Presence Interaction Management in SIP SOHO Architecture

Zohair Chentouf[1] and Ahmed Khoumsi[2]

[1] Dialexia Communications Inc., Montreal, Canada
czohair@dialexia.com
[2] Université de Sherbrooke, Sherbrooke, Canada
Ahmed.Khoumsi@USherbrooke.ca

Abstract. A SOHO (Small Office or Home Office) architecture can be sketched as an architecture that involves an ITSP (Internet Telephony Service Provider) and subscribers. The ITSP offers SIP protocol based telephony and presence services for subscribers. A subscriber can have several presence capable devices that periodically publish the user presence status. The paper defines and proposes a solution to the presence interaction (PI) problem.

Keywords: SOHO, Presence interaction (PI) detection and resolution, PI Management Agent (PIMA), PI Management Language (PIML), order relations.

1 Introduction

In this article, we consider a particular architecture, called SOHO (Small Office or Home Office), containing an Internet Telephony Service Provider (ITSP) that offers SIP based telephony and presence services for users. Each user can own one or more devices. The devices that are presence capable, periodically publish the user presence status to the ITSP presence server. A *presence interaction* (PI) arises when two or more devices owned by the same user publish contradictory presence status, for example, *available* and *out-for-lunch*.

We propose a multi-agents architecture for managing PI. The detection and resolution procedure is based on order relations. A PI Management Language (PIML) is used to express formally the relevant information for managing PI.

Section 2 introduces the presence service and our proposed architecture. In Section 3, we introduce PI and the approach used for solving them. In Section 4, we propose a Multi-agents approach for managing PI. Section 5 presents PIML that is used to model presence status and resolution policies. And we conclude in Section 6.

2 Presence Service and Proposed Architecture

SIP offers an architecture and communication mechanisms for implementing a presence publishing service. The SIP presence architecture [1] encompasses user terminals that publish presence information to a presence server, using the PUBLISH message [2]. Users which publish their presence information are called presentities.

The presence server composes the presence information that is published by different terminals that belong to a same presentity in order to produce a single presence document. The users who are interested to be notified about the presence status of a given presentity, subscribe to this service by sending a SUBSCRIBE message to the presence server. Those users are called watchers. Every time his presence status is changed, the presentity publishes the new presence information. The presence server then notifies all the watchers of that presentity about his new presence information by sending a NOTIFY message to every one, including the presence document. In [3], presence information as well as filtering policies are coded in an XML-based language called PIDF. In [4], Schulzrinne proposed RPID that extends PIDF.

We extend the SIP presence architecture by proposing the architecture depicted in Figure 1. The SOHO (Small Office or Home Office) network gathers the devices owned by the same user. The ITSP (Internet Telephony Service Provider) extends the SIP presence server in order to manage the SOHO network.

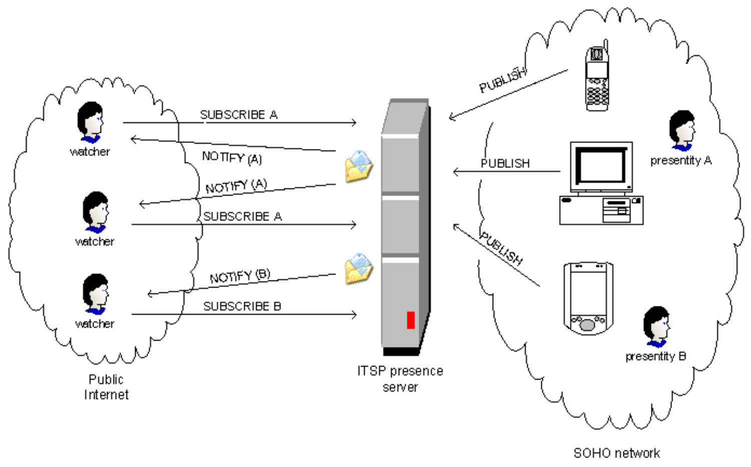

Fig. 1. ITSP presence architecture

3 Presence Interactions and Solution Approach

For the purpose of the current work, we propose the following eight presence status, which are for example used in [5] and [6]: Available (1), Away (2), Busy (3), In-a-meeting (4), Not-available (5), On-the-phone (6), On-vacation (7), Out-for-lunch (8). Notice that the On-the-phone status is special in that it is temporary and not intentional. All the other presence status are intentionally set by the user.

We define the presence interaction (PI) problem as the situation where two or more devices owned by the *same user* report two or more different presence status of the user. This could happen, for example, when a device publishes Out-for-lunch while another publishes On-the-phone. The resolution of a detected PI will consist in keeping one among the conflicting presence status in the manner we will explain. For simplicity, we consider here only PI involving exactly *two* (conflicting) devices.

3.1 PI Classification, Order Relations

PI are classified into two categories, denoted OR and RP:

Obvious Resolution (OR) interactions: among two conflicting status, the selection of the one to be excluded is obvious. In our case, OR interactions involve the On-the-phone status, because the latter is special in that it is not intentional. All the other presence status are intentionally set by the user. The On-the-phone status should not contradict a user who, for example, receives a call on a device at a period during which he has chosen to appear as Out-for-lunch on another device. The On-the-phone status should be excluded and the other status should appear instead.

Resolution Policy (RP) interactions: among two conflicting status, the selection of the one to be excluded needs to conform to a specified policy. The latter is based on the following two types of order relations denoted SOR and DOR:

Status order relations (SOR): A SOR is an order relation between status. Let us consider two status *S1* and *S2* and a SOR s*or*. If *S1 sor S2*, then the policy based on so*r* consists in excluding *S2*. “is more precise than” is an example of SOR. For example, Away is more precise than Not-available. For our eight presence status identified by 1, 2, …, 8, the SOR “is more precise than” implies the following pairs (2,5), (3,5), (7,2), (8,2), (4,3), where *(i,j)* means “*i* is more precise than *j*”.

Device order relations (DOR): A DOR is an order relation between devices. Let us consider two devices *D1* and *D2* and a DOR *dor*. If *D1 dor D2*, then the policy based on *dor* consists in excluding the status published by *D2*. “is more trustworthy than” is an example of DOR, which can be used by assigning trustworthiness weights to devices. For example, the user can decide to assign more trustworthiness to cell phone than to office phone.

3.2 PI Resolution Procedure

PI resolution policies are specified by the SOHO administrator and the end users. We suppose the ITSP provides the suitable interface for the SOHO administrator as well the users in order to specify those policies. The ITSP presence resolution solution is contained in a PIMA (Presence Interaction Management Agent) and is based on the use or SOR and DOR. We consider that for *every user*, we may have a set of SORs $\{sor_1, \ldots, sor_n\}$ which are ordered by priority, that is, sor_i has priority over sor_{i+1}. We also may have a set of DORs $\{dor_1, \ldots, dor_n\}$ where dor_i has priority over dor_{i+1}. We also assume that priorities may be defined between some pairs (sor_i, dor_j).

Some order relations correspond to policies specified by the SOHO administrator and will therefore be called *admin-based order relations*. Other order relations correspond to policies specified by the users (presentities) themselves and will therefore be called *user-based order relations*. For example, the SOR “is more precise than” should be specified by the SOHO administrator, while the DOR “is more trustworthy than” should be specified by the users.

Given two status *S1* and *S2* published by devices *D1* and *D2*, respectively, PIMA solves the interaction *S1*-*S2* by applying the following resolution procedure:

Step 1: *Comparison using SOR, assuming that each sor_i has priority over sor_{i+1}*

Check if *S1* and *S2* are comparable using sor_1, i.e., "*S1* sor_1 *S2*" or "*S2* sor_1 *S1*". If this is the case, the best status wrt sor_1 is the solution of Step 1.

If this is not the case, check if *S1* and *S2* are comparable by sor_2. And so on, we iterate until either we reach a sor_i that permits to compare *S1* and *S2*, or we reach sor_n without being able to compare *S1* and *S2*. In the latter case, we say that *S1* and *S2* are SOR-incomparable. In the former case, the best status wrt sor_i is the solution of Step 1.

Step 2: *Comparison using DOR, assuming that each dor_i has priority over dor_{i+1}*

We proceed iteratively as in Step 1, but by comparing devices instead of status. If no dor_i permits to compare *D1* and *D2*, we say that *D1* and *D2* are DOR-incomparable. Otherwise, Step 2 provides a solution *Dv* (v = 1, 2).

Step 3: we have the following six situations:

3.a: Neither Step 1 nor Step 2 provides a solution. In this case, the resolution procedure provides no solution.

3.b: Step 1 provides a status *Su* as a solution and Step 2 provides no solution. *Su* is the adopted solution.

3.c: Step 2 provides a device *Dv* as a solution and Step 1 provides no solution. The status published by *Dv* is the adopted solution.

3.d: Steps 1 and 2 provide compatible solutions, that is, the solution of Step 1 is the status published by the device which is the solution of Step 2. This status is the adopted solution.

3.e: Step 1 and 2 provide incompatible (or contradictory) solutions, that is, the solution of Step 1 *is different from* the status published by the device which is the solution of Step 2. In this case, let sor_i and dor_j be the two order relations providing the solutions of Steps 1 and 2, respectively. Recall that a priority may have been defined between sor_i and dor_j.

3.e.1: if such a priority has effectively been defined: we select the solution provided by the order relation that has priority over the other.

3.e.2: otherwise: the resolution procedure provides no solution.

4 Multi-agents Architecture for Managing PI

We propose a multi-agent architecture solution to manage the problem of PI. The agents are called FIMA (Feature Interaction Management Agent) because the proposed solution is aimed to be integrated with a method for managing feature interactions (FI) proposed in [7]. Two types of FIMA are used: several UFIMA (User FIMA) and one NFIMA (Network FIMA) (Fig. 2). A UFIMA is assigned to each device and the NFIMA contains the PIMA and is assigned to the ITSP.

A user has a single interface to manage presence preferences. This interface may be managed by any UFIMA that is located on any device owned by the user. At any time, the user can access the interface in order to set his presence preferences. Those preferences are used to specify the so-called user-based order relations, that is, order relations corresponding to policies specified by the users (presentities). For simplicity, in the following we consider we have a single user-based relation, namely the DOR "is more trustworthy than". The user presence preferences should contain the trustworthiness weighting of all the devices. UFIMA uses the SIP REGISTER message to communicate the weighting information to PIMA (contained in NFIMA), coded in PIML (Figure 2). The general purpose of PIML is to express formally the relevant information for managing PI.

We suppose the registrar server (the server that is responsible of processing REGISTER) located in the same node as the presence server. Otherwise, the registrar has to communicate the received PIML models to the presence server in a suitable manner. Based on this trustworthiness information (coded in PIML) provided by UFIMA, PIMA constructs the DOR "is more trustworthy than" that will be used when executing the resolution procedure for that user.

In the same way, the SOHO administrator has an interface to transmit to PIMA (using REGISTER) necessary information (coded in PIML) for the construction of admin-based order relations, that is, order relations corresponding to policies specified by the SOHO administrator. The SOHO administration interface is managed by any UFIMA that runs on any device owned by the SOHO administrator. For simplicity, in the following we consider we have a single admin-based relation, namely the SOR "is more precise than".

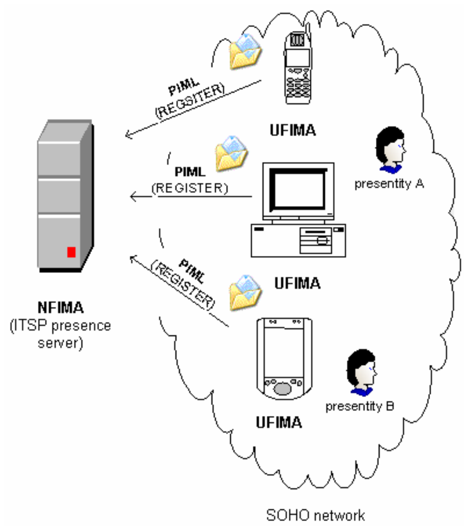

Fig. 2. FIMA-SOHO architecture

5 Presence Interaction Management Language (PIML)

Example of PIMLcode using a SOR

```
1. Owner
2.           caller = soho_admin@company.com
3. SOR: MorePrecise
4.           AWAY, NOTAVAILABLE
5.           BUSY, NOTAVAILABLE
```

```
6.          VACATION, AWAY
7.          LUNCH, AWAY
8.          MEETING, BUSY
```

Lines 1-2 indicates that soho_admin@company.com is a SIP address that is bound to the SOHO administrator currently used device.
Line 3: indicates that the following lines define a SOR called MorePrecise.
Lines 4-8: each line S1,S2 means that S1 is more precise than S2.

Example PIML code using a DOR

```
1. Owner
2.          caller = user_21@company.com
3. DOR: MoreTrustworthy
4.          user_21@company.com, beloxi@company.com
5.          beloxi@company.com, manager@company.com
```

Line 3: indicates that the following lines define a DOR called MoreTrustworthy.
Lines 4-5: each line D1,D2 means that D1 is more trustworthy than S2.

6 Conclusion

In this article, we proposed a solution to the presence interaction (PI) problem that arises when two or more devices owned by the same user publish contradictory presence status. For future work, we plan to study PI involving more than two status and to consider other types of relations. We also plan to consider status that can be combined, instead of selecting a single status.

References

1. Day, M., Aggarwal, S., Mohr, G., Vincent, J.: Instant Messaging/Presence Protocol Requirements. RFC 2779, IETF, February 2000.
2. Niemi, A.: Session Initiation Protocol (SIP) Extension for Event State Publication. RFC 3903, IETF, October 2004.
3. Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., Peterson, J.: Presence Information Data Format (PIDF). RFC 3863, IETF, August 2004.
4. Schulzrinne, H.: RPID: Rich Presence Extensions to the Presence Information Data Format (PIDF). draft-ietf-simple-rpid-10 (work in progress), December 2005.
5. http://www.dialexia.com/pub/products/dial_office.jsp Accessed on April 2006.
6. http://messenger.msn.com Accessed on April 2006.
7. Z. Chentouf, S. Cherkaoui, A. Khoumsi, "Service interaction management in SIP user device using Feature Interaction Management Language", NOTERE, June 2004, Saïdia, Morocco.