The modified end-to-end call flow "stack view" (with FIH included) is shown in Figure 17.

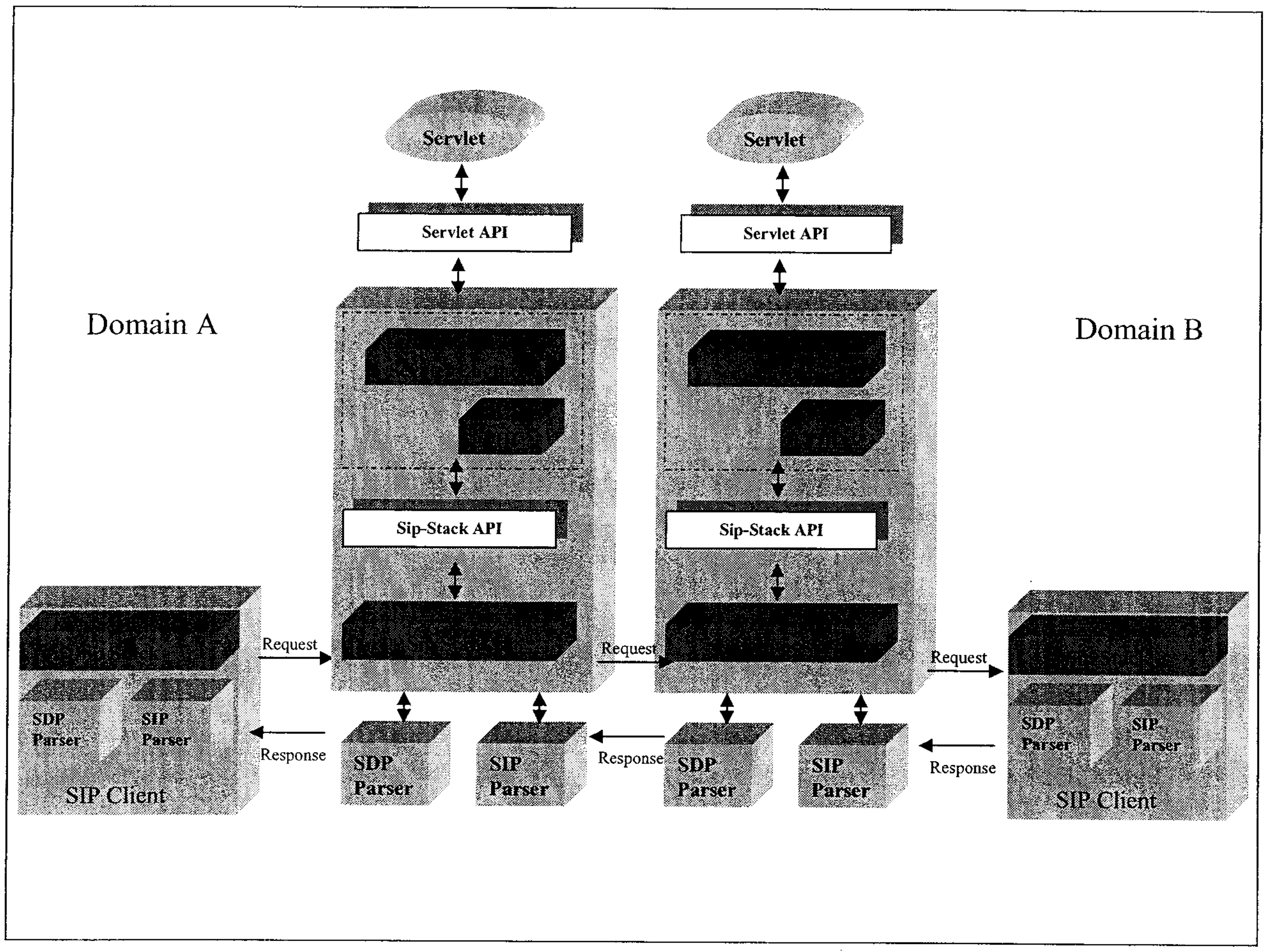


Figure 17 End-to-End Call Flow Stack View With FIH

5.3 *Online Feature Interaction Detection Architecture*

In this section, we will discuss online feature interaction detection architecture. This includes the FIH and the feature servlet. The SIP stack (including Parsers) and Servlet engine implementation are not covered in this thesis. Both entities are using the existing product.

5.3.1 Interface Description

The FIH interfaces directly with the SIP stack and the servlet engine. This interface follows the standard SIP API [39].

The interface supports the following SIP request messages.

- BYE – Used in both “backward” and “forward” detection
- INVITE – Used in both “backward and “forward” detection
- NOFITY – Used in “backward” detection

SIP “CANCEL” and “ACK” message are not required from FIH perspective. There are no features in this thesis utilizing “CANCEL” message. In case the SIP stack receives it, it will not invoke any feature servlet. Therefore SIP “CANCEL” message will not be supported in the FIH.

Some features may require SIP “ACK” message. It is used to acknowledge the reception of “200 OK” response message. However, the detection algorithms discussed in this thesis do not use information from SIP “ACK” message. Thus the SIP “ACK” message will not be supported in the FIH.

The interface supports the following SIP response messages:

- 1xx
- 2xx
- 4xx

All other SIP response messages (3xx, 5xx and 6xx) are not relevant from the FIH perspective. The interaction detection algorithms discussed in Chapter 4 does not use any of them.

As defined in IETF RFC 3261 [29], there are over 40 different parameter fields. With more extension defined, such as Internet Draft "SIP Caller Preferences and Callee Capabilities" [34], the total number of parameter field keeps increasing. This thesis only address limited amount of parameter fields.

The interface supports the following SIP parameter fields:

- Call-ID
- Contact
- Cseq
- Request-Disposition
- From
- To
- VIA

5.3.2 Function Description

All functions specified in online detection algorithm in Chapter 4 are covered and included as below:

- Feature category interaction matrix – This matrix defines all possible interactions between feature categories.

- SIP message header field extraction – This is to store the received SIP message parameter fields. Input is the message ID and parameter field ID. There is no output for this function.
- SIP message header field comparison – This is to compare the stored SIP message header fields within the same session. The input to this function is "session ID", SIP message parameter field identity. The output is the service indicator, which depends on the SIP message parameter field comparison result.
- User profile look up – This is to get the user profile. The profile should contain all features that user has registered and other relevant information. The input is user identity and output is the feature list.
- Feature category interaction matrix table look up – This is to get the feature interaction detection result. The input is the feature list and output is the feature interaction detection result.
- Session controller – This is to keep track the session status of each call. The input is "Call-ID" and "Cseq" of each received SIP request/response message.

5.3.3 FIH Structure

We propose four modules to implement the FIH, which are Transaction manager, Call leg, User Profile and Feature Category Interaction Matrix. Figure 18 shows the FIH structure.

The Transaction manager module extends the SIP API. It is responsible to keep track all received/sent SIP messages.

The Call leg module extends transaction manager. It extracts and stores the parameter fields of received SIP messages; compares the same parameter field between different messages for the same session.

The User profile module is responsible for user profile modification. It includes adding/deleting of feature and returning feature list of a user.

The Feature category interaction matrix is to analyze the received session info and detect if there is potential feature interaction. When the analysis is finished, it will return the feature interaction detection result.

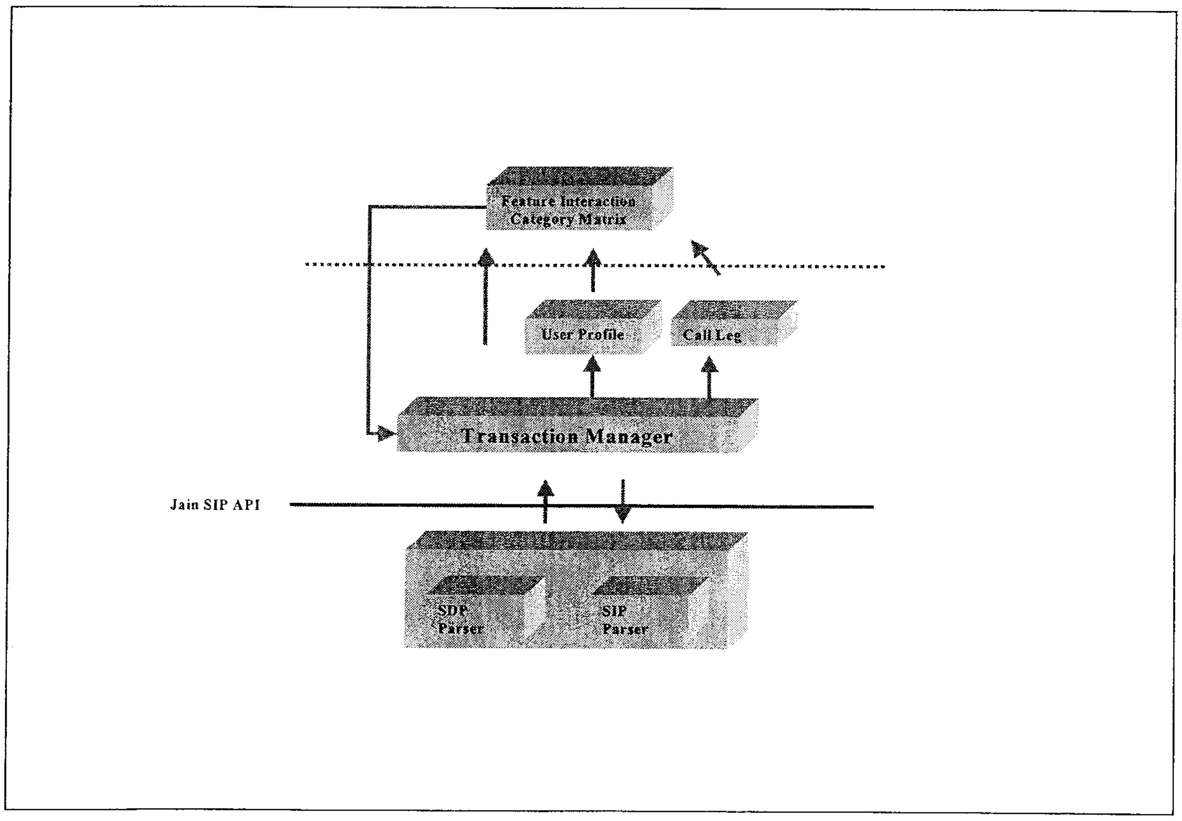


Figure 18 FIH Structure

5.3.3.1 Transaction Manager

A SIP transaction occurs between a user and a server. It comprises all messages from the first “Request” sent from the user to the server up to a “final response” sent from the server to the client.

The “Cseq” sequence number within a single call leg identifies a transaction.

A transaction can be in one of the following states.

INITIAL: a transaction is set to INITIAL state when an INVITE, BYE message is sent or received.

INPROGRESS: a transaction is set to INPROGRESS state when a response 1xx is sent by the server to indicate the progress.

FINAL: a transaction is set to FINAL state when receiving a final response 2xx, 4xx.

Classes and methods for Transaction Manager are:

SipTransactionManager: SIPTransactionManager class is a skeleton class. It manages all transactions that are sent and received by the SIP stack. It is responsible for matching responses to requests and mapping messages to the transaction they are associated with. Methods provided by this class are:

- **addTransaction()**: adds a transaction into the transaction map.
- **removeTransaction():** removes a transation from the transaction map.
- **getTransaction():** returns the client transaction if it exists in the transaction map.

- **UpdateTransaction():** updates the transaction map with the message received to an existing transaction in the transaction map

SipTransaction: an abstract base class for the client and server transaction classes. A transaction consists of all messages from the first request sent from the client to the server up to a final response sent from the server to the client.

- **getTransactionId():** returns the transaction id of the current transaction.
- **getRequestMessage():** returns transaction's initial request.
- **getTransactionState():** returns transaction's state.
- **setTransactionState():** sets transaction's state.

SipClientTransaction a client transaction is created when a request message (INVITE, BYE) is sent from the client to the server.

- **sendRequest():** sends a request message
- **processResponse():** processes a response message associated with the current transaction

SipServerTransaction a server transaction is created when server receives a request message (INVITE, BYE) from the client.

- **processRequest():** processes a request message
- **sendResponse():** sends a response to the request associated with the current transaction

5.3.3.2 Call Leg

Call Leg is handling all detailed information related to a “transaction”. It extracts and compares information received between different states of the call.

Call Leg API hides the creation and manipulation of SIP transactions. One Call Leg consists of one or more transactions. A typical call has at lease one call leg that comprises two request messages (or transactions) “INVITE” and “ACK”.

Methods that are supported by the class **CallLeg** are:

- **getCallID():**returns CallID of the call leg
- **getFromHeader():**returns “From” header of the call leg
- **getToHeader ():**returns “To” header of the call leg
- **getContactHeader ():** returns “Contact” header of the call leg
- **getViaHeader ():** returns “Via” header of the call leg
- **getRequestDispositionHeader ():** returns “Request-Disposition” header of the call leg
- **compareFromHeader():** compares “From” header between request and response in the same session
- **compareToHeader():** compares “To” header between request and response in the same session
- **compareViaHeader():** compares “Via” header between request and response in the same session

5.3.3.3 User Profile

UserProfile handles user profile modification and extraction. Methods supported by this class are:

- **getUser();** gets user identify from the message received from entered from offline tool
- **addUser():** adds new user into the user database
- **deleteUser();** deletes user from the user database
- **getUserList();** returns current user list from database
- **addFeature():** adds new feature into a user profile
- **deleteFeature():** deletes feature from a user profile
- **getFeatureList():** returns a user's feature list

5.3.3.4 Feature Category Interactions Matrix

InteractionMatrix handles the feature category interaction detection. Methods supported by this class are:

- **addFeatureCategory();** adds new feature category into the matrix
- **deleteFeaturecategory():** deletes feature category from the matrix
- **featureToCategory();** maps feature to feature category
- **getFeatureCategory();** returns the feature category list from the matrix

- **detectFeatureInteraction();**returns the feature interaction detection result based on the pre-set logic
- **bye():** sends a SIP "4xx" response message for the "INVITE" message that initiated the call leg

5.3.3.5 FIH Class Structure

The overall FIH class structure shown in Figure 19.

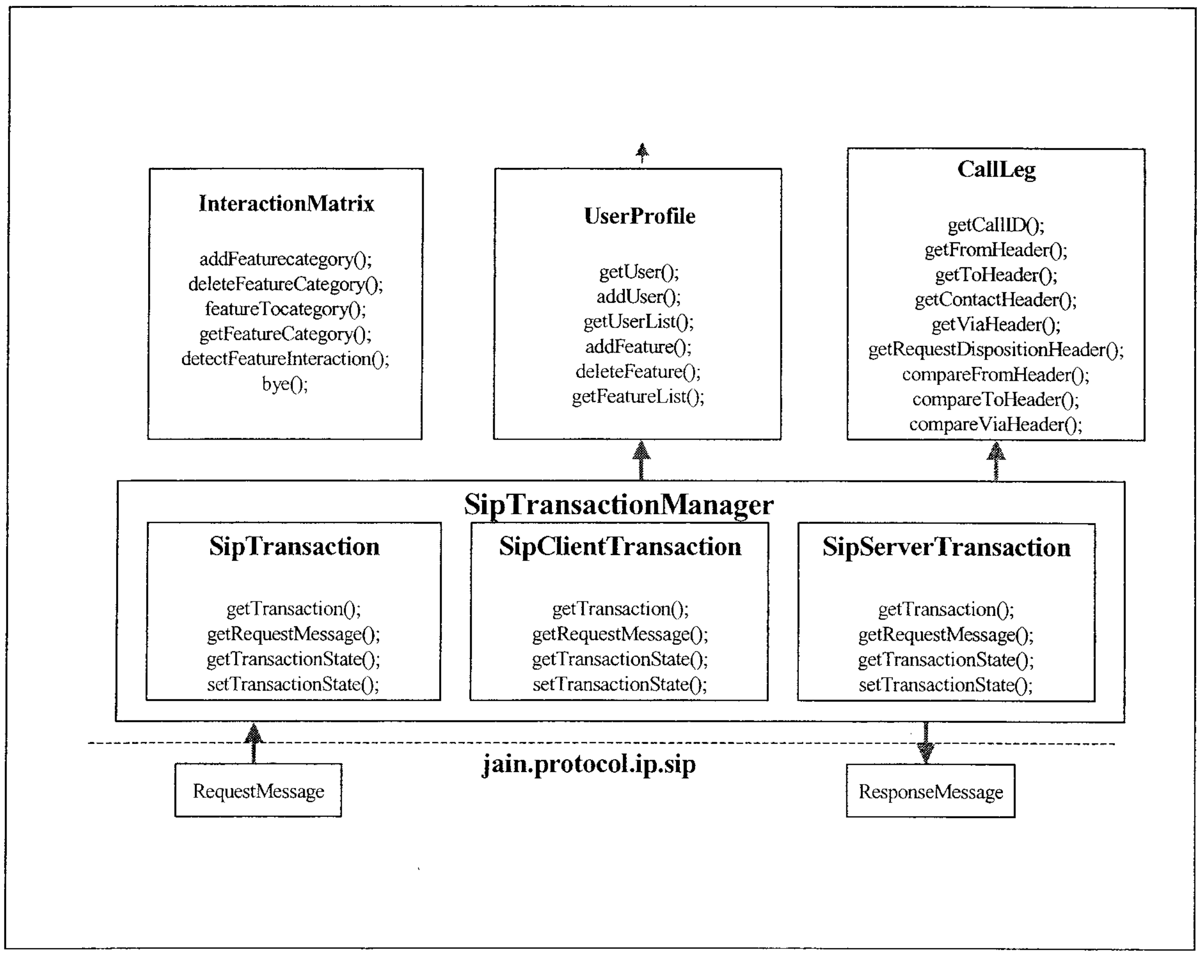


Figure 19 FIH Class Structure

5.3.4 Feature Servlet

In order to simulate individual feature in SIP servlet environment, we developed two feature servlets, which are "Call Forwarding" and "Outgoing Call Screening". Feature servlet is independent from feature interaction detection techniques analyzed in this thesis.

The **javax.servlet** package and the **javax.servlet.sip** package from SIP Servlet API provide the classes and interfaces to define servlets. The **SipServlet** abstract class defines additional methods for handling SIP requests. These methods correspond to the request methods defined by SIP specification. The service method of the **SipServlet** class automatically calls these methods for processing a SIP request. The **SipServlet** interface defines the following methods for request processing:

- **doInvite();** for handling SIP INVITE requests
- **doAck();** for handling SIP ACK requests
- **doOptions();** for handling SIP OPTIONS requests
- **doBye();** for handling SIP BYE requests
- **doCancel();** for handling SIP CANCEL requests
- **doRegister();** for handling SIP REGISTER requests
- **init();** for initiating SIP servlet
- **destroy();** for destroying SIP servlet

The Figure 20 below shows the servlet structure.

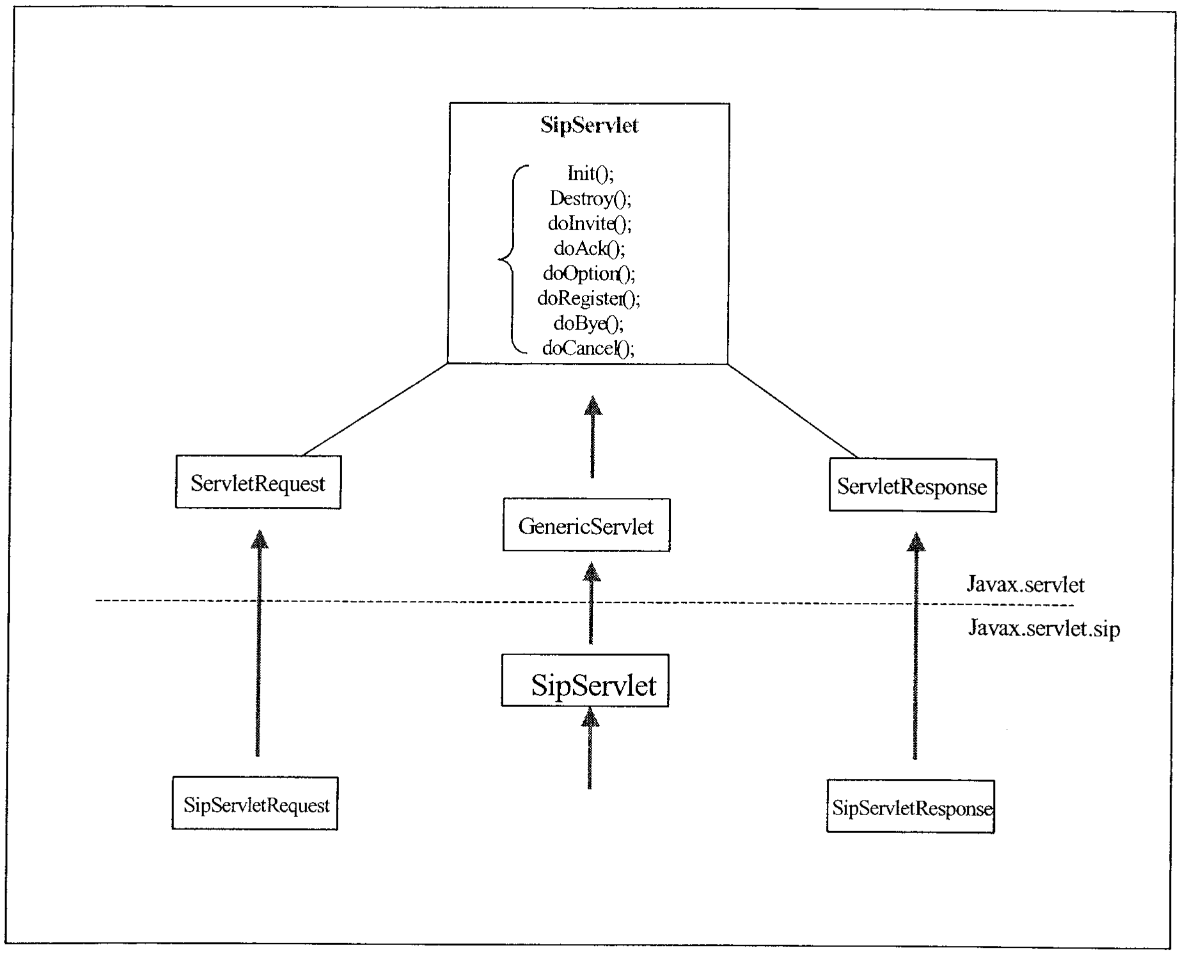


Figure 20 Servlet Structure

The doInvite method is the heart of the servlet, where instructions and the purpose of the servlet are carried out.

The doInvite method instructs the servlet engine about what it must do, whether playing a media, writing to a database, or simply redirecting the client to a requested URL. Within this method we use Java programming syntax to give specific instructions for the servlet to interact between the client and the server [40].

5.4 *Offline Tool Architecture*

Offline tool is developed for offline feature interaction detection. The objective is to demonstrate the offline feature interaction detection principle. The feature interaction detection point is when assigning features to the selected user. The tool has the following input and outputs:

- It provides a Graphic User Interface
- It can add or delete users from the database
- It shows available feature list for the selected user
- It is able to assign the selected features to a user if no interaction detected
- It is able to generate error message if feature interaction is detected.

5.4.1 Offline Tool Class Structure

Class **FiCheckerTool** extends the class **UserProfile**. Methods supported by this class are:

- **getFeatureList();** returns the available feature list
- **getUserList();** returns all the users in the database
- **getUserProfile();** returns the user profile
- **addFeature();** assign feature to user

- **deleteFeature();** delete feature from user
- **returnResult();** returns the feature interaction detection result

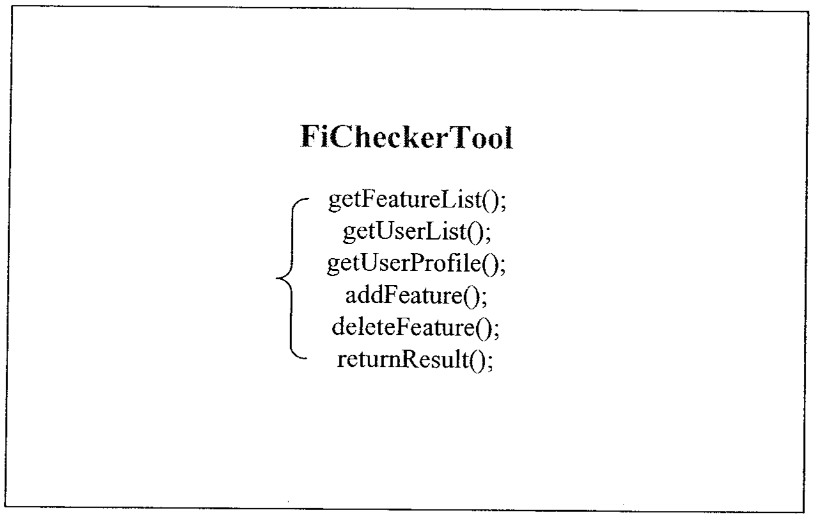


Figure 21 Offline Tool Class Structure

5.5 Programming Environment

The following programming tools are used.

- The programming is using Borland Company IDE Jbuilder 5 professional edition
- JDK 1.3.1(JAVA 2) [41]
- JAVA Media FrameWork 2.1.1a [42]
- Servlet Tomcat [43]

5.6 Application

This section shows the implementation results and validation of the detection algorithm. For the FIH, we apply the feature interaction benchmarks and check the feature interaction detection result. If the relevant feature interaction can be detected, it will prove the viability of both feature interaction detection algorithm and its implementation.

For offline tool, we will show several snap shot examples of the tool usage. The examples will demonstrate how to list users in the database, how to assign features to a user and how feature interaction is detected.

5.6.1 Bellcore Benchmark

The Bellcore Benchmark [11] presents two different ways of categorising feature interactions:

- By nature of the interaction which is divided into:
    - Type of features involved
    - Number of users involved
    - Number of network components involved
- By causes of the interaction

In this benchmark a collection of features and their interactions have been presented. We used some of the feature in this benchmark and their combination to test our approach.

5.6.2 European Benchmark

In the EURESCOM project P509 [38]. A benchmark for evaluating the approaches proposed and elaborated in the project has been developed. This benchmark is based on a new definition of feature interaction from which a classification of interactions is inferred in a rather straightforward manner. Same as Bellcore benchmark, a set of features has been introduced and a collection of experienced feature interactions has been presented. We use some of the features in this benchmark and some of their combinations to verify our approach.

5.6.3 FIH Application

Since the FIH is an implementation of our detection algorithm, we use it as a mean to validate the feature interaction detection. In the section below, part of the combinations from both benchmark documents are checked and presented. In each case, the feature interaction detection method is explained.

Since we have only implemented two feature servlets, "Call Forwarding" and "Outgoing Call Screening", therefore part of the validation are based on analysis only.

We did not validate all the feature interaction as defined in the benchmark document. There are several reasons.

- It is not possible to pass some of the calling user's profile or its feature status without extends the SIP "INVITE" message. Only limited information is available to terminating side based on the current SIP standard.
- The SIP response message 1xx, 2xx has limited information available to the originating side. To support our approach, it would also require extension to the SIP response message.
- The PROXY "Forking" function will further complicate the feature implementation and feature interaction situation.

European Benchmark:

- Connected Line Presentation (COLP) / Connected Line restriction (COLR)
    - Feature interaction cannot be detected since the current SIP extension doesn't support user preference that "doesn't display number". However, if such an extension is available, then feature interaction can be detected through the "Forward Detection Header Info Only" Algorithm. The feature interaction type is **FI-AB.**
- Originating Call Screening (OCS) / Abbreviated Dialling (ABD)
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA.**
- Originating Call Screening (OCS) / Abbreviated Dialling (ABD)
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA**.

- Call Waiting (CW) / Three-way Calling
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA**.
- Call Waiting / Call Forwarding on busy (CFB)
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA**.
- Calling Number Delivery (CND) / Unlisted Number (UN)
    - Feature interaction cannot be detected since the current SIP extension doesn't support user preference that "doesn't display number". However, if such an extension is available, then feature interaction can be detected through the "Forward Detection Header Info Only" Algorithm. The feature interaction type is **FI-AB**.
- Do not Disturb (DND) / Emergency Response Service
    - Feature interaction can be detected through "Forward Detection Header Info Only" Algorithm. Normally "Emergency Response Service" has a special digit pattern and it can be used as feature indication. The feature interaction type is **FI-AB**
- Call completion on busy (CCBS) / Call Back
    - Feature interaction can be detected through "Backward Detection". The feature interaction type is **FI-AB**.
- Call Waiting (CW) / Answer Call (AC)
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA.**

- Call Forwarding Unconditional (CFU) / Terminating Call Screening (TCS)
    - Feature interaction can be detected through "Forward Detection Header Info and incoming leg" Algorithm. The feature interaction type is **FI-AB.**
- Wake up call (WUC) / Call Forwarding On No Reply (CFNR)
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA**.

Bellcore Benchmark:

- 911 (emergence call) / Three-way call
    - Feature interaction can be detected through "Forward Detection Header Info Only" Algorithm. "911" is treated as a special dialled digit string and used as feature indication. The feature interaction type is **FI-AB.**
- Terminating Call Screening (TCS) / Automatic Recall (ARC)
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA.**
- Originating Call Screening (OCS) / Area Number Calling (ANC)
    - Feature interaction can be detected at offline stage. The feature interaction type is **FI-AA.**
- Operator Service / Originating Call Screening (OCS)
    - Feature interaction can be detected through "Forward Detection Header Info Only" Algorithm. Normally all operator service has a

special numbering pattern and it can be used as operator service feature indication. The feature interaction type is **FI-AB.**

- Long Distance Calls / Message Rate Charge Service
    - Feature interaction cannot be detected since it is related to service provider administrative area. The service level agreement has to be reached before cross-network inter-working is possible. Therefore, there is no need for either offline or online feature interaction detection.
- Calling from Hotel / Message rate Charge Service
    - Cannot be detected since it is related to service provider administrative area. The service level agreement has to be reached before cross-network inter-working is possible. Therefore, there is no need for either offline or online feature interaction detection.
- Call Forwarding Unconditional (CFU) / Originating Call Screening (OCS)
    - Feature interaction can be detected through the "Backward Detection". Feature interaction type is **FI-AB.**
- Automatic Call Back (ACB) / Automatic Recall (ARC)
    - Feature interaction can be detected through the "Backward Detection". Feature interaction type is **FI-AB.**
- Originating Call Screening (OCS) / Multiple Directory Number Line with Distinctive Ringing
    - Feature interaction cannot be detected. If different number, which is associated with the same pilot number, is used as called directory

number, there is no way for originating side to know. There is also no mechanism for terminating side to pass the information back to the originating side.

- Call Waiting (CW) / Automatic Call Back (ACB)
    - Feature interaction can be detected at offline stage. The feature interaction type **FI-AA.**

5.6.4 Offline Tool Demonstration

The offline tool provides a graphic user interface as shown in Figure 22 below. The offline tool includes two parts, one is user definition and the other is feature assignment to a particular user. The user definition is used to define or delete individual users in a particular service provider domain. The user is recognized by its name, which can be any combination of character. Once user is defined in the network, we could assign certain features to the user, which is done through the feature assignment part.

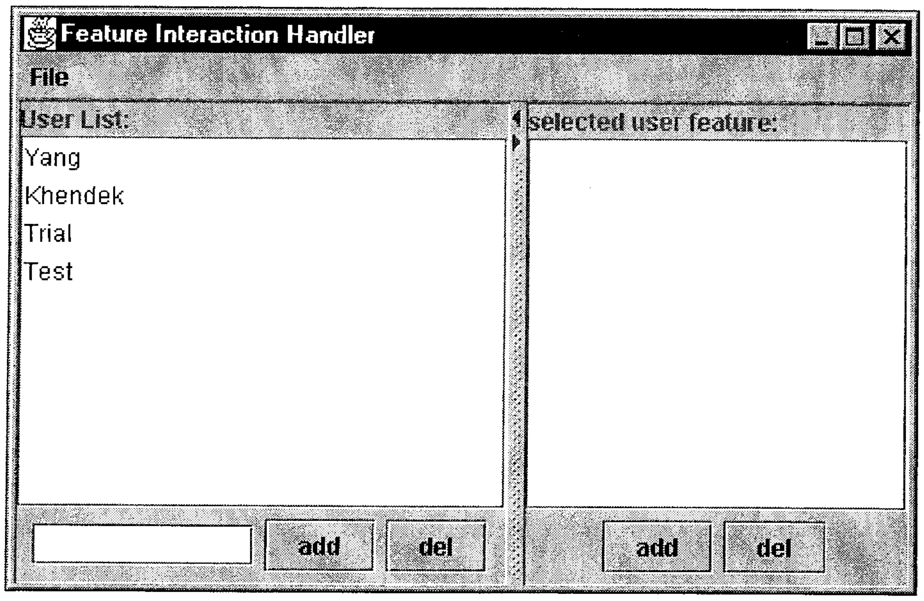


Figure 22 Offline Tool

When adding features to a particular user, simply clicking on the user name on the left and clicking the "add" button on the right side of the graphic user interface. Depending on the current status of the user, a list of allowed features would pop up as shown in the Figure 23 below.

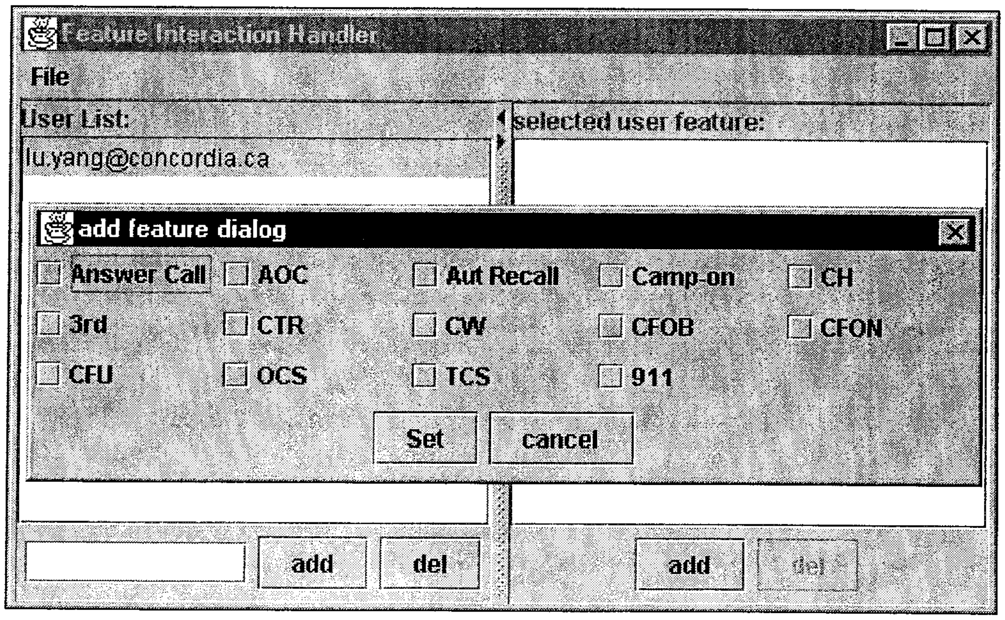
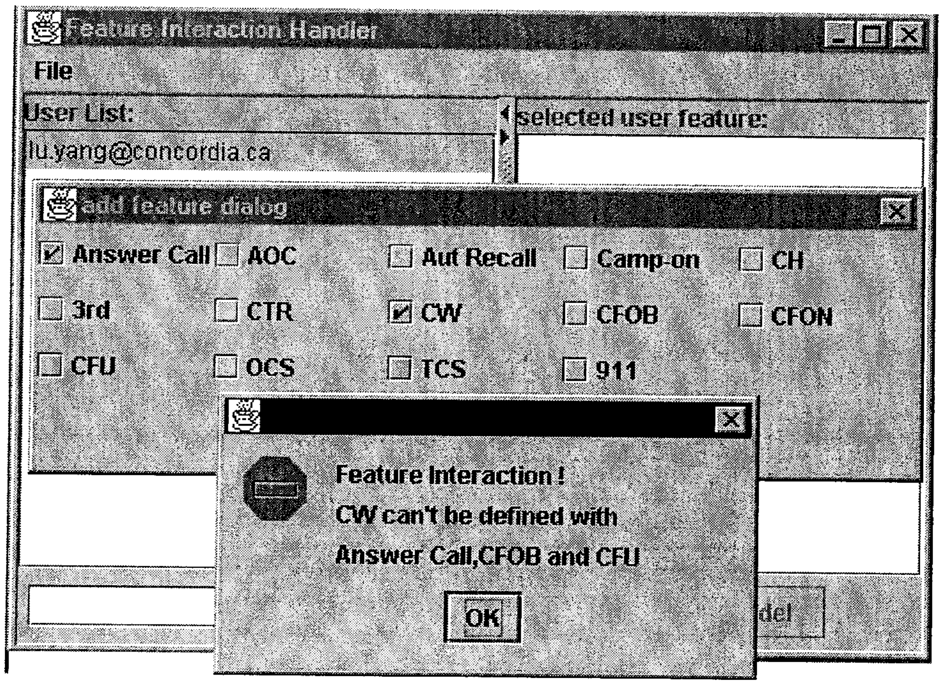


Figure 23 Offline Tool Dialogue

In this case, the user lu.yang@concordia.ca is newly defined and has no feature defined in the database. So all available features are available for use. In case user would like to have, for instance "Answer Call" and "CW" together, feature interaction will occur. For the simulation purpose, a warning message “Feature Interaction!” will be displayed. The features have been selected are also shown as part of the warning message. This is shown in Figure 24.



Feature Interaction !
CW can't be defined with
Answer Call,CFOB and CFU

Figure 24 Offline Tool Usage Example

5.5 Chapter Summary

In this chapter, both online detection architecture and offline tool architecture are explained in details. For the online detection architecture, the focus is given to the FIH implementation. The FIH interface/function description, the component structure and the class structure are described. For the offline tool, architecture and class structure are specified.

We also presented the application of FIH and offline tool to detect known feature interactions for validation purpose. The validation includes both European and Bellcore benchmarks. Detection results were shown and explanations were given.

Chapter 6

CONCLUSION

6.1 Main Contributions

IP telephony is becoming the new leader in the telephony world. The current industry trends show that more and more service providers or operators are transforming their network from traditional circuit based to IP. It is especially true for the new 3rd generation wireless network. Day by day the number of service providers interested in IP telephony is increasing. Users also demand new capabilities and services that must be provided by the service providers using the best tools to achieve their objectives.

The conventional feature interaction will still be a problem in the SIP environment and moving to an IP world will not simply take it away. It is true that the richness of SIP protocol may get rid of some of the existing feature interactions though not all of them, but at the same time it also creates new feature interactions with the introduction of new capabilities and functions.

In this thesis, we have proposed an approach to address the feature interaction detection issue in SIP servlet environment. It is proven through the project that our solution of handling feature interaction detection in the SIP servlet environment is viable. The detection algorithms have been validated by applying them to both Bellcore and European Benchmarks.

However, its performance and efficiency is not part of the project scope and thus not verified during the project. It can be a good candidate for future study. Since the solution proposed is on the handling of low-level SIP signaling, it can be potentially also used in a generic SIP service environment, not necessarily bundled with the SIP servlets.

6.2 Main Advantages Of Our Approach

The proposed method is a pragmatic method; it is based on the experience, the understanding of the feature behavior and the observed cases.

Our approach has the following advantages:

- It has the simplicity of the pragmatic methods.
- The approach covers all feature interactions introduced by European Benchmark
- The tool is user friendly and provides useful prompts and messages for the user
- The detection algorithms are simple.

6.3 *Potential Extensions And Future Work*

The developed approach is based on SIP servlet concept and its components. To develop the approach further, a study of new SIP service is required. The study will probably lead to an extension of the approach and the tool.

New Feature Handling:

Due to distributed feature/service environment in SIP protocol based network, it is possible that many new features, which we have no idea about at this stage, will cause new type of interactions. It is impossible to predicate what type of feature interactions will occur and how to manage them. This issue, it is not addressed in this thesis and should be explored in a future study.

Feature Interaction Resolution:

As a complete solution for the feature interaction problem, a detection approach must be followed by a resolution approach. Since the current approach and the FIH provide all required information about the place and reason of interactions between the features, a resolution approach can simply use this information to completely solve the problem.

Combination of more than two features:

Despite the fact that, through this thesis we have mostly chosen the combination of two features in order to describe different situations, our approach does not have any restriction related to the number of features. During the implementation, our effort was aimed toward adding the required functionality to different components in order to handle the combination of more than two features. It is worth to carefully investigate the possibility of fully extending the FIH in handling the combination of more than two features.

REFERENCES

[1] "Principles of Intelligent Network Architecture", ITU-T Recommendation Q.1201, Geneva, 1992

[2] Bellcore Technical Advisory, "Switching Systems Generic Requirements", Advanced Intelligent Network (AIN) Release 1, May 1991

[3] H. Schulzrinne and Jonathan D. Roseberg, "Internet telephony: Architecture and protocols – an IETF perspective", Computer Networks and ISDN Systems, vol. 31, pp. 237–255, Feb. 1999

[4] Jonathan Lennox, Henning Schulzrinne and Thomas F. La Porta, "Implementing Intelligent Network Services with the Session Initiation Protocol", Tech-Report Number CUCS-002-99

[5] M. Handley (ACIRI), H. Schulzrinne (Columbia U), E. Schooler (Cal Tech), J. Rosenberg (Bell Labs), "SIP: Session Initiation Protocol", RFC 2543, Mar 1999

[6] ITU-T, Recommendation H.323 Version 5, Packet-based multimedia communications systems, 2003

[7] Henning Schulzrinne, “SIP for mobile applications”, Dept. computer science, Columbia University, VON developper’s summer 2000 (Boston), July 18,2000

[8] Henning g. Schulzrinne and Jonathan D. Roseberg, “The Session Initiation Protocol: Providing Advanced Telephony Services Across the Internet”, Bell Labs Technical Journal, Oct-Dec 1998

[9] Werner Van Leekwijck, Dirk Brouns, “Siplets: Java-based Service Programming for IP telephony”, Alcatel Telecommunication Review-2nd Quarter 2000

[10] Haojin Wang, “Telecommunications Network Management”, ISBN: 0070681708, Computing McGraw-Hill,1999

[11] E. J. Cameron, N. D. Griffeth, Y.-J. Lin, M. E. Nilson, and et al, “A Feature Interaction Benchmark for IN and Beyond”, Feature Interactions in Telecommunications Systems, IOS Press, pp. 1--23, 1994

[12] FIW’92, St. Petersburg, FL, USA; FIW’94 Amsterdam, The Netherlands; FIW’95, Kyoto Japan; FIW’97, Montreal, Canada

[13] Muffy Calder, Evan Magill, editors, “Feature Interactions in Telecommunications and Software Systems VI”, Glasgow, Scotland, IOS Press (Amsterdam), 2000.

[14] Universal Personal Telecommunication (UPT), ”Requirements on feature interaction and network functionality”, ETR 064, reference DTR/NA-70304

[15] T.F. Bowen et al., "The Feature Interaction Problem in Telecommunication Systems." Proc. Seventh Int'l Conf. Software Eng. for Telecommunications Switching Systems, Institution of Electrical Engineers, London, 1989, pp. 59-62

[16] Dirk O. Keck, Paul J. Kuehn, “The Feature and Service Interaction Problem in Telecommunications Systems: A Survey”, IEEE Transactions on Software Engineering, Vol. 24, No. 10, October 1998

[17] M. Calder and E. Magill, editors, *“Feature Interaction in Telecommunications and Software Systems VI”*, IOS Press, Glasgow, May 2000

[18] F.S. Dworack, “Approaches to Detecting and Resolving Feature Interactions”, Proceeding GLOBECOM 1991, Phoenix, Arizona, pp. 1371-1377, Dec.1991

[19] Y. Peng, F. Khendek, P. Grogono and G. Butler, Feature Interaction Detection Technique Based on Feature Assumption, Fifth International Workshop on Feature Interactions in Telecommunications and Software Systems, FIW'1998, Sweden, 1998

[20] A.Sefidcon and F. Khendek, "A pragmatic Approach for Feature Interaction Detection in Intelligent Networks", Proceedings of IEEE conference on Computer Communication and Networks, Oct. 1999

[21] Alessandro De Marco, Ferhat Khendek, "eSERL: Feature Interaction Management in Parlay/OSA Using Composition Constraints and Configuration Rules", Proc. of 8th International Conference on Feature Interactions, FIW'2003, Ottawa, Canada, 2003

[22] Jonathan Lennox and Henning Schulzrinne, "Feature Interaction in Internet Telephony", Columbia University, USA, URL: http://www.cs.columbia.edu/~lennox/Lenn0005_Feature.pdf, 2000

[23] J. Lennox and H. Schulzrinne, "Feature Interaction in Internet Telephony", Proc. of Sixth Int'l. Workshop on Feature Interactions in Telecommunication Networks and Distributed Systems (FIW'00), pp.38-50, May. 2000

[24] H. Schulzrinne and J. Rosenberg, “Internet telephony: Architecture and protocols,” *Computer Networks and ISDN Systems*, 1998

[25] R. Manione, M. Festa, P. Renditore, D. Sereno, M. Spaziani, “Service Issues in Web Call Centers”, Telecom 99 Forum, Geneva, 1999

[26] C. Gbaguidi, et al, Integration of Internet and telecommunications: “An Architecture for Hybrid Services”. IEEE JSAC vol.17, no.9, Sept. 1999

[27] Ajay P. Deo, Kelvin R. Porter and Mark X. Johnson, “Java SIP Servlet API Specification”, MCI Worldcom, April 27, 2000

[28] Roch Glitho, Riad Hamadi and Robert huie, “An Architectural Framework for Using Java Servlets in a SIP Environment”, Proceedings of ICN (2) 2001, 707-716, Colmar, France, July 9-13, 2001

[29] M. Handley, H. Schulzrinne, E. Schooler, J. Rosenberg, “SIP: Session Initiation Protocol”, RFC 3261, IETF, March, 1999

[30] M. Handley, V. Jacobson, “SDP: Session Description Protocol”, RFC IETF, April, 1998

[31] J. Lennox and H. Schulzrinne, "Call processing language framework and requirements," Request for Comments 2824, Internet Engineering Task Force, May 2000

[32] Java Community process, "SIP Servlet API", Java Specification IETF Internet Draft (work in progress)

[33] Lennox/Schulzrinne, "CPL: A language for user control of Internet Telephony Services" IETF Internet Draft, January 15, 2002 (work in progress)

[34] Schulzrinne and Rosenberg, "SIP Caller Preferences and Callee Capabilities IETF Internet Draft", IETF Interenet draft, Nov 24, 2000 (work in progress)

[35] Alan Johnston, Steve donovan, Robert Sparks, "SIP Telephony Call Flow Example", IETF Internet Draft, November 2000 (work in progress)

[36] A. B. Roach (Dynamic soft), Session Initiation Protocol (SIP) – Specific Event Notification, RFC2543

[37] Paul M. Tyma, Gabriel Torok, and Troy Downing, “Java Primer Plus, supercharging web applications with the Java programming language”, Waite Group Press, 1996

[38] K.Kimbler and H. Velthuijsen, ”Feature Interaction Benchmark”, Proc. of the third Feature Interaction Workshop (FIW95), Japan, Oct. 1995.

[39] JAIN ™ SIP specification 1.1 release, 2003

[40] Mick O'Doherty, “SIP servlet delivery”, IETF internet draft, July 2000

[41] Core JAVA, J2SE 1.3.1, Sun Microsystem, 2003

[42] Java Media Framework API (JMF) 2.1.1, Sun Microsystem, 2003

[43] Apache Jakarta Tomcat, Jakarta Project, The Apache Software Foundation (ASF)