Multi-agents SIP architecture for online Feature Interaction detection and resolution

Zohair Chentouf* – Ahmed Khoumsi**

* Dialexia Communications Inc.
1755 Saint Régis, Suite 210 Montreal (QC) H9B 2M9, CANADA
czohair@dialexia.com

** Université de Sherbrooke, Département de génie électrique et informatique
2500 boulevard de l’Université, Sherbrooke (QC) J1K 2R1, CANADA
Ahmed.Khoumsi@USherbrooke.ca

RÉSUMÉ. Cet article présente une solution multi-agents pour la gestion en temps-réel d’interactions de services (IS) dans les réseaux utilisant SIP. Le problème abordé résulte d’une propriété importante de l’architecture de SIP : les services sont répartis sur plusieurs nœuds du réseau, y compris les serveurs d’applications, les tiers fournisseurs, et les appareils d’utilisateur final sophistiqués. Par conséquent, l’information sur la logique des services et la décision d’exécuter ces services sont réparties sur ces nœuds. Ceci complique le problème de gestion des IS. Cet article explique comment ce problème peut être résolu en assignant un Agent de Gestion d’IS (AGIS) à chaque nœud du réseau qui exécute des services. Il explique aussi comment ces agents coopèrent en temps-réel pour trouver une solution commune permettant d’éviter des situations d’IS.

ABSTRACT. This article presents a multi-agents architecture solution for runtime management of feature interactions (FI) in SIP-based networks. The problem that is addressed results from a major SIP architecture property: services are distributed over many nodes in the network, including application servers, third party providers, and sophisticated end user devices. Consequently, information about service logic as well as decision to run services are distributed over those nodes. This complicates the FI problem management. The present paper explains how this problem can be solved by assigning a Feature Interaction Management Agent (FIMA) to every network node that runs services. It also explains how those agents co-operate at runtime to find a common solution to avoid FI situations.

MOTS-CLÉS : Interaction de service, détection et résolution en-ligne, agent de gestion d’interaction de services.

KEYWORDS: Feature Interaction, on-line detection and resolution, Feature Interaction Management Agent.
1. Introduction

An undesired feature interaction (FI) (Cameron et al., 1993) occurs when the joint use of several services induces an undesired behaviour. Henceforth, the terms interaction and FI mean: undesired FI. As an illustrative example, let us consider two services: Originating Call Screening (OCS) and Call Forwarding (CF). OCS allows to block outgoing calls based on the destination number. CF is used to program an automatic redirection of incoming calls towards another destination. We consider three users A, B and C, where A and B are subscribers of OCS and CF, respectively.

To present an example of interaction involving OCS and CF, we consider the following situation: A (by using OCS) has put C’s phone number in a screening list Locs, which implies that any attempt from A’s phone to call C is aborted; and B (by using CF) has programmed a redirection towards C. Thus, if A calls B, the former will be connected to C due to the redirection programmed by B. There is a FI because A has succeeded to call C although the latter is in the screening list Locs.

The most studied topics in the FI scientific community are detection and resolution, most of the results are published in a series of workshops (FIW). Detection consists in checking for the existence of FI, it may be done off-line (before the implementation) or online (during the execution). Once a FI is detected, resolution consists in removing the FI. This could be performed by prevention or by restriction. Prevention means to intervene at the service design level in order to avoid interactions. Restriction means either to run sequentially the interacting services or to exclude one among them. Similar to detection, restriction can be done either off-line or online. Prevention is performed off-line. In the present work, the adopted resolution method is the service exclusion. FIs are classified as single (resp. multi)-component FI when they involve services running in the same (resp. several) network component(s) (Cameron et al., 1993).

This article is structured as follows. Sect. 2 presents the specific aspects of the FI problem in SIP environments. Sect. 3 summarizes the related work. A brief presentation of SIP is given in Sect. 4. The problematic is detailed in Sect. 5 and the proposed architectural solution in Sect. 6. Sect. 7 explains how the proposed solution operates at runtime. This operation is described in an algorithmic detailed manner in Sect. 8, and Sect. 9 concludes the article.

2. FI in SIP architecture

With the new telecommunication architectures, such as Parlay, JAIN, and 3GPP, the telephony service creation is becoming more flexible than before. In such architectures, SIP protocol became in a few years the most dynamic signaling protocol in the service market. These architectures as well as SIP contribute in bringing more flexibility but also in complicating the FI problem. The main characteristics of these technologies, which affect the FI problem complexity, are:
– **Distribution**: the services are located in various network nodes.
– **Openness**: service rooming nodes and third party service providers of different networks may be involved.
– **Heterogeneity**: various underlying technologies are then involved in such a global network. For example, SIP and H.323.
– **End user programmability**: well equipped users are able to design and program their own services.
– **Deregulated service market**: service designers often use non standardized functionalities, under the pressure of the customer demand.

Distribution and openness create service deployment situations where the services are located on different nodes. Thus, the Multi-Component FI problem aspect becomes more frequent. Heterogeneity and non standard service design create problematic situations where services cannot interoperate like expected by their respective designers. This could result in FI. The end user programmability may result in malicious programs that may affect the network security.

3. Related work

There are few related research work reported in the literature. For Internet telephony, (Lennox *et al.*, 1998) identify the technical particularities of IP networks that make it easier to detect and solve some FIs but complicate many others. (Rizzo *et al.*, 2000) and (Kolberg *et al.*, 2001) propose a negotiation protocol between services in order to avoid FIs in SIP-based Internet telephony. The main weakness of the negotiation protocol is that it does not permit to detect single-component FIs nor those that do not contain end user parameters.

At our knowledge, the only research work that studied the whole problem as resulting from the underlying technologies of the emergent architectures has been presented in a series of articles. In (Chentouf *et al.*, 2002), the off-line FI detection procedure has been detailed. (Chentouf *et al.*, 2003) presented the online FI detection in the network nodes. (Chentouf *et al.*, 2004) outlined the off-line and online FI detection and resolution in end user devices. The present article explains the online FI detection and resolution in the network and how it co-ordinates its operation with a detection-resolution procedure which is located at the user device.

4. Brief overview of SIP

SIP (Rosenberg *et al.*, 2002) is inspired from HTTP (Fielding *et al.*, 1999) and has been standardized by IETF. SIP is used for *initiating* and *terminating* an interactive call or media session, and for *changing parameters* of a current session. A session can involve voice and other types of media such as video and text. SIP uses SDP (Session Description Protocol) for defining and negotiating session parameters at either end-point of the communication. SIP supports name mapping and redirection functionality, and thus, permits user mobility.
A typical SIP architecture consists of SIP UAs (User Agents) and SIP Servers. A SIP UA is associated to a user and consists of two parts: a UAC (UA Client) and a UAS (UA Server). UAC sends requests to UASs and SIP servers, and receives responses from UASs and SIP servers; and UAS receives requests coming from UACs and proxies (the latter being defined below), and accepts or rejects them by sending response messages. INVITE is the most known request, which is used to initiate a session. There exist three types of SIP servers:

A registrar handles REGISTER requests received from UACs. Each REGISTER request contains the current address of the UA that has sent it, and the registrar can then update its database of addresses.

A redirect server receives an INVITE request from a UAC and replies back with a response containing the right address where the INVITE request must be sent.

A proxy receives a request and forwards it to another proxy or to a UAS. When a proxy receives a response, it forwards it to the sender of the corresponding request (may be proxy or UAC). Thus, a SIP request or response can traverse several proxies before reaching its final destination (UAS or UAC).

Each SIP response (in reply to a request) has a reply code (RC) ranging from 100 to 699 and a reason phrase (RP) and will be denoted by a “RC-RP”. As a simple example, when a UAS receives an INVITE (request), it may reply by sending sequentially the following three responses:

100-Trying means the UAS has received the INVITE and is processing it.
180-Ringing means the UAS has received the INVITE and is alerting the user.
200-Ok means the INVITE is accepted (e.g., the called user has picked-up).

Note that a UA is connected to no more than one proxy. In the sequel, by SIP domain we mean any network consisting of a proxy and all the UAs connected to it.

5. The problem and the solution principle

Let us call Administration Authority (AA) any party (network operator, service provider, or end user) capable of service rooming and execution on a single self-owned network node. Let us now adopt the number of the involved AAs in a given telecommunications network as a FI problem classifying point of view. The following situations could then be considered:

– S1: All the services are owned by a single AA. The FIs are single-component and their detection can be done off-line because all the services are known. The resolution may be done off-line in order to correct service designs (prevention). It also can be performed online if it consists of a restriction. See, for example, (Chentouf et al., 2004).

– S2: Services are owned by different AAs. The FIs may be single-component or multi-components. In the latter case, if the involved components belong to the same AA, the problem can be reduced to S1. If the components belong to more than one AA, two problems have to be solved:
- Lack of information: each AA lacks information about the other AA services.
- Lack of control: a solution that is implemented in a given AA cannot act on services that belong to another AA.

The proposed solution to these problems is based on the following requirements:
- Transparency: the solution has to know every service that is deployed in the network. This requirement solves the first problem (lack of information).
- Visibility: the solution has to be able to act on any service that is deployed in the network. This requirement solves the second problem (lack of control).

6. FIMA architecture

A multi-agent architecture solution is proposed. It consists of a set of software agents called FIMA (FI Management Agents) that are assigned to the network nodes that room services. Hence, the application server as well as the end user devices that are capable of rooming and executing services, are equipped with FIMAs. In other words, every AA in the network should be equipped with a FIMA. All the FIMAs are provided with the FI detection-resolution procedure.

The FIMA deployment in a SIP architecture is done as follows. The application server that rooms the network-located services has to contain a special FIMA, called NFIMA (Network FIMA), which plays the role of centralized FI solution. NFIMA has the responsibility of co-ordinating the operation of all the other FIMAs. The latter are called UFIMA (User FIMA). The NFIMA has to be able to control the proxy server, which is the SIP communication central node. The aim is for the NFIMA to be able to control all the SIP communication (as explained in Sect. 7). In most cases, the application server is located with the proxy server (Fig. 1).

This deployment manner corresponds to a single SIP domain. Recall that a SIP domain is the pool of user devices that register with the same SIP registrar server. Usually, the latter is the proxy server itself, but it also can be a stand-alone server. In such a case, all the SIP communication still goes through the proxy server. That is, a SIP domain contains one NFIMA assigned to the network application server, which is a centralized AA. It also contains as much UFIMAs as other AAs (user devices).

Figure 1. FIMA deployment

Figure 1 depicts an example of the FIMA deployment in a simple SIP architecture. There is only one SIP domain that encompasses a proxy, which also is
the registrar server and the application server. The operation of the FI solution is the following: Each time an end user changes the user device-located services design, his local UFIMA does the following two tasks:

- Off-line FI management in UFIMA: the local UFIMA operates a FI detection and resolution. This mechanism is described in details in (Chentouf et al., 2004).

- Off-line service model provision: the local UFIMA sends a model of its local services to NFIMA. Such a model can be written in FIML: a language that has been designed for that purpose (Chentouf et al., 2004). The provision of an FIML model is done using REGISTER: the SIP message that is sent by any SIP node to the registrar (the proxy itself, in Fig. 1) in order to provide its location. The FIML code has to be inserted in the message body of REGISTER. Since the SIP standard does not make any restrictions on REGISTER issuing, the FIML models can be sent each time UFIMA has to do that.

7. Online FI management in NFIMA and UFIMA

Given a call session that involves a caller and a callee user devices, each one equipped with a UFIMA, the basic runtime tasks that these UFIMAs and the NFIMA have to perform are the following:

- The caller UFIMA has to postpone running services that are triggerable by INVITE until receiving Trying message from NFIMA.

- At the receipt of a SIP message, NFIMA launches the detection-resolution procedure with the call participants service models. The adopted resolution method is the exclusion of one of the interacting services in order to avoid the interaction. The derived FI resolution may result in the following:
  - NFIMA may operate exclusion on the proxy-located services. In such a case, NFIMA has to inform the caller and the callee UFIMAs about this exclusion because they could have some actions to run in reaction to that resolution (see (Chentouf et al., 2004) for more details).
  - NFIMA may need to order the caller and/or callee UFIMA to operate exclusion on some of their local services.
  - NFIMA may need to order the caller and/or callee UFIMA to operate exclusion on some of their local services depending on the next SIP message the UFIMA will issue. This manner to detect possible interactions is called anticipation.

- The information about the NFIMA operated exclusions and the exclusion orders that have to be sent (by the NFIMA) to the UFIMAs, are all written in a report that is sent in the following manner.
  - If the report was derived after receiving a request, the NFIMA informs:
    - the callee UFIMA by including the report in the request, and
    - the caller UFIMA by including the report in the response (Fig. 2).
  - If the report was derived after receiving a response, the NFIMA informs:
    - the caller UFIMA by including this report in the response, and
    - the callee UFIMA by including the report in a new request (Fig. 3).
Figure 2. *NFIMA operation after receiving a SIP request*

Figure 3. *NFIMA operation after receiving a SIP response*

This is an abstraction of the (online detection-resolution) algorithm of Sect. 8. This algorithm presents in detail how the above abstracted operations are distributed and synchronized with the SIP messages that arise during a whole SIP call initiation, establishment, and termination. Here are some examples that will help understanding the logic of the algorithm. In the following, user A is the caller and user B the callee. The corresponding UFIMAs are UFIMA_A and UFIMA_B, respectively.

**Example 1.1: Email – OCS**

User A has a device-located programmed Email client that sends an advice email to the callee, each time a call is requested to that callee. User A has also a device-located OCS service (see Sect. 1) that allows A to specify a screening list Locs of users. If a call is requested to a destination that belongs to Locs, OCS blocks the call. Suppose user A has put user B on Locs. In terms of SIP, the two services are triggerable by the INVITE message.

The off-line detection-resolution already made by UFIMA_A on its own services should have detected an interaction between Email and OCS that consists in sending an email to a destination that is on Locs. The derived resolution is expressed in terms of a condition-action rule:

- if a call is requested to X, which means when an INVITE is about to be sent to X, and if X belongs to Locs, then block emails to X.

An online resolution is operated by UFIMA_A in the following manner: before
sending an INVITE, UFIMA_A checks the off-line detection-resolution derived rules and apply those among them which are triggerable by INVITE. In our example, INVITE triggers the (above-mentioned) rule that consists in checking if user B is in the black list and blocks the email to user B.

This corresponds to Step 3.a of the algorithm. None of Figs. 2 or 3 scenarios are applied because no SIP message is sent to the proxy or the callee UFIMA.

\[
\begin{array}{ccc}
\text{user A} & \text{proxy} & \text{User B} \\
\text{restriction: email B} & \text{Display: forbidden destination} & \\
\end{array}
\]

**Figure 4. Example 1.1**

**Example 1.2: Email – OCS**

Suppose this time OCS is located in the proxy. Thus, the off-line detection operated by UFIMA_A derives no resolution rule. UFIMA_A sends an INVITE to the proxy but, contrary to the usual basic behaviour, does not run any service triggered by INVITE until receiving the Trying message from NFIMA (step 3.b of the algorithm).

NFIMA receives INVITE, loads the service models of A and B and launches an online detection-resolution (step 4 of the algorithm). Recall that the device-located service models were sent to NFIMA by their respective UFIMA owners in the off-line service model provision step (step 1 of the algorithm).

The detection-resolution procedure detects the FI between Email and OCS (the same as in Ex. 1.1) and derives a rule that consists in blocking the email. Since the Email service is located in the user A device, NFIMA includes the order to block the email in the Trying response that is sent back to user A (step 4.b of the algorithm).

UFIMA_A receives Trying and extracts the NFIMA order. Applying the latter results in blocking the email towards user B (step 7 of the algorithm).

This example scenario corresponds to the Figure 2 scenario. Since no SIP message is issued towards UFIMA_B, none of the two services will be triggered and, thus, NFIMA does not need to send a report to UFIMA_B.

\[
\begin{array}{ccc}
\text{user A} & \text{proxy} & \text{user B} \\
\text{INVITE} & \text{100 Trying} & \\
\text{execute: restrict Email} & \text{403 Forbidden} & \\
\text{ACK} & \\
\end{array}
\]

**Figure 5. Example 1.2**
Example 2: AR - VMBL
User B has a device-located AR (Automatic Recall) service. It is triggerable by (the SIP response) Busy and consists in initiating a call towards a user that has called user B when B line was busy. User B also has a network-located VMBL (VoiceMail on Busy Line) which is triggerable by Busy and consists in forwarding the caller, when B line is busy, to a voicemail server in order to leave a voice message.

UFIMA_A sends an INVITE to the proxy. NFIMA receives the INVITE, loads user A and user B service models and starts the detection-resolution procedure. The latter detects a FI between AR and VMBL. The FI arises if user B responds Busy. The latter signal triggers both of AR and VMBL. The two services are not possible to run together, either user A is asked to leave a voice message (VMBL) or hears a busy tone and then is recalled by AR when user B line becomes idle.

Suppose the resolution rule derived by NFIMA consists in excluding AR service at the user B device. NFIMA includes the order to exclude AR in the INVITE message to forward to B (step 5.a of the algorithm). However, the order contains a condition: if B line is busy, exclude AR. As one can notice, NFIMA has operated an anticipation here on what the callee response could be, since the callee response has not been issued yet. This mechanism of anticipation is the step 5 of the algorithm.

UFIMA_B receives INVITE and extracts the conditioned order to exclude AR (step 7 of the algorithm). Suppose user B was busy talking to someone. So, issuing the Busy signal towards the proxy implies the effective exclusion of AR. This example scenario corresponds to the Figure 2 scenario.

![Diagram of Example 2](image)

Figure 6. Example 2

Example 3: F-Email – Email
User A has a F-Email (Forbid Email) service that allows to ask the proxy to order any user device with which user A device is involved in a call not to send an email to user A. User B has Email service that is triggerable by the BYE request. This means every time user B ends a call, an email is sent to the other call participant, say for commercial advice, for example.

When NFIMA receives INVITE from UFIMA_A, it loads user A and user B service models, launches the online detection-resolution. No interaction is detected at this stage of call processing. When receiving the 200 OK response from the callee, NFIMA launches again the detection-resolution algorithm anticipating on the
future actions of UFIMA\textsubscript{A} and UFIMA\textsubscript{B} (step 9 of the algorithm). A resolution rule is then derived. It consists in excluding Email at user B device if user B sends BYE to end the call. NFIMA waits to receive a request from the caller to write the order in (step 9.b of the algorithm). When NFIMA receives the ACK request from the caller, it includes the order and sends it to the callee. UFIMA\textsubscript{B} receives ACK, extracts the exclusion order and waits for the condition to be true in order to apply the exclusion. When user B hangs on to end the call, UFIMA\textsubscript{B} excludes the Email service (step 13 of the algorithm). This example scenario corresponds Figure 3.

8. FI online detection-resolution algorithm

The FI online detection-resolution algorithm has been outlined and illustrated in Sect. 8. Outlined by the two situations of Figs. 2 and 3, and illustrated by Examples 1.1, 1.2, 2 and 3. Let us present in detail this algorithm which has been implemented. It has been validated with many examples.

1. Each time the end user updates the services or preferences roomed in his terminal, UFIMA sends the corresponding FIML models to NFIMA. We use the SIP REGISTER message to realize this communication. The models are contained in the message body. Recall that REGISTER is used to inform of the localization of the sender of the message and that he is ready to send and receive calls. NFIMA saves the models received in a local database.

2. Then, UFIMA launches the detection procedure to detect the possible interactions that might occur between services or preferences. If an interaction is detected, the resolution procedure determines the restriction action to be executed. This action is saved in a database so as to be executed before the conflicting services or preferences are executed during a call.

3. When the caller sends a call, his UFIMA:
   a. Applies the possible restrictions decided in Step 2.
   b. Does not start immediately the services and preferences having INVITE as a starting condition. It does it only when it receives Trying message.

4. When INVITE is received by the proxy, NFIMA: 1) loads the models of the services and preferences owned by the call participants which are located in user agents, and 2) passes the models to the detection-resolution procedure in order to detect and solve FIs, if any. Thus, the problem of unknown services and preferences is reduced. When the proxy receives an INVITE message from the caller, NFIMA, which is located with the proxy and is capable to intercept its incoming and outgoing SIP messages, can identify the caller and callee because they are indicated in the INVITE message. If the resolution procedure has identified an action to apply, several cases are possible.
   a. If the action consists in excluding a service or a preference located in the proxy, NFIMA executes the restriction. If the callee (resp. caller) has a reaction to this restriction, NFIMA includes this restriction information in the INVITE (resp. Trying). Thus, the caller and callee can start their reactions to the restriction.
b. If the restriction is related to a service or preference triggered by INVITE in the caller terminal, NFIMA includes the restriction order in the Trying message. At the reception of the latter, caller UFIMA will execute this order.

c. If the service to exclude is located in the callee terminal, NFIMA includes the order in INVITE.

5. We call anticipation the FI detection which is based on possible events, instead of actual event executions. Thus, the detection procedure determines if some services and preferences need to be excluded depending on the callee response (messages SIP 486 Busy Here, 200 OK, timeout, or others). If this is the case and
   a. if the restriction must be executed in the callee terminal, then NFIMA includes a corresponding order in the INVITE message.
   b. if the restriction must be executed by the caller or NFIMA, then the latter waits the callee response.

6. NFIMA sends Trying to the caller and INVITE to the callee.

7. The caller UFIMA applies the restrictions included in Trying, if any. The callee UFIMA applies the restrictions decided in Step 2 and those included in INVITE, if any. It then sends a SIP response to the proxy.

8. After it receives the callee’s response, NFIMA checks if some restrictions identified in Step 5 must be triggered by this response.
   a. If this is the case, the restrictions in question are executed immediately if they are related to services located with the proxy. The information taken by the NFIMA from the services must be:
      - included in the callee response, if the caller has anticipated a reaction, and
      - saved in order to include it in the message that will be sent by the caller in reaction to this response, if the callee has anticipated a reaction.
   b. If, on the contrary, the restrictions of Step 5 triggerable by this response are related to services located in the caller terminal, then they are included in the response received from the callee.
   c. Then, NFIMA checks if new interactions will be generated by this response. Once again, the restriction is executed by NFIMA if it concerns services or preferences located with the proxy, included in the response if it concerns the caller, and saved in order to include it in the message that will be sent by the caller in reaction to the callee’s response.
   d. If NFIMA must execute restrictions in this step, then it will inform the caller (resp., callee) if the latter has anticipated reactions, by using the callee response (resp., the message sent by the caller in reaction to the callee response).
   e. If any NFIMA action (triggering a service or executing a restriction) must prevent the callee response from reaching the caller, NFIMA will include the original callee response in the first SIP message that it will send to the caller.
The response towards the caller has not still be sent; NFIMA is waiting to receive the results of a last anticipation.

9. After this call processing step, and depending of the response just received, NFIMA launches the detection in order to anticipate on the possible interactions considering the possible actions of the caller and the callee (abort the call, termination after conversation, transfer, put on hold, etc.)
   a. If the caller has to execute restrictions, NFIMA includes the corresponding order in the response received from the callee.
   b. If the callee has to execute restrictions, NFIMA will wait the reception of a SIP message from the caller, in reaction to the response received, in order to include in it the restriction order.
   c. If the proxy must execute restrictions to which the caller (resp., the callee) has anticipated reactions, then the proxy informs him using the response (resp., the message sent by the caller in reaction to the response). For example, the message ACK sent in reaction to 200 OK.

Naturally, the order restrictions depending on possible future events are executed by the involved UFIMAs only if the latter produce themselves these events. For example, the caller or callee executes a restriction conditioned by the BYE message only if the latter is really sent.

10. NFIMA sends a SIP message towards the caller.
    a. This message may be the response received from the callee.
    b. It may also be another message which replaces the callee response. In this case, it must include the information inferred in Step 7.e.

11. Caller UFIMA extracts the order restrictions and the NFIMA’s action reports and then applies the necessary exclusions and reactions. It could then send to NFIMA a SIP reaction to the message just received. That depends on the type of the latter.

12. If this is the case, NFIMA could introduce in this message some restrictions or information on actions it has executed. These restrictions and actions, if any, have been inferred in Steps 7.a, 7.c, 7.d, 8.b and 8.c. NFIMA sends the message to the callee.

13. Callee UFIMA receives the message, from which it extracts the restrictions and applies them.

9. Conclusion

This article explained how the FI problem can be handled at runtime in a SIP architecture by a set of software agents capable of FI detection and resolution. These agents co-operate in order to infer a common solution to avoid runtime FI situations. The FI solution the agents bring to the SIP architecture, aims to address the distribution issue that results from the fact that services are located on different network nodes (application servers, third party providers, and end user devices). That is to say, the service location distribution engenders a problem of lack of
information about service logic. The agents’ co-operation purpose is to exchange information about service logic, and to execute at the edge of the network (end user devices) the FI resolution actions that are inferred by a centralized FI solution. The latter is called NFIMA (Network FI Management Agent), and agents that are situated in the edge of the network are called UFIMA (User FIMA).

The agent co-operation uses a synchronous communication, that is conveyed by SIP: NFIMA inserts FI resolution orders in SIP messages, and UFIMAs extract those messages and execute them. Synchronization of the FI resolution actions with the SIP traffic is handled by NFIMA. Given a step of the call processing, in order for NFIMA to intervene at the suitable moment, it performs an anticipation on what the future SIP messages could be. According to all the possible SIP messages that can be issued, NFIMA issues FI resolution orders that are suitable to each of the corresponding possible SIP situations. The UFIMAs receive these orders and decide which to execute, according to the SIP messages they have to issue.

The presented solution considered the case of a single SIP domain. Its extension to multi-SIP domains is not difficult, it has not been presented for lack of space.

In a future work, a quantitative study of the processing time costs of such a FI solution will be addressed.

Bibliographie


FIW, Series of eight Feature Interactions Workshops (FIW), 1994-2005. IOS Press.


