Detection and Resolution of Feature Interactions in IP Multimedia Subsystem

Journal:	*International Journal of Network Management*
Manuscript ID:	NEM-07-0055.R2
Wiley - Manuscript type:	Research Article
Date Submitted by the Author:	n/a
Complete List of Authors:	Gouya, Anahita; Institut National des Télécommunications (INT), Wireless Networks and Multimedia Services Crespi, Noel; Institut National des Télécommunications (INT), Wireless Networks and Multimedia Services
Keywords:	Feature Interaction, IMS, SIP, Telecommunication Services

Detection and Resolution of Feature Interactions in IP Multimedia Subsystem

Anahita Gouya
PhD Student
GET/INT- Institut National des Télécommunications
Mobile Networks & Multimedia Services Department
9, Rue Charles Fourier, 91011, Evry, France
anahita.gouya@int-edu.eu

Noël Crespi
Professor
GET/INT- Institut National des Télécommunications
Mobile Networks & Multimedia Services Department
9, Rue Charles Fourier, 91011, Evry, France
noel.crespi@int-edu.eu

ABSTRACT

The IP Multimedia Subsystem (IMS) enables the delivery of multimedia services through different access networks. 3GPP adopted Session Initiation Protocol (SIP) as the IMS signalling protocol. However, IMS interaction with these services faces the "*Feature Interaction*" problem encountered in traditional telephony networks. This problem occurs as features invoked during a session behave correctly when processed separately from each other, but not when running together. Although much research work has explored managing feature interactions, applying these solutions in IMS remains challenging and further investigation is needed. This paper aims to analyze the feature interaction issue in IMS and to propose a new solution. We defined a SIP-based algorithm and the associated mechanisms to enhance IMS service invocation. This algorithm is the core of the Service Broker, a new functional entity in charge of managing feature interaction. We validated our proposal through a performance evaluation and prototyped the Service Broker on an open source IMS platform.

Keywords

Feature Interaction, IMS, SIP, Telecommunication Services.

1. INTRODUCTION

The evolution of fixed and mobile core networks towards an all-IP architecture, particularly by the introduction of the IMS [1], is an important driver for achieving service merging and network convergence. Based on SIP [2], IMS supports the interaction with a variety of multimedia services. However, even with the additional call control approaches offered by IMS, services may behave unexpectedly when running together. This problem, known as feature interaction in the traditional telephony networks, has been widely studied within academic and industrial research. The term *feature* was initially used in Intelligent Network (IN) services. Bellcore defined feature as a unit of one or more telecommunications or telecommunication management based capabilities, which a network provides to a user [3]. ITU-T defined a feature as the smallest part of service that can be perceived by a user [4]. Alongside these definitions, we will use in this paper, the word feature as an independent technical

function that the network provides to subscribers in order to personalize the services residing in the operator network or in a third party location.

The well-known call control features are *Call Barring*, which restricts all outgoing calls to the network-defined forbidden destinations, *Call Forwarding Unconditional*, which allows a subscriber to have the network send all incoming calls addressed to the called subscriber's directory number, to another directory number, *Calling Line Identification Restriction*, which masks the caller identity, *Operator Service*, which enables the caller to establish a call, through an operator, *Originating Call Screening*, which blocks outgoing calls to the destinations that are on the blacklist and *Terminating Call Screening*, which blocks calls coming from the users that are in the blacklist. These call control features can be used by several Next Generation Network (NGN) services. For instance, an operator can provision a Call Barring with a Push-to-Talk service to handle call restrictions. An interactive gaming service triggers the activation of the Call Forwarding Unconditional feature in order to control the call diversions.

If features invoked during a session behave correctly when processed separately, but not when running together, feature Interaction occurs. We define two types of feature interactions: *Intra Domain Feature Interactions*, that take place between the features of caller *or* callee, and *Inter Domain Feature Interactions*, that happen between the features of caller *and* callee. For example, in the following scenario, an intra domain feature interaction occurs between the features of the caller: Assume that Alice calls Bob to join multiparty gaming. She has a Call Barring feature that restricts her outgoing calls to Bob. If Alice uses an Operator Service feature to call Bob indirectly, then the Call Barring feature will not work as supposed to, and although Alice's calls to Bob are forbidden, she establishes a call with him indirectly. As an example of inter domain feature interaction, suppose that Alice wants to establish a video call with Bob: Alice has an Originating Call Screening feature that blocks all the outgoing calls to the end users who are on the black list. On the other hand, Bob has a Call Forwarding Unconditional feature that diverts all the incoming calls to Anne. What happens if Anne is on the blacklist of Alice? In this case, if the Originating Call Screening feature works correctly as defined for Alice, then the Call Forwarding Unconditional feature will be neglected. Otherwise, if Call Forwarding Unconditional feature behaves as it is defined for Bob, then Alice's Originating Call Screening feature will be ignored.

Managing feature interactions consists of *preventing* whenever possible or *detecting and resolving* the interactions that may occur between different features. Feature interaction prevention is achieved by enforcing necessary constraints on the service platform and end users. These constraints mainly deal with odd feature configurations, inaccuracy in feature definitions, bad resource allocations, software application programming errors and the lack of interoperability between heterogeneous telecommunication systems. However, even if the service platform and end users respect these constraints, the feature interactions are likely to occur. In the literature, two methods are considered for detecting and resolving feature interactions: *offline* and *online*. A number of feature interactions can be detected and resolved before features are invoked. This management mechanism is called offline. However, due to the unpredictable behavior of features as well as the vast introduction of new features, all feature interactions can not be detected and resolved offline. In other words, some interactions occur during feature run-time and consequently must be detected and resolved at run-time, that is, online. Furthermore, we refer to a feature interaction detection and resolution mechanism as *static*, if the management of feature interactions is based on the predictable and predefined feature interaction detection and resolution information. On the contrary, the management mechanisms that are based on real time feature information are called *dynamic*.

Initial research work on the feature interaction problem started with IN concept where undesired interactions were encountered between the features that were invoked during a call. While the advent of IMS provides new approaches for the improvement of multimedia service delivery, feature interaction management still remains a challenging issue. IP Telephony is likewise inadequate, since it introduces new feature interactions and also new approaches for dealing with them. In fact, the main contributors of IP Telephony believe that: “the architecture of Internet Telephony is sufficiently different from traditional telephony to make it necessary to revisit the feature interaction issue in this context [5]”.

The main contribution of this paper is to investigate the feature interaction problem in IMS by discussing the corresponding requirements. The lessons learned from previous feature interaction management solutions, help us provide IMS with appropriate means to deal with this issue. In the following section, we present a brief overview of SIP and IMS functional architecture, then we review the research work dealing with the feature interaction issue. Afterwards, we expose the requirements for the introduction of a feature interaction management mechanism in IMS. Based on the results of related work and the discussed considerations, we propose a feature interaction management algorithm that fits IMS requirements. The application of this algorithm is studied through several feature interaction scenarios. Finally, we analyze and evaluate the impact of our solution through analytical and experimental results.

2. SIP: TOWARD A UNIFIED SIGNALING SOLUTION

SIP is the *de facto* signaling protocol specified by IETF for enabling the establishment, modification and termination of multimedia sessions in IP networks. SIP messages are in the form of Request and Response, exchanged between User Agents and Servers. Figure 1 illustrates the flow of SIP messages required for a simple call set-up, in which Alice invites Bob to a communication by sending to him an “INVITE” request. “100 Trying” is a provisional response, indicating that the Server has received the INVITE request and continues to process it. The “180 Ringing” response states that Bob has received the invitation and is alerted to answer the incoming call. When Bob picks up the phone, he sends a “200 OK” response to Alice to indicate that the call has been answered. Alice confirms the session establishment by sending back an acknowledgment (ACK) request. Then, the end-to-end media packets will be exchanged between the end users.

In fact, the text-based nature of SIP offers the application layer an expressive and extensible signaling protocol, adaptable to the needs of IP telephony services. Moreover, SIP supports other enriched and customized services such as Presence and Instant Messaging. Today, multiple Working Groups in IETF continue working on this domain to improve the shortcomings of SIP and to deal with remaining Voice over IP issues [6]. Moreover, several Voice over IP service providers [7] carry on interesting research and development efforts in order to propose innovative and value-added SIP-based telephony services.

3GPP specified IMS as a standard NGN service control overlay to provide mobile and fixed multimedia services over heterogeneous networks. The functional architecture of IMS is shown in Figure 2. The following SIP Servers, called Call Session Control Function (CSCF), are included in the session control layer of IMS: *P-CSCF* (Proxy-CSCF), which is the first contact point of IMS for the user, *I-CSCF* (Interrogating-CSCF), which is the entry point within the IMS operator’s network for other operators and *S-CSCF* (Serving-CSCF), which is the brain of IMS for session controlling and service invocation.

The IMS subscriber’s profile is stored in central databases called Home Subscriber Server (HSS). During user registration, S-CSCF retrieves the user profile from HSS and stores it for further session

establishment procedures. This service profile contains a list of initial Filter Criteria (iFC) consisting in a pair of a Boolean expression called Trigger Point, and an Application Server address associated with this Trigger Point. The conditions in a Trigger Point may be applied to the SIP method, the SIP headers and their content, originating or terminating session cases and to the session description in the body of the SIP message. If a Trigger Point is met, S-CSCF sends the request to the associated Application Server.

The functional architecture of IMS is available for various services and IMS supports the access to them through a SIP-based interface called IP multimedia Service Control (ISC). 3GPP defines three types of application servers: SIP-based application server, Open Service Access (OSA) application server (through a Parlay/OSA gateway called OSA Service Capability Server) and Customized Applications for Mobile Network Enhanced Logic (CAMEL) server (through an inter-working entity called IP Multimedia Service Switching Function). Even if innovative approaches are introduced by IMS, managing feature interactions remains a critical issue and further exploration is required. In the next section, we review the research work in this field. Then, based on the advantages and shortcomings of these works, we discuss the requirements of a feature interaction management mechanism in IMS.

3. MANAGING FEATURE INTERACTION: A SURVEY

Surveys in [8] and [9] give a comprehensive overview of the work carried out in this field during 90s. Cameron et al. [10] classify the causes of feature interactions in IN and present two general approaches to manage the feature interactions: 1) avoiding the interaction by improving the service architecture and 2) providing the service architecture with specific feature interaction detection and resolution methods.

3.1 Feature Interaction Avoidance: Negotiation

Kolberg and Magill [11] propose a feature interaction avoidance mechanism in which end users negotiate their available services before the SIP session is established. In order to achieve this negotiation, the caller extends the INVITE request, addressed to the callee, by adding to the request a list of its services as well as the possible actions that the callee could perform if it does not agree with the indicated service list. Afterwards, the callee may accept the call or it may select one of the offered actions. It is also possible that the callee refines some of the proposed actions and suggests new ones to the caller. Then, the caller will decide which of the options is preferable. Another negotiation approach for managing feature interactions in IN is described by Griffeth and Velthuijsen in [12]. This proposition is based on the negotiation between automated agents that contain the profile of end users and decide the eventual occurrence of interaction between these profiles, before the call is established.

The main shortcoming of these propositions is the lack of transparency for the end parties, while the end parties must be engaged in the feature interaction management procedure. Moreover, the end parties should trust each other during the negotiation; otherwise these mechanisms are vulnerable to various attacks such as identity usurpation or denial of service. Finally, these propositions require the end parties to agree on a service negotiation template in order to understand the used negotiation vocabulary.

3.2 Feature Interaction Avoidance: Feature Description

Harada et al [13] propose a method based on predefined feature interaction tables that contain a description of features. These feature specifications are bound to terminals. Feature interactions may therefore occur due to combinations of two feature specifications. The combinations are stored in tables

that use state transaction diagrams for describing the potential feature interactions as well as the avoidance mechanism. Even if the experimental implementation and evaluation of this proposition confirm its effectiveness in terms of interaction avoidance process time, this proposition is not flexible when new features are introduced as the feature interaction tables should be regularly updated.

3.3 Feature Interaction Avoidance: Formal Models

Ohta and Harada [14] propose a mechanism which uses finite state machines to describe a system's behavior. In this proposition, the network is considered as a black box and service descriptions express the desired behavior of terminals. Furthermore, feature interactions are modeled as unwanted properties of these descriptions. Chan and Bochmann [15] provide a formal model of SIP features and feature interactions, using Specification and Design Language which covers some of the characteristics of the SIP message, such as caller-id, addressing header fields and sequence number. In this proposition, the following feature interaction prevention strategies are discussed: preventing resource contention and limitation, incoherent interactions, unexpected non-determinism and unfair interactions, deadlocking interactions (due to the mutual deployment of features). By means of the proposed feature interaction classifications the number of potential feature interactions is supposed to decrease. However, these propositions face limitations regarding the quantification of feature interaction instances and their behaviors. Furthermore, the proposed models must be regularly updated in order to manage the interactions that may occur once new features are introduced.

3.4 Feature Interaction Detection and Resolution: Offline

Chentouf et al. [16] propose a feature interaction detection and resolution method by integrating a Feature Interaction Manager (FIM) between the end users and the SIP proxies. In this proposition, users inform the FIM about their intention to run a feature. The FIM launches the detection procedure to check if this feature will interact with those already running on the same session. If an interaction is detected, the FIM instructs the SIP proxy to stop the call and launches the resolution procedure. Khoumsi [17] presents a formal model based on finite state machines for specifying the features and a methodology to detect and resolve the feature interactions based on predefined interaction cases. In this proposition, feature interactions are defined as undesirable states of finite state machines. As for the resolution method, the undesirable states are forced to behave in an acceptable way. Jouve et al. [18] present the features in the form of operational specifications and define a method to compute feature triggering and to detect the interactions. Furthermore, this work proposes to adapt these feature specifications and interaction detection methods to SIP-based services. Finally Kolberg and Magill propose in [19] an offline approach for describing features and defining the rules to detect the interaction scenarios. They follow up this approach in [20] and propose a distributed feature interaction detection and resolution mechanism that is applicable on SIP-based services. In this proposition, the behavior of services is described via an extended SIP header. The triggered service includes this SIP header in the SIP message. Afterwards, the next invoked service compares the pairs of invoked services to detect the interactions. Once an interaction is detected, the resolution algorithm decides about disabling one service in order to resolve the occurred interaction. The main drawback of these propositions is that they are limited to offline feature interactions and the conflicts occurring at service run-time could not be managed.

To the best of our knowledge, not enough research on IMS has been conducted so far. However we believe that IMS, as a leading all-IP service control plane, needs further consideration in this domain.

Based on the lessons learned from previous research work, we define the following requirements for introducing a feature interaction management mechanism in IMS: 1) Adaptability to IMS via SIP-based mechanisms, 2) Transparency to the users without requiring their engagement for feature interaction management, 3) Efficiency without considerable impact on the session establishment time and 4) Extensibility to answer to the needs of any new feature. The principles for introducing a feature interaction management approach that fulfils these requirements are discussed in the next section.

4. PRINCIPLES FOR MANAGING FEATURE INTERACTION IN IMS

We define the following principles for providing IMS with feature interaction management mechanism:

- Introducing a functional entity that orchestrates successive feature invocations in one session and is responsible for detecting and resolving feature interactions. We defined this entity as a Service Broker that interacts between an S-CSCF and the application server(s).

- Defining a SIP-based feature interaction detection and resolution algorithm over Service Broker that covers the defined requirements.

A standardization effort is ongoing at 3GPP [21] for defining the architecture of Service Broker and the associated procedures. The discussions are still at a very high level and our solution may be a candidate to accomplish the above-mentioned principles. As shown in Figure 3, four architectural models are possible for introducing a Service Broker between the S-CSCF and the application server(s).

In Figure 3.a, the Service Broker is defined in the service layer, as a stand-alone application server that interacts between S-CSCF and application servers. The main weakness of this model is lack of interoperability support between application servers. In fact, the introduction of an intermediate application server in the service invocation path requires additional security and trust control functionalities, in order to ensure the interaction between third party application servers. In Figure 3.b, the Service Broker is defined as an additional functionality over each application server. From the architectural point of view, this model is not optimal as the autonomy of services is not ensured. Besides, by assigning the feature interaction management to services, once a new service is introduced, the feature interaction management functionality (over each application server) should be updated, to answer to the needs of the new service. Another drawback of the models proposed in Figures 3.a and 3.b, is that services need to be engaged in the service conflict management procedure. Hence, the NGN approach defined by ITU-T, in which services and control functionalities are entirely separated, will not be respected.

In Figure 3.c, the session control layer of IMS is provided with a Service Broker. This model is based on a central feature interaction manager, handling the potential interactions that may occur between features invoked during a session. The inadequacy of this model is due to the fact that the improvements in both service and control layer will raise problems related to the scalability of the network, flexibility of new services and bottlenecks that may be created over the central manager points. To prevent these problems, in Figure 3.d, we propose a distributed feature interaction management approach in which the functionality of a Service Broker is introduced over the S-CSCF. Therefore, instead of introducing a Service Broker as the center of feature interaction management decisions, we delegate the tasks to the S-CSCF that is provided with a Service Broker functionality. In the next section, we present our proposed feature interaction detection and resolution algorithm in detail.

5. A SIP-BASED ALGORITHM FOR MANAGING FEATURE INTERACTIONS

Along with the mentioned architectural and session routing requirements and following our proposed feature interaction management approach in [22], in this section we present an algorithm that provides feature interaction detection and resolution to the Service Broker. We should recall that this proposition is a possible functionality for a Service Broker and further functionalities may be considered in future. This algorithm is composed of *Service Identifier Comparison* and *Service Rule Comparison* modules.

5.1 The Service Identifier Comparison Module

Before describing this module, we present the new terms included in this module:

- *Service identifier* is a unique identity that we propose to be associated to each of the widely deployed communication services and features. This proposition is also discussed in a former IETF SIPPING Working Group draft [23], where, the needs for defining the Service Identification were studied. The use of the service identifier in our proposition emphasizes the advantages that this unique namespace will bring for the communication services.

- *Service-ID header*. We extend SIP by introducing a SIP header called "Service-ID", which must be added by the invoked application server to the SIP message, containing its service identifier. Hence, once an application server is invoked, we force this latter to include a Service-ID header in the SIP answer that it sends back to S-CSCF.

- *Service conflict database*. We provide the Service Broker with a service conflict database containing a list of known feature interactions and the related resolution method.

Before invoking an application server, the Service Identifier Comparison module acts as following: all through the session establishment procedure, this module compares the identifier of the already invoked application server(s) (included in the Service-ID header) with the identifier of the application server that is going be invoked. In other words, by referring to the service conflict database, this module verifies if the application server to be invoked is not in conflict with the already invoked application server(s). In case of interaction, this module behaves as defined by the resolution method of the service conflict database. The function of this module is illustrated in Figure 4.

The feature interaction detection and resolution mechanism defined in this module is static, since it is based on predefined service conflict management information. Furthermore, this mechanism is offline i.e. the interactions are detected and resolved before service execution. In order to provide the Service Broker with means for handling interactions which occur at service run-time, in the next section, we propose an online and dynamic feature interaction management module that recovers the interactions that were not handled by the Service Identifier Comparison module.

5.2 The Service Rule Comparison Module

The Service-Rule is a SIP header extension proposed in [24], enabling the invoked application servers to indicate which modifications on the SIP message, by the services to be invoked, are not accepted for this application server. The proposed syntax for the Service-Rule header is as follows:

Service-Rule: [Applicability]; [messagePart]; [forbiddenValues]

This header indicates which values (i.e. *forbiddenValues*) are not accepted over which elements (i.e. *messagePart*) of which SIP message (i.e. *Applicability*).

In our proposition, for each added Service-Rule, the Service Rule Comparison module verifies two aspects: 1) that this rule is compatible with the rules initially defined by the network (called Unauthorized Rules) in order to prevent the application server from defining abusive rules, 2) that this rule is compatible with the rules defined by the previously invoked application server(s).

If the defined Service-Rule is not compatible with the Unauthorized Rules or the rules defined previously invoked application severs, the session establishment process will be interrupted and this module sends an error message to the user. This module is illustrated in Figure 5.

5.3 The Feature Interaction Management Algorithm

The interoperation between the Service Identifier and Service Rule Comparison modules for fulfilling the needs of our proposed feature interaction management algorithm is presented in Figure 6.

As presented in this algorithm, once an iFC matches, the Service Broker performs offline feature interaction management by invoking the Service Identifier Comparison module. Based on the decision performed by the Service Identifier Comparison module, if the Service Broker recognizes that invoking this application server will result in a feature interaction, the module behaves as defined by the resolution method of the service conflict database and sends the answer back to S-CSCF. Otherwise, the Service Broker accepts the application server triggering request and hence, in this case the SIP request will be forwarded to the application server. This latter provides services and may modify the request. Furthermore, the application server will provide the SIP message with service information (i.e. the Service-ID and Service-Rule headers) that enables the Service Broker to perform online feature interaction management by invoking the Service Rule Comparison module. This module controls the compatibility of the service rule(s) with the Unauthorized Rules and with the previously defined rules. In the case of incompatibility, the session establishment process will be interrupted and this module sends an error message to the user. Otherwise, the session establishment procedure will continue.

This feature interaction detection and resolution approach brings interesting advantages to IMS:

- First, since the Service Broker represents a functional entity integrated in the S-CSCF additional routing considerations (between S-CSCF, Service Broker and Application Server) will be handled by S-CSCF, and no further session routing consideration is required.

- Another advantage brought by this proposition is that it provides a distributed feature interaction management mechanism in which, instead of having a central feature interaction manager, a Service Broker is assigned to each S-CSCF. In fact, along with IMS specifications, wherein once a user registers to IMS, an S-CSCF will be associated to it that is responsible for session establishment and service invocation, in our proposed mechanism, the IMS network operator assigns, to each S-CSCF, a Service Broker to manage the feature interactions for the user who is served by this S-CSCF.

- Furthermore, the SIP-based nature of our proposition enables the defined modules and functionalities to be extended so that they answer to the new needs and requirements of future features.

- Finally, this proposition remains transparent to the users and does not require the user engagement for feature interaction management.

In a nutshell, this algorithm starts the detection and resolution of a number of feature interactions, offline, i.e. before the application server is invoked. Then, other feature interactions will be detected and resolved online, i.e. at service run-time. Furthermore, in this algorithm, a number of interactions will be managed statically (based on predefined service conflict database and the Unauthorized Rules list). Others will be managed dynamically and based on real-time service information (Service-Rule).

6. SCENARIOS

In this section we present the application of our proposed feature interaction management algorithm when intra domain and inter domain feature interactions occur.

6.1 Intra Domain Feature Interaction

> *Alice is not allowed to call Bob. This restriction is controlled by Call Barring service. However, Alice establishes a call to Bob by the intermediate of an Operator Service.*

This scenario, illustrated in Figure 7, presents an intra domain interaction occurs between Call Barring and the Operator services invoked on Alice's side. The iFC defined in the service profile of Alice is as shown below indicating that on the reception of INVITE request, Call Barring service must be invoked:

```
<InitialFilterCriteria>
        <Priority>0</Priority>
        <TriggerPoint>
                <ServicePointTrigger>
                        <Method>INVITE</Method>
                </ServicePointTrigger>
        </TriggerPoint>
        <ApplicationServer>
                <ServerName>Call_Barring@int-edu.eu</ServerName>
        </ApplicationServer>
</InitialFilterCriteria>
```

Once the INVITE request arrives at S-CSCF, it evaluates the iFC to find out which application server should be invoked. Based on the service profile of the caller, the reception of the INVITE request necessitates the invocation of the Call Barring service. Before invoking the Call Barring service, the Service Broker executes the Service Identifier Comparison module to verify that this service is compatible with the previously invoked services (Service-ID headers in the INVITE request indicate which services have been invoked earlier). In this example, no service is invoked before the Call Barring service, hence this control will be bypassed and the Service Identifier Comparison module accepts the Call Barring invocation request. Therefore the request will be forwarded to this service.

Based on the Call Barring service logic, if the call to the callee is forbidden, the Call Barring service sends back a SIP error answer (e.g.: 403 Forbidden). In this example, we suppose that the control is satisfied and that the Call Barring service adds its identifier to the Service-ID header and includes

eventual Service-Rule header(s). Then, it sends back the INVITE request to the Service Broker. Before continuing the iFC evaluation of the S-CSCF, the Service Broker runs the Service Rule Comparison module to verify if the Service-Rule introduced by the Call Barring service is compatible with Unauthorized Rules and with the previously defined rule(s). In the case of incompatibility, this module interrupts the session establishment process and sends an error message to the caller. We suppose that these verifications are performed successfully and the Service Broker sends the request back to S-CSCF.

Once these controls are performed and no interaction has occurred, the S-CSCF forwards the request to the Operator Service in order to call Bob on behalf of Alice. The Operator Service sends a "181 Call is forwarded" response and a new INVITE request with Bob as SIP URI to the S-CSCF of Alice. (Other behaviors are also possible for the Operator Service. For example it may send a new INVITE request directly to Bob. However, we suppose that the message sent from the Operator Service is forced to be forwarded to the S-CSCF of Alice before being routed to Bob's domain). On the reception of this answer, the Service Rule Comparison module verifies if the new INVITE request respects the previously defined rules. We suppose that the Call Barring service had defined a service rule for forbidding calls to Bob. This rule can be as formulated as:

Service-Rule: Applicability= INVITE; messagePart=requestURI; ForbiddenValues =Bob

Therefore, the Service Rule Comparison module recognizes that the new INVITE request is not compatible with the service rule defined by the Call Barring service. Thus, this module, rejects the INVITE request and the 181 response, and sends back a 403 forbidden response to Alice.

6.2 Inter Domain Feature Interaction

Alice is not allowed to call Eve and this limitation is controlled by the Call Barring service of Alice. Alice calls Bob. Bob has a Call Forwarding Unconditional service that forwards all incoming calls to Eve. Hence, Alice's call is unwillingly forwarded to Eve.

Figure 8 presents this scenario in which inter domain interaction occurs between the services invoked on Alice's and Bob's sides. The service profile of Alice contains the following iFC:

```
<InitialFilterCriteria>
        <Priority>0</Priority>
        <TriggerPoint>
                <ServicePointTrigger>
                        <Method>INVITE</Method>
                </ServicePointTrigger>
        </TriggerPoint>
        <ApplicationServer>
                <ServerName>Call_Barring@int-edu.eu</ServerName>
        </ApplicationServer>
</InitialFilterCriteria>
```

In this scenario, the interaction between the S-CSCF and the Call Barring service through the Service Broker is the same as the procedure explained in the previous scenario: iFC evaluation, service identifier

comparison, Call Barring control, service rule comparison and potential interaction detection and resolution. Once these controls are satisfied, the S-CSCF forwards the request to Bob's domain. Bob has a Call Forwarding Unconditional service indicated in the following iFC:

```
<InitialFilterCriteria>
        <Priority>0</Priority>
        <TriggerPoint>
                <ServicePointTrigger>
                        <Method>INVITE</Method>
                </ServicePointTrigger>
        </TriggerPoint>
        <ApplicationServer>
                <ServerName>Call_Forwarding_Unconditional@int-edu.eu</ServerName>
        </ApplicationServer>
</InitialFilterCriteria>
```

Hence, S-CSCF evaluates the iFC, and using the service profile of Bob, recognizes that the Call Forwarding Unconditionally service must be invoked. Before invoking this service, S-CSCF invokes the Service Identifier Comparison module to verify if the Call Forwarding Unconditional service is not in conflict with the already invoked services (that are included in the Service-ID header of the INVITE request). Once the Call Forwarding Unconditional service receives the request, it sends the S-CSCF a "181 Call is being forwarded" response for the previous INVITE request and a new INVITE request (same Call-ID but new SIP URI) to Eve. Then, the S-CSCF invokes the Service Rule Comparison module and recognizes that the "INVITE Eve" request is not compatible with the service rule previously defined by the Call Barring service that is:

Service-Rule: Applicability= INVITE; messagePart=requestURI, To; ForbiddenValues =Eve

Hence, this module rejects the new INVITE request and sends back a 403 forbidden response to Alice.

7. PERFORMANCE EVALUATION

The Service Broker functionality proposed in this paper tackles the feature interaction problem and provides IMS with the means to deal with this issue. This proposition has the advantage of being SIP-based and hence compatible with the service invocation mechanism of IMS. However, the cost introduced by the Service Broker, principally, at the session establishment time is a critical performance issue that must be investigated in order to evaluate the effect of adding the proposed feature interaction management solution in IMS. To achieve this goal, we will study in this section the additional delay in the session establishment that the introduction of the Service Broker induces. Table 1 presents the parameters that are applied for evaluating the impact of Service Broker on session establishment time.

Since the Service Broker is executed for each service invocation time, T_{caller} will be $n_{caller} \times t$ and T_{callee} will be $n_{callee} \times t$. Besides, based on the number of application server(s) invoked on the caller and callee sides, the session establishment time when S-CSCF is not provided with a Service Broker will be $(1 + n_{caller}) \times X_S + (1 + n_{callee}) \times X_S$. This expression indicates that firstly, each of the S-CSCFs of the caller and callee will proceed one time ($X_S + X_S$). Secondly, for each service invocation request S-CSCF will

take (n_{caller} x X_S + n_{callee} x X_S) time. By including the Service Broker in the session establishment path, the session establishment time will be: $(n_{caller} + 1)$ x X_S + $(n_{callee} + 1)$ x X_S + $(n_{caller} + n_{callee})$ x t.

In fact, the cost introduced by Service Broker particularly depends on how the Service Identifier and Service Rule Comparison modules are designed.

The process time at the Service Identifier Comparison module (T_{S_ID}) depends on two parameters:

- Message parsing time ($T_{parsing}$) for retrieving previous Service-ID headers (referring to the invoked services) and Route header (referring to the service to be invoked)
- Service conflict database lookup time (T_{db_lookup}) to search the pair of (invoked service, service to be invoked)

This process time is presented as following:

$$T_{S\_ID} = \begin{cases} 0 & \text{If } n_{caller} = n_{callee} = 0 \\ (n_{caller} + n_{callee} - 1) \times T_{parsing} + (n_{caller} + n_{callee} - 1) \times T_{db\_lookup} & \text{Otherwise} \end{cases}$$

We can prove this expression through mathematical induction by demonstrating that 1) the first statement in the statement sequences is true and 2) if any given statement in the statement sequences is true, then so will the next one. The proof of this expression is shown in the appendix.

The process time at the Service Rule Comparison module (T_{S_Rule}) depends on three parameters:

- Message parsing time ($T_{parsing}$) for retrieving Service-Rule headers (referring to the rules defined by invoked services)
- Unauthorized Rules database lookup time (T_{db_lookup}) to verify if the services respect the network defined constraints
- Previous rules list lookup time (T_{list_lookup}) to verify if the invoked service respect these rules

T_{S_Rule} may be computed as:

$$T_{S\_Rule} = \begin{cases} 0 & \text{If } n_{caller} = n_{callee} = 0 \\ (n_{caller} + n_{callee} - 1) \times T_{parsing} + (n_{caller} + n_{callee} - 1) \times T_{db\_lookup} + (n_{caller} + n_{callee} - 1) \times T_{list\_lookup} & \text{Otherwise} \end{cases}$$

This expression can be confirmed through the same mathematical induction as T_{S_ID}.

We showed earlier that the delay introduced by Service Broker on the session establishment time can be computed as $(n_{caller} + n_{callee}) \times t$. Then, we can affirm that the Service Broker cost, i.e. $T_{S\_ID} + T_{S\_Rule}$ equals $(n_{caller} + n_{callee}) \times t$. Therefore, T_{S_ID} and T_{S_Rule} are explicitly related to t. Furthermore, since T_{S_ID} and T_{S_Rule} depend on the number of invoked services at caller and callee sides, the session establishment time increases as both the number of invoked services and the Service Broker process time increase. The relation between number of invoked services, maximum process time on Service Broker (t) and the session establishment time is presented in Figure 9. Here, we suppose that X_S is 10 ms and session timeout is 500 ms. Concerning t, we considered this value equals 10, 30 and 50 ms respectively, while the number of invoked services at the caller and callee sides increases. Since we did not have access to the information related to the real IMS (or SIP-based) platforms, we considered heuristic values for t, X_S and session timeout that are often used in academic studies [25, 26, 27]. Nevertheless, such estimation will not affect the performance evaluation results, as the goal is to analyze the Service Broker cost whatever the session establishment time and process time on S-CSCF are. Besides, we have limited our test to a maximum 4 service invocations at each side for two reasons: firstly, in practice the number of invoked services during a session should be restricted; otherwise the session time exceeds the session timeout. Secondly, the first results show that the next service invocations will follow the same linear function ($a \times X + b$, where $a = X_S + t$, $X = n_{caller} + n_{callee}$, and $b = 2 \times X_S$) i.e.:

$$(n_{caller} + 1) \times X_S + (n_{callee} + 1) \times X_S + (n_{caller} + n_{callee}) \times t =$$

$$(n_{caller} + n_{callee} + 2) \times X_S + (n_{caller} + n_{callee}) \times t =$$

$$(n_{caller} + n_{callee}) \times (X_S + t) + 2 \times X_S$$

The session establishment increase in this Figure can be explained as follows: As the number of invoked services increases, more process time will be observed at the proposed feature interaction management mechanism on the Service Broker. The session establishment time may even reach the session timeout while the number of invoked services increases. Secondly the more time costing the Service Broker is, the higher the session establishment delay will be. Therefore, the session establishment performance degrades as the Service Broker process time increases. Although these results are estimated, they show that for a limited number of services invoked during a session (which is usually the case), when the Service Broker is well designed, the cost of introducing the Service Broker is reasonable. This design is directly related to the SIP message parsing and database lookup functions.

Parsing SIP messages requires important processing time. According to [28], which presents an exhaustive SIP performance analysis, 25% of processing time on SIP proxies is due to SIP message parsing. Besides, the studies performed in [29] on the performance of SIP message processing on SIP proxies by focusing on the impact of message parsing, point out the effects on the message parsing time while additional SIP headers are introduced (18 µs of parse time for standard formatted SIP messages against 20 µs of parse time when additional header fields are included). In order to recover the supplementary delay introduced in message parsing time by the Service Broker, we can provide the Service Identifier and Service Rule Comparison modules with cache databases maintaining the content of already parsed Service-ID and Service-Rule headers. These caches help to expedite the process time on the Service Broker and reduce the impact of feature interaction management on session establishment time. Furthermore, in order to prevent excessive SIP message size (due to the added Service-ID and

Service-Rule headers), the Service Broker should control that the additional headers do not result into exceeding the maximum SIP message size.

The database lookup time on the Service Identifier and Service Rule Comparison modules depends on the number of iterations performed between these modules on the Service Broker and the databases (Service Conflict database, Unauthorized Rules database and previous Service-Rule list). Besides, these iterations are repeated for each service invocation request. Using a linear search, the feature interaction will be detected at *O (n)* time, where *n* is the number of database elements. Via a binary search algorithm this lookup procedure will be reduced to *O (log n).*

8. PROOF OF CONCEPT PROTOYPE

We have implemented the proposed feature interaction management mechanism on a Service Broker in the INT IMS platform (I^2P) [30]. In this section after describing I^2P, we present the experimental results gained through this implementation.

8.1 INT IMS Platform (I^2P)

The INT IMS Platform (I^2P), conforming to IMS specifications, is deployed by the Core Network and Service Architectures team of INT as a service control overlay test bed. I^2P offers an open source platform that is compliant with new technologies and aims to validate and extend the existing IMS standard. This platform contains two IMS domains each consisting of one host server (Windows Server, VMware) that emulates the IMS components (x-CSCF and several SIP application servers). Each of these domains contains a DHCP server, in order to manage the inter-domain mobility and routing aspects. These domains contain a gateway to Internet access. Besides, a router is configured to enable the creation of multiple sub networks. An open domain is defined in this platform compounding one emulated HSS (MySQL database), one Subversion server, one public Internet access and one PSTN access. Routers, switches and Wi-Fi access points are also integrated in I^2P. Key IMS core components such as x-CSCF, HSS and application server included in this platform have the following characteristics:

- x-CSCF: Different functionalities of the x-CSCF are implemented on an open-source SIP proxy called SER (SIP Express Router) [31] that is a recognized SIP-based call session control reference implementation. SER is developed in C language by iptel [32] and enables parsing and routing of SIP messages. Registration, authentication and presence management are some examples of the session handling modules already defined on the SER. In order to adapt the SER to IMS environment certain extensions need to be added on the SER. In the current implementation of I^2P, the P-CSCF supports the Path header (that indicates the address of proxy to the S-CSCF) and the P-Visited-Network-ID header (that identifies the origin of a request).Roaming right verification and the S-CSCF selection based on user profile are provided on the I-CSCF. Retrieving service profile from the HSS, triggering iFC, service invocation control and the Service Route header support are included on the S-CSCF.

- HSS: A MYSQL 5.1 database is designed for organizing the HSS information that consists of the service profile of IMS users, their roaming rights and the S-CSCF associated with each user. This information can be created, stored, retrieved and modified from this database. Contrary to the IMS specifications where Diameter protocol [33] is used for handling the user data in the HSS, in I^2P, the interface between the HSS and other IMS components are implemented via a mysql request/response.

- Application server: Java based application servers are implemented on I^{2}P for enabling the IP telephony features such as Call Barring, Call Forwarding Unconditional, Terminating Call Screening and Calling Line Identification Restriction. These application servers follow the SIP protocol specification and based on their service logic, they can act as a SIP Proxy, Registrar or Back to Back User Agent. Furthermore, I^{2}P is provided with the IP multimedia Service Control (ISC) interface between application server and S-CSCF to handle the service invocations.

As a temporary solution, until getting a real IMS client, the VoIP SIP and video software eyeBeam is used as a SIP client in I^{2}P. Besides a SER server is configured to play the role of an IMS gateway, i.e. once the SIP client sends a request, the IMS client receives it, adds IMS-related headers and then forwards the request to the IMS network.

8.2 Service Broker on I^{2}P

The SIP server is composed of two main parts:

- A configuration script is defined to contain the routing logic for processing SIP messages by each of the x-CSCF components. The SER's configuration file used in I^{2}P for defining the S-CSCF routing logic contains the global configuration parameters, loaded external modules, module parameters and routing blocks that define the routing logic for SIP messages.

- A directory is provided to contain various functions for running the server, parsing the SIP messages and allowing users to process the messages.

The first step for implementing the Service Broker in I^{2}P was to include the Service-ID and Service-Rule headers, into the platform in order to enable the SIP message parsing. This procedure was performed as follows: defining parsing macros for these headers, defining the headers in the header field name parser file and the SIP message parser files of the SER. Once the headers are defined, we included the Service Identifier and Service Rule Comparison modules to the S-CSCF message handling file.

In order to implement the Service Identifier Comparison module, we introduced the successive invocation of the Calling Line Identification Restriction and Terminating Call Screening services as a predefined conflict case in the Service Conflict database. Hence, once this module recognizes that an INVITE request contains the "*Service-ID: Calling Line Identification Restriction*" header, and the content of the Route header is pointed to the Terminating Call Screening service, the module rejects the INVITE request and sends a "403 Forbidden" response back to the S-CSCF.

In order to implement the Service Rule Comparison module, we first defined the following Unauthorized Rule on the Service Broker, indicating that calls from anonymous users are forbidden:

Service-Rule: Applicability = INVITE; messagePart = requestURI; ForbiddenValues = anonymous

Once a new Service-Rule header is added to the SIP message, this module compares this Service-Rule with the defined Unauthorized Rule. If these rules are identical, the module rejects the SIP. Otherwise, the Service Rule Comparison module compares the potential new Service-Rule header(s) defined in the incoming message with previously defined Service-Rule header(s). In the case of incompatibility, the module rejects the SIP message and sends the "403 Forbidden" response back to the S-CSCF.

We tested the efficiency of our feature interaction management algorithm through the following use cases. In the first use case, we suppose a session establishment process between Alice and Bob. In this use case, the service conflict scenario is as follows:

> *Alice has a Calling Line Identification Restriction service for masking her identity.*
> *Bob has a Terminating Call Screening service that restricts calls from Alice.*

Based on the service profiles of Alice and Bob (stored in I^2P HSS), once Alice calls Bob, the Calling Line Identification Restriction service of Alice masks the identity of Alice and forwards an anonymous call to Bob's side who receives an anonymous call from Alice. In other words, even if the Terminating Call Screening service of Bob must restrict the calls from Alice, the conflict between the Calling Line Identification Restriction and Terminating Call Screening services prevents the correct behavior of the Terminating Call Screening service. Our proposed Service Broker deals with this issue as follows: Before invoking the Terminating Call Screening service the Service Identifier Comparison module detects that the "Calling Line Identification Restriction, Terminating Call Screening" pair exists in the service conflict database indicating that these services are in conflict. Hence, the session establishment procedure will not proceed, and Alice will receive a 403 Forbidden response.

The second use case demonstrates how the Service Rule Comparison module of the Service Broker will deal with the feature interactions that can not be detected and resolved by the Service Identity Comparison module. The feature interaction scenario is as follows:

> *Bob has a Call Forwarding Unconditional service that forwards all the incoming calls to Eve. Furthermore Alice has a Call Barring service restricting calls to Eve.*

Hence, if Alice calls Bob, the Call Barring service will verify that Alice is not calling Eve, and permits the call to be sent to Bob. However, the Call Forwarding Unconditionally service of Bob will forward the call to Eve. In other words, Alice establishes a call with Eve, even if such a call should have been barred by the Call Barring service. To deal with this issue, based on the proposed feature interaction management mechanism, once the Call Barring service of Alice is invoked, it adds the following Service-Rule header to the INVITE request, indicating that calls to Eve are forbidden:

Service-Rule: Applicability= INVITE; messagePart=requestURI;ForbiddenValues =Eve

Then, the INVITE request arrives at Bob's S-CSCF and the Call Forwarding Unconditional service is invoked. This service should forward the call to Eve, but before this call forwarding is accomplished, the Service Rule Comparison module detects that the new INVITE request sent from the Call Forwarding Unconditional service (with "Eve" in the request URI) is not compatible with the service rule previously defined by the Call Barring service. Therefore, the conflict between the Call Barring and Call Forwarding Unconditional services is detected and the session establishment procedure will not proceed.

9. CONCLUSION

The main goals of this work are to point out the need for revising the feature interaction management problem in IMS and to propose a solution to deal with this issue. After reviewing the current characteristics of IMS, particularly at the service invocation level, we showed the IMS inadequacies for managing the feature interactions. Subsequently, and based on an inclusive survey that we have made on

the related research works, we defined the requirements and principles for introducing a feature interaction management mechanism in IMS. Then, we proposed the architectural and functional improvements in the service invocation mechanism of IMS in order to fulfill the defined requirements and principals.

We presented the Service Broker as a functional entity between the application server in the service layer and the S-CSCF in the session control layer of IMS to perform feature interaction management during the session establishment procedure. Assigning this functionality to the Service Broker (and not the S-CSCF or to application servers), removes the feature interaction management tasks from the service and the session control layers and enables the independent extensions of each layer. We should recall that this architectural choice remains valid for the operators that prefer to implement the Service Broker as a separate physical entity and not a functional one over S-CSCF. We provided the Service Broker with a feature interaction detection and resolution algorithm that manages a variety of feature interactions offline (before service invocation), and the rest will be detected and resolved online (at service run-time). This proposition has the advantage of being SIP-based and hence compatible and easily adaptable to the SIP-based service invocation mechanism of IMS. Finally, we showed the application of our proposed approach through various feature interaction scenarios and studied, through analytical and experimental results, the impact of the Service Broker on the session establishment time on IMS.

In our algorithm, we proposed two extended SIP headers, one for identifying the invoked services (Service-ID header) and the other for enabling services to indicate the undesired behaviors that are not accepted from the next services that may be invoked (Service-Rule header). Further effort is needed to improve and develop these SIP header extensions, and to express more precisely the service behavior. Furthermore, inter-operator agreement considerations and privacy control should be included in our proposition for achieving the proposed service information exchange approach between the end parties. These topics are considered as perspectives to our work. We believe that these efforts will bring fruitful insights to the ongoing researches in this field, mainly at 3GPP and IETF.

REFERENCES

[1] *IP Multimedia Subsystem (IMS)*, 3GPP TS 23.228.

[2] Rosenberg, J. et al., *SIP: Session Initiation protocol*, IETF RFC 3261, June 2002.

[3] *Advanced Intelligent Network (AIN) Release 1, Switching Systems Generic Requirements*, Bellcore Technical Advisory TA-NWT-001123, May 1991.

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

[5] Lennox, J., Schulzrinne, H. *Feature Interaction in Internet Telephony*. Feature Interaction in Telecommunication and Software Systems VI, IOS Press, 2000, 38-50.

[6] *Active IETF Working Groups*, available from: <http://www.ietf.org/html.charters/wg-dir.html>

[7] *A portal of commercial development of SIP*, available from: <http://www.sipcenter.com>

[8] Keck, D.O., Kuehn, P.J. *The Feature and Service Interaction Problem in Telecommunications Systems: A Survey*. IEEE Transactions on Software Engineering, Volume 24, Number 10, 1998, 779-796.

[9] Calder, M., Kolberg, M., Magill, E.H., Reiff-Marganiec, S. *Feature Interaction: A Critical Review and Considered Forecast*. The International Journal of Computer and Telecommunications Networking, Volume 41, Number 1, 2003, 115-141.

[10] Cameron, E. J., Griffeth, N. D., Lin, Y.-J., Nilson, M. E., Schnure, W. K., Velthuijsen, H. *A Feature-Interaction Benchmark for IN and Beyond*, IEEE Communication Magazine, Volume 31, Number 3, 1993, 64-69.

[11] Kolberg, M., Magill, E.H. *Handling Incompatibilities between Services deployed on IP-based Networks*, IEEE Intelligent Networks, 2001, 360-370.

[12] Griffeth, N.D., Velthuijsen, H. *The Negotiating Agents Approach to Runtime Feature Interaction Resolution*. Feature Interaction in Telecommunication Systems, IOS press, 1994, 217-235.

[13] Harada, D. Fujiwara, H., Ohta, T. *Avoidance of Feature Interactions at Run-time*, IEEE International Conference on Software Engineering Advances, 2006, 6-12.

[14] Ohta, T., Harada, Y. *Classification, Detection and Resolution of Feature Interactions in Telecommunication Services*. Feature Interaction in Telecommunication Systems, IOS press, 1994, 60-72.

[15] Chan, K.Y., Bochmann, G.V. *Methods for Designing SIP Services in SDL with Fewer Feature Interactions*. Feature Interaction in Telecommunications and Software Systems VII, IOS press, 2003, 59-77.

[16] Chentouf, Z., Cherkaoui, S., Khoumsi, A. *Implementing online Feature Interaction detection in SIP environment: early results*. IEEE International Conference on Telecommunications, 2003, 515-521.

[17] Khoumsi, A. *Detection and Resolution of Interactions between Services of Telephone Networks*. Feature Interaction in Telecommunications Networks IV, IOS Press, 1997, 78-92.

[18] Jouve, H., Gall, PL., Coudert, S. *An Automatic Off-line Feature Interaction Detection Method by Static Analysis of Specifications*. Feature Interactions in Telecommunications and Software Systems VIII. IOS Press, 2005, 131-146.

[19] Kolberg, M., Magill, E.H. *A Pragmatic Approach to Service Interaction Filtering between Call Control Services*. The International Journal of Computer and Telecommunications Networking, Volume 38, Number 5, 2002, 591-602.

[20] Kolberg, M., Magill, E.H. *Managing Feature Interactions between Distributed SIP Call Control Services*. ELSEVIER Computer Networks, Volume 51, Issue 2, 2007, 536-557.

[21] *Study on Architecture Impacts of Service Brokering*, 3GPP TR 23.810.

[22] Gouya, A., Crespi, N. *Service Broker for Managing Feature Interactions in IP Multimedia Subsystem*. IEEE International Conference on Networking, 2007, 54-59.

[23] *Input 3rd Generation Partnership Project (3GPP) Communications Service Identifiers Requirements on the Session Initiation Protocol (SIP)*, IETF draft-loreto-sipping-3gpp-ics-requirements-00.txt

[24] Crespi, N. *A Distributed Mechanism to Resolve Dynamically Feature Interaction in the UMTS IP Multimedia Subsystem*. International Workshop on Applications and Services in Wireless Networks, 2006, 199-206.

[25] Kamioka, E., Yamada, S., *Performance Evaluation of a Seamless Communication on 3GPP-based IP Networks*, The Society for Modeling and Simulation International, the Huntsville Simulation Conference, 2001, 59-64.

[26] Narayanan, S., *Scaling SIP Servers*, IRT Group Meeting, 2002.

[27] Abhayawardhana, V.S., Babbage, R., *A Traffic Model for IMS*, IEEE VTC 2007, 787-791.

[28] Cortez, M., Ensor, J.R., Esteban, J.O., *On SIP Performance*. Bell Labs Technical Journal, 2004, 155-172.

[29] Wanke, S., Sharf, M., Kiesel, S., Wahl, S., *Measurements of the SIP Parsing Performance in the SIP Express Router*, Lecture Notes in Computer Science, Springer Berlin, Volume 4606, 2007.

[30] *INT IMS Platform*, available from: "www.int-evry.fr/rs2m/recherche/noel_eq.php#platform"

[31] *SIP Express Router*, available from <http://www.iptel.org/ser>

[32] *Iptel; the IP Telecommunications Portal*, available from <http://www.iptel.org/>

[33] Calhoun, P., et al. *Diameter Base Protocol*, IETF RFC 3588, September 2003.

APPENDIX

We consider the case when $n_{caller} = n_{callee} = 0$ as exception, and we start by the first statement of this sequence i.e. when one service is invoked: "$n_{caller} = 0$ *and* $n_{callee} = 1$" or "$n_{caller} = 1$ *and* $n_{callee} = 0$". In this case T_{S_ID} must be 0 (only one service is invoked and no interaction could happen), and this result is confirmed in first statement. Next, if we suppose that statement is true after the n_{caller} invocation at the caller side and the n_{callee} invocation at the callee side, we must prove that the next statement, related to next service invocation, must be true. We assume that next service invocation has occurred at the caller side and we show that this statement is true:

$$T_{S\_ID} = ((n_{caller} + 1) + n_{callee} - 1) \times T_{parsing} + ((n_{caller} + 1) + n_{callee} - 1) \times T_{db\_lookup}$$

$$= (n'_{caller} + n_{callee} - 1) \times T_{parsing} + (n'_{caller} + n_{callee} - 1) \times T_{db\_lookup}$$

The same result will be achieved if we consider that the service invocation occurs at the callee side:

$$T_{S\_ID} = (n_{caller} + (n_{callee} + 1) - 1) \times T_{parsing} + (n_{caller} + (n_{callee} + 1) - 1) \times T_{db\_lookup}$$

$$= (n_{caller} + n'_{callee} - 1) \times T_{parsing} + (n_{caller} + n'_{callee} - 1) \times T_{db\_lookup}$$

Table 1: Parameters for Evaluating the Performance of Service Broker

Parameter	Description
T_{caller}	Maximum process time added on S-CSCF of caller for feature interaction management
T_{callee}	Maximum process time added on S-CSCF of callee for feature interaction management
n_{caller}	The number of invoked services at caller side
n_{callee}	The number of invoked services at callee side
t	Maximum process time on Service Broker
X_s	Process time on S-CSCF

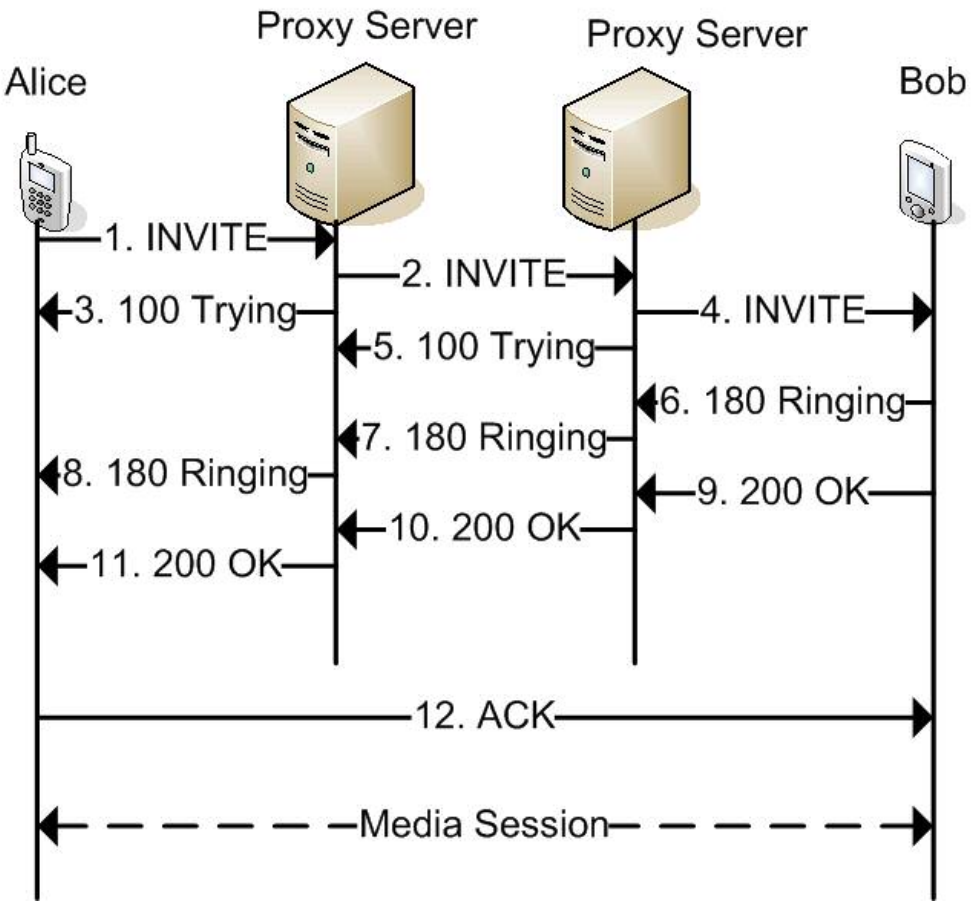

Figure 1: SIP Session Set-up Example
55x50mm (300 x 300 DPI)

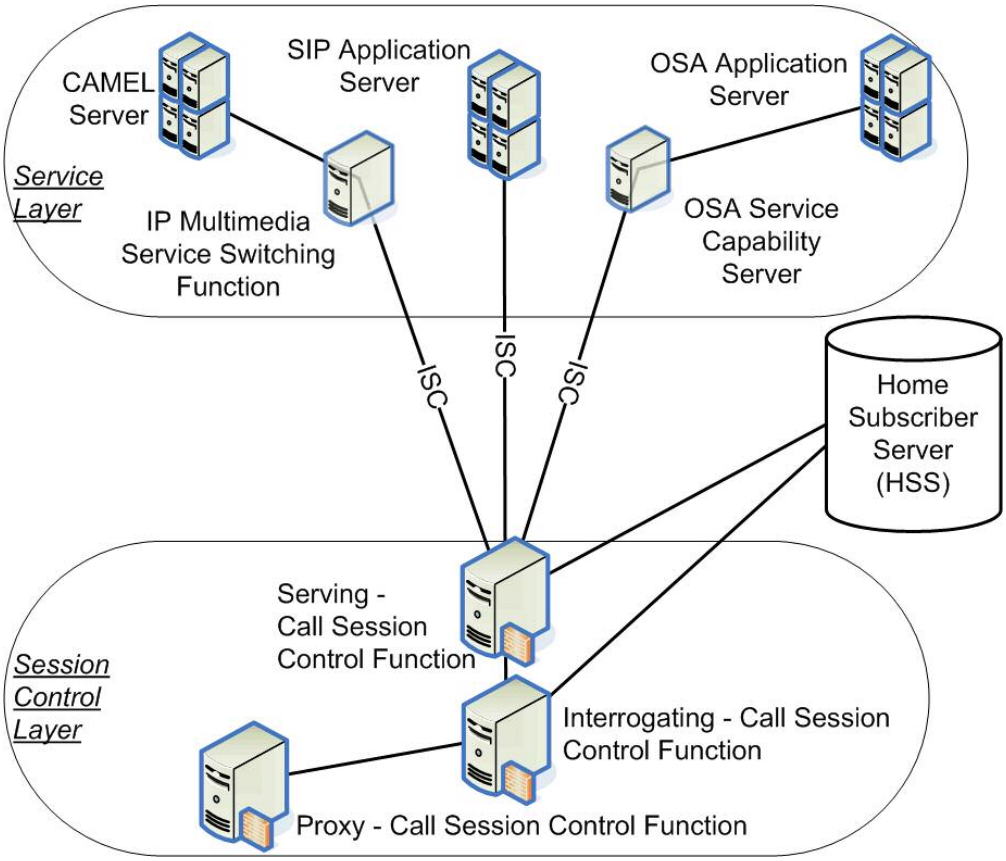

Figure 2: Functional Architecture of IMS
77x66mm (300 x 300 DPI)

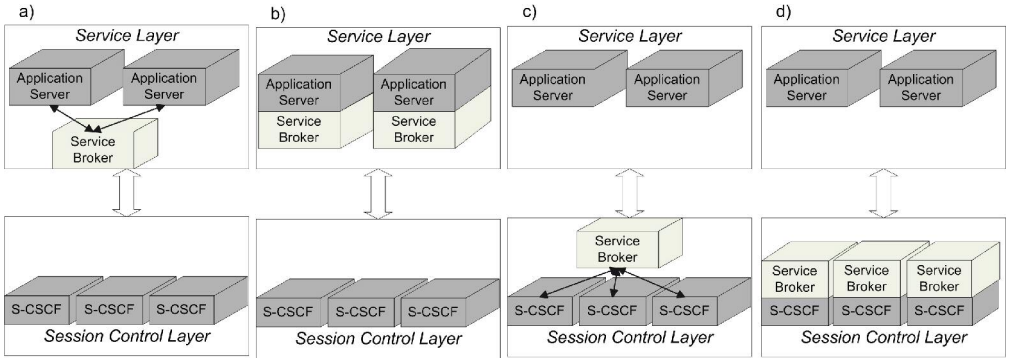

Figure 3: Different Architectural Models for Service Broker
209x74mm (300 x 300 DPI)

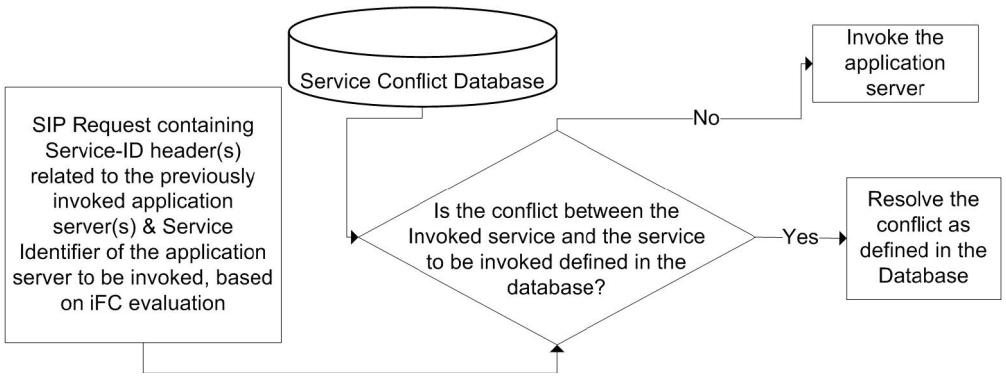

Figure 4: The Service Identifier Comparison Module
132x48mm (300 x 300 DPI)

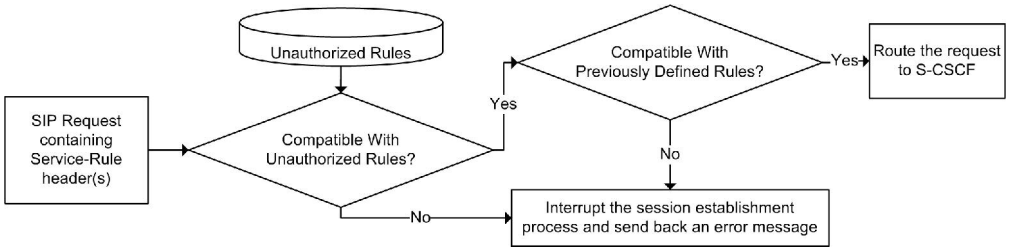

Figure 5: The Service Rule Comparison Module
176x42mm (300 x 300 DPI)

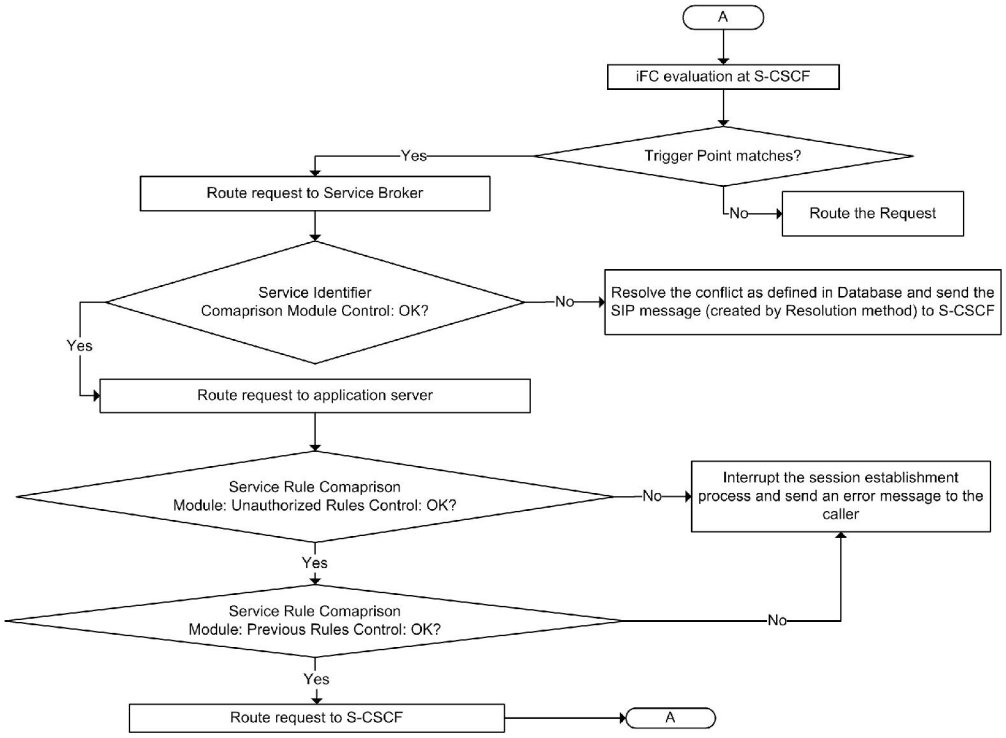

Figure 6: Proposed Feature Interaction Management Algorithm
185x135mm (300 x 300 DPI)

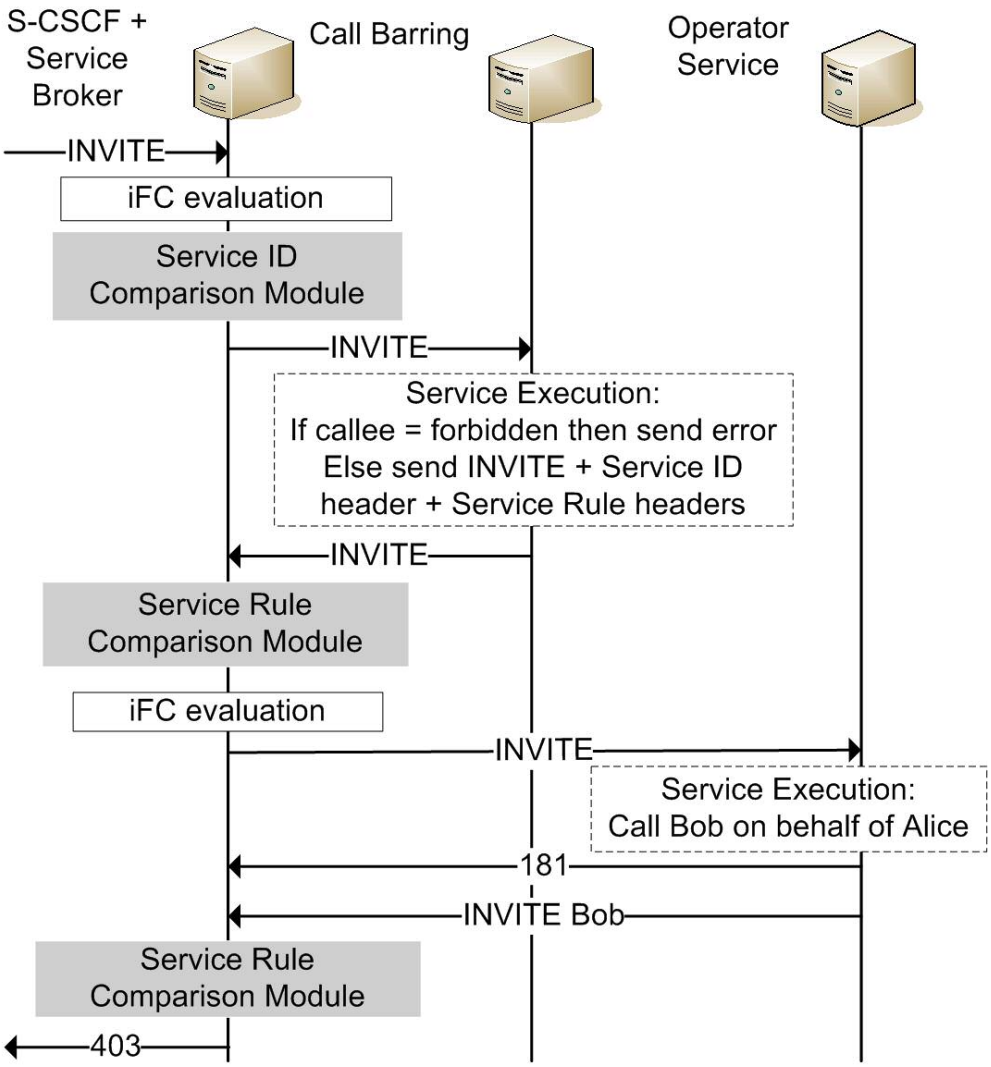

Figure 7: Intra Domain Interaction between Caller's Features
91x97mm (300 x 300 DPI)

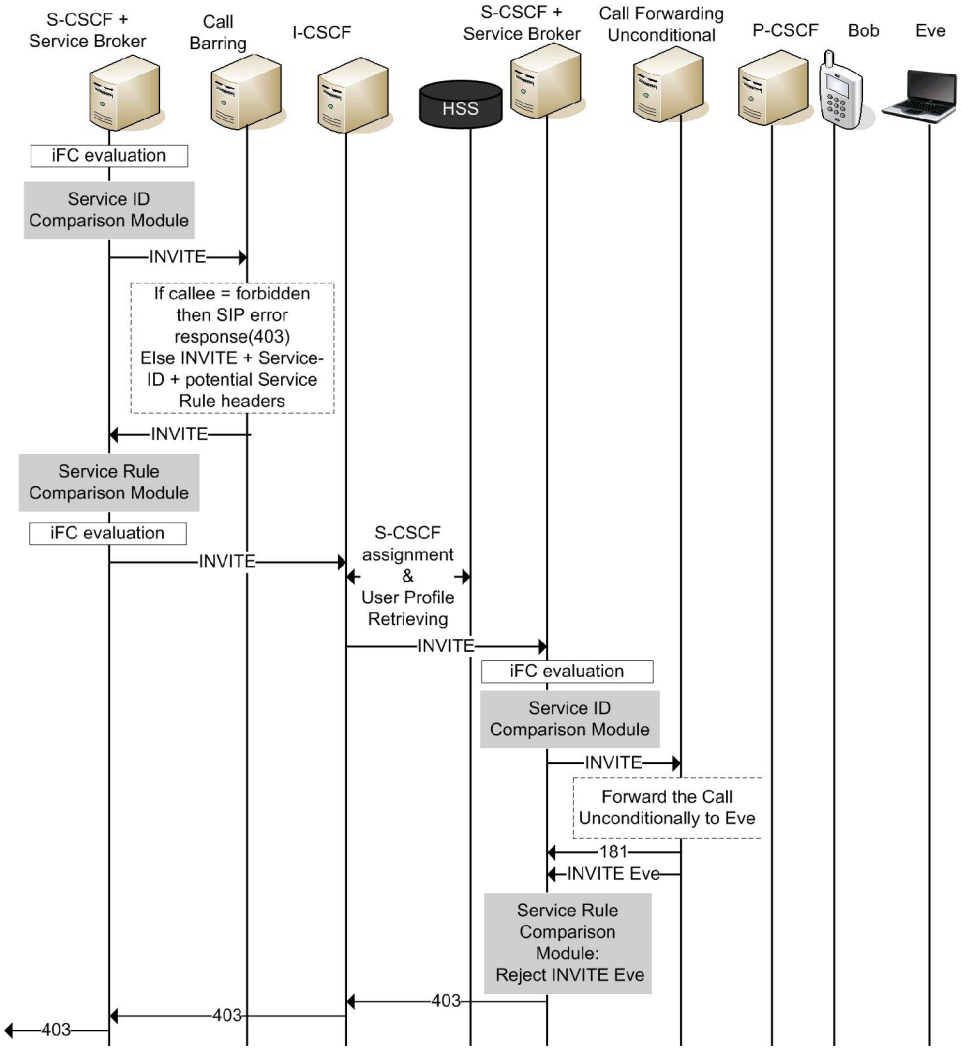

Figure 8: Inter Domain Interaction between Caller's and Callee's Features
137x144mm (300 x 300 DPI)

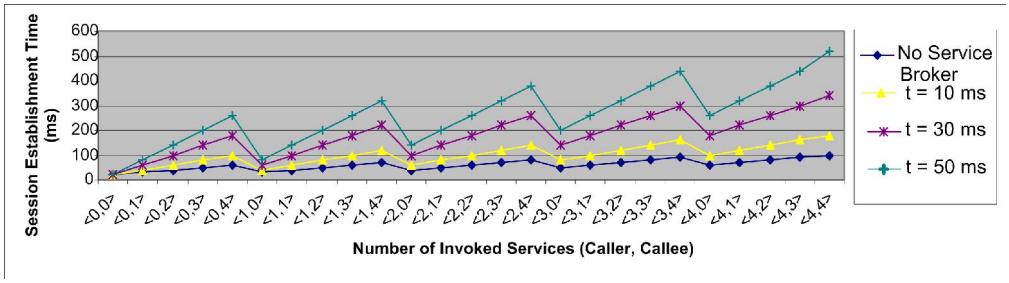

Figure 9: Service Broker Impact on Session Establishment Time
218x60mm (300 x 300 DPI)