Privacy Enforcement Embedded in Mobile Services

Ronald van Eijk, Mortaza Bargh, Alfons Salden and Peter Ebben

Telematica Instituut, P.O. Box 589, 7500 AN, Enschede, The Netherlands

Abstract. Next generation mobile services in business-to-employee (B2E) settings put very high demands on the privacy protection features of context-aware, personalization and adaptation enabling technologies. To this end we propose a middle agent framework that allows parties to securely exchange personal or business sensitive contextual information independently of the available networks. In order to demonstrate our privacy enforcing middle agent framework, we build a scheduling service, in which the middle agents collectively arrange an update of a meeting between employees by adapting location and time on the basis of privacy and scheduling policies of the traveling employees themselves or the companies they work for. We developed and deployed this scheduling service on a LEAP agent platform and used a PDA to communicate with the middle agents on the server using WLAN and GPRS networks.

1. Introduction

Mobile services must be very highly resilient in a heterogeneous and dynamic environment, where service components are generally distributed over many physical domains –such as networks and terminals –and administrative domains. Mobile service provisioning consequently requires a sophisticated and intelligent service broker [1] that initiates, maintains and terminates such services securely and privately.

A truly generic solution to the optimization problem of a secure and private mobile and distributed service brokerage can be provided by collective intelligent agent systems [2]. Following them, we propose a functional problem-solving environment for agent-based service brokerage problems. Our problem-solving approach, however, also provides novel agent-based brokerage mechanisms to enforce the privacy of the parties involved by protecting the information and behavior of mobile end-users or their agents acting on their behalves. In this respect intermediary security and privacy issues like which information you want to share with whom and how, are as important aspects as data (transmission) security aspects. Here we focus on intermediary privacy enforcement of service brokerage by trusted third parties [1]. Mediation makes fraudulent logging, processing and tracing of user or employee behavior impossible, because such user behavior cannot be identified with that of a specific individual or employee. Furthermore, the disclosure of private or business

van Eijk R., Bargh M., Salden A. and Ebben P. (2004).
Privacy Enforcement Embedded in Mobile Services.
In *Proceedings of the 3rd International Workshop on Wireless Information Systems*, pages 42-49
DOI: 10.5220/0002669000420049

information is restricted to entrusted parties.

In order to automate privacy enforcement and business security in delivering mobile services we extend the middle agent framework proposed in Decker et al. [3]. (see section 2). The middle-agents help to locate and to connect the service provider and the service requester in a private way. We demonstrate the added value of the middle agent framework in building a scheduling service that protects privacy in a mobile B2E setting (see Section 3).

2. Privacy Protection Strategies

In its simplest form, a *sub-service* is requested by a Requester (R) from a Provider (P) that has access to the resources to deliver this service. A Requester Agent (R_A) and a Provider Agent (P_A) represent R and P, respectively. Note that such agents are not just software agents, but it might also be a piece of hardware, the user herself/himself, etc.

R_A and P_A are in possession of their own preference and capability information, respectively. An agent that deals with preference or capability information that is *neither* a requester *nor* a provider is called a *middle-agent* (denoted by M_A). Any agent (R_A, P_A or any other 3rd party agent, e.g. M_A) that is informed of *both* preferences and capabilities is in a position to make a decision. Using this approach, the privacy issues involved in sub-service brokerage can now be resolved by ensuring that only entrusted parties can access the preference and capability information.

Upon close investigation of the nine classes of Middle Agents mentioned in Decker et al. [3] (each corresponding to one specific combination of R_A, M_A and P_A being aware of the preferences and capabilities at the time of decision-making), we distinguish the following four main privacy enforcement strategies. For each strategy we will mention the most probable decision-making authority (see Fig. 1 for an illustration):

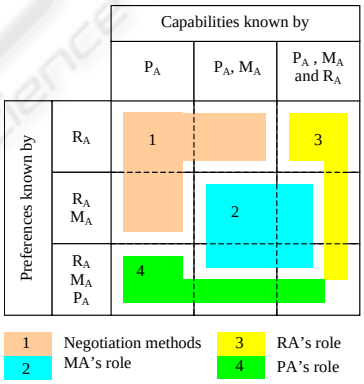

Fig. 1. Categories of decision-making strategies.

1. For strategies within region 1, none of the actors has both preferences and capabilities information at his disposal. Here we propose to use negotiation strategies that are widely studied in AI and multi-agent systems to reach an agreement. Hereby, the R_A and the P_A (or their representative M_A in the role of a front-agent or anonymizer) can withhold the sensitive information regarding the preferences and capabilities (for example, the price range that they are willing to pay for the service). Of course, as in real life, on the one hand agents may try to learn about the preferences and the capabilities of their opponents (by studying their behaviors for a long period of time). But on the other hand, each agent may do its outmost to hide such information (or deceive the opponent by its behavior).
2. For strategies within region 2, an M_A is aware of both preferences and capabilities when it is a broker, recommender, introducer, or arbitrator. As an entrusted entity, such an M_A is allowed to make a decision on behalf of the others, when acting as a broker, or to provide support, while acting in the other three cases.
3. For strategies within region 3, an R_A solely (or together with another agent) is in the position of making a decision based on full information of preferences and capabilities. This is the case when M_A has acted as a yellow pager, recommender or arbitrator.
4. For strategies within region 4, a P_A solely (or together with another agent) is in the position of making a decision based on full information of preferences and capabilities. This is the case when M_A has acted as a blackboard, introducer or arbitrator.

Note that it depends on the organizational role of an agent whether a decision-making authority will be in favor of the requester, the provider or both.

3. Scheduler Agent System

In this section, we outline how to design and implement a location aware personalized scheduling service, called Scheduler Agent System (SAS), using our middle agent framework presented in section 2. Enforcing privacy in provisioning of such B2B and B2E services is a prerequisite for successful M-commerce applications [4].

3.1 B2E scenario underlying SAS

Our SAS aims at realizing personalization, device and time-critical aspects, and location-awareness of mobile services in a common B2E setting, as illustrated by the following scenario. Assume that three employees, each from a different company A,

B or C, have scheduled a meeting in a city (service initiation phase). They all have to drive to this city to attend the meeting and when it is finished, each of them has to drive to their home or to a second meeting. However, on his way, one attendee (or some monitoring agent) listens to the traffic information on the radio, reporting about a traffic jam on the way to the meeting point. Given this situation, since he is not able to make the planned meeting on time, he activates his Scheduler Agent (SA) to rearrange the meeting somewhere else and at some later point in time (service maintenance phase). The employees work for different organizations with different security or privacy policies. Because of these policies one of the employees (thus, also the corresponding SA) may not be willing to share his current location or schedule with the others.

3.2 Implementation of SAS

The SA's can be conceived as the main enablers of the "scheduling service" in the brokerage plane that deliver the service of arranging a scheduled appointment for the employees. During negotiation phases the SA's will access and collect all required information about preferences, locations, schedules and privacy policies. This information will be taken into account during the negotiation and will be hidden from the other SA's when necessary.

System Architecture

For our implementation we used the LEAP agent platform, see [5]. Within this Java-based and FIPA compliant agent platform, agents communicate by sending FIPA ACL messages [6]. The platform has a Directory Facilitator (DF) agent where other agents can register and expose their service and functionality. Furthermore, the platform inhabits an Agent Management System (AMS) agent that takes care of all agents' life cycles. The overall platform also takes care of the communication between agents, so that local names can be used when sending messages and agents are not aware of the actual physical location of other agents.

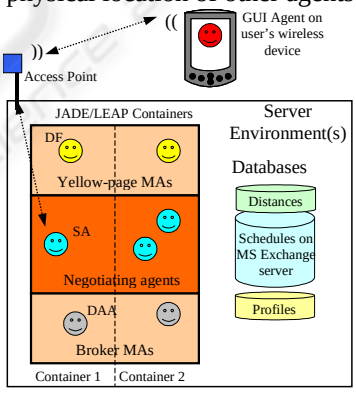

Fig. 2. Simplified SAS architecture.

One or more local servers will host, besides schedule data and profile information, the containers (runtimes) of the LEAP platform hosting all SA's. The LEAP platform hosts several agents and connects them logically, even when they run at several different physical locations (different servers and devices). In addition to the AMS and DF agents, the following types of agents can be distinguished on the platform of the SAS (see Fig. 2):

- One Scheduler Agent (SA) for every employee, located on a LEAP main container in a server machine,
- A Database Access Agent (DAA) agent on each server machine,
- One Graphical User Interface (GUI) agent for each end-device.

The SA's on the server are able to access the schedule of their corresponding users. For privacy reasons, the SA's are *not* allowed to access schedules of *other* users. In order to avoid that the SA's need to have knowledge about the way the data is stored in the database, a separate Database Access Agent (DAA) is used to facilitate data exchange between the SA's and the database. The DAA accesses the schedule of a user and his/her profile data. Harmony® for MS Exchange® was used to enable the JAVA based DAA to login, extract and update appointment information of schedules on an MS Exchange server.

Each client device (notebook or Compaq iPAQ) is a portable device that runs a LEAP peripheral container and is able to communicate over a wireless link (WLAN as well as GPRS). The peripheral agent container hosts one single GUI agent. This is basically a very simple agent, since it is only used to provide a way to interact with the user, so the user can use it to activate his/her SA on the server to cancel or reschedule an appointment. We successfully implemented the system using PDA's and notebooks connected to the server using WLAN or GPRS networks. The main negotiation functionality has been implemented into the SA's. The interaction and negotiation functionality of these agents will be explained in detail in the next sections

3.3 Middle-agents in SAS

This section elaborates on the different agents as implemented in SAS and their middle agent roles (see [3] for definitions of roles).

DF's as yellow-page middle agents

All FIPA compliant agent platforms provide yellow page services that allow agents to search for other agents and inspect the services they offer. In our platform the DF, automatically created upon platform launch, provides this functionality. The GUI agent on the device of the user that initiates the rescheduling uses the DF to contact and activate its corresponding SA. Then, this SA uses the DF to find the names of the agents that represent the other attendees of the meeting to be rescheduled.

DAA's as broker middle agents

An SA is programmed to perform a specific task and it will search for the information it needs to carry out its task. Part of the information can be obtained from schedules and databases. Therefore, the SA can access the DAA for the following information:

- Schedule of the corresponding employee's meeting times and locations,
- Travel times to the location of next and consecutive meeting.

In the current implementation the user has to enter his current position on his device manually. The DAA logs into the MS Exchange server as one of the users and extracts information from the corresponding schedule. On the other hand, user preferences are stored in databases, to be requested from and accessed by the DAA's.

The DAA is a trusted-third party in our implementation and it will refuse any direct request for schedules or locations from outsider agents. In other words, it will only access the schedule of the user that corresponds to the SA that made the schedule request (See **Fig. 3**). It will only do so if this agent provides the proper username and corresponding password required to access the schedule. The DAA has a front-agent middle-agent role on behalves of databases.

SA's as negotiating middle agents

The agent that triggers the rescheduling process, referred to as the initiator, can be considered as a R_A that requires resources (time) from the responding agents to set a meeting. The responders have to provide time and thus can be considered as a P_A. We assume that R_A and P_A (s) do neither share their strategies nor share the main part of user preferences directly. Also they do not give this information to an M_A to decide. Generally R_A (the initiator) is not aware of the capabilities (available time-space slots) of P_A (the responder) or even R_A may not be authorized to reschedule the meeting by itself. Therefore, R_A and P_A will have to negotiate (as in case 1 of section 2.1). In other words, our SA's are negotiating agents.

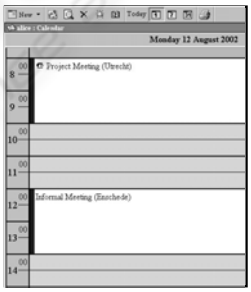

Fig. 3. Each Scheduler Agent can request today's schedule of its user from the DA agent.

Due to the key roles of SA's in realizing the SAS brokerage, this section elaborates on the details of interaction between these SA's. Based on the FIPA Iterated Contract Net Interaction Protocol (ICNI protocol), a fully functional interaction algorithm was

developed and implemented for the SA's, including the possibility to query, collect and process information from other agents. An important feature in the ICNI protocol is the distinction between *initiator* and *responder*. The initiator starts and manages the interaction. It sends a Call for Proposal (`CFP`) message, setting the conditions under which the responders would have to act after an agreement; evaluates the proposals sent back by the responders; continues negotiation by rejecting proposals or finishes it by accepting them. The responder role is assigned to all other participants in the interaction. Responders can respond to a `CFP` by defining a proposal and sending a `PROPOSE` message.

After the initiator has received the information about travel times from its own current location and about his schedule for today, it will ask the responders to supply similar information about their corresponding employees. However, because of privacy policies of other agents, the responders may not be willing to share such information. Based on possibly limited amount of information, the initiator prepares his first `CFP`. Within this `CFP` he puts a proposal including time and location for the new meeting. When the responders receive a `CFP`, they will request and process similar information about their own users in the same way.

The following steps are taken in the negotiation:

1) First Call for Proposal from Initiator.

The initiator defines a full proposal and calls for it by sending his proposal as part of a `CFP` message. Each proposal for a meeting can be described by a few parameters: subject, names of attendees, start time, end time and location. The SA's will exchange proposals by communicating to each other their accepted parameter values. This implies that the initiator calls for a possible location and asks the responders to do a proposal. For example, when it is close to Amsterdam, it sends a message like `CFP(Amsterdam?)` to all responders.

2) Proposals from Responders.

The responders can respond to a `CFP` by sending a `REFUSE` (if they refuse the proposal completely) or a `PROPOSE` message. When the proposal in the `CFP` fits their requirements the proposal to be sent back is *equal* to the proposal that was in the `CFP`, i.e. `PROPOSE(Amsterdam)`. However, when the proposal in the `CFP` does not fit the requirements, an *adjusted* alternative proposal will be sent back e.g. `PROPOSE(Utrecht)`.

3) Reject or Accept Proposals.

Because there is no interaction between the responders, most intelligence is in the initiator, who has to define a proposal in its `CFP` that will fit all other responders at the same time. This is different from a standard auction-like protocol where the responders try to bid in a smart way. The interaction is finished when all responders answer to a specific `CFP` by sending the same proposal back. Because after each negotiation step, all parties will give way (willing to accept locations requiring more travel time), at some point, most locations will be acceptable.

4) Finish and deliver service

Under normal conditions (no `REFUSE` or `NOT UNDERSTOOD` messages), the interaction is finished when all responders answer to a specific proposal in a `CFP` by sending the same

proposal back. Another situation that may finish the interaction is when all responders coincidently respond by sending the similar alternative and this alternative is acceptable for the initiator. In both cases, the initiator will send `ACCEPT` message to all responders and the agents will have to update schedules and notify the employees they represent.

3.4 Negotiation Strategies

A basic strategy has been defined and implemented where each SA mainly varies the location of the meeting in their proposals during the negotiation, i.e. the issue to negotiate about is location and travel time. The agents acts in a *competitive way* and starts to bid from the their own most preferable location with respect to travel time, even when this is not expected to fit the other users requirements at all. Thus, the responders will not immediately accept this.

4. Conclusion

Future services require dynamic binding of business and customer preferences and embedding privacy enforcement in the service provisioning process. We worked out a functional problem-solving environment for service brokerage based on an agent paradigm. Our environment ensures the privacy of the parties involved using novel agent-based brokerage mechanisms. To this end, the middle-agent framework of Decker et al. [3] is exploited in different brokerage steps enforcing service privacy. By deploying an appropriate middle-agent, privacy enforcement follows. Therewith, the middle-agents collectively realize the brokerage of the mobile service, preserving the privacy of the actors involved. A scheduling service has been developed based on privacy protecting middle agents that reschedule a meeting arranged among traveling employees of different companies. It clearly proved the feasibility of enforcing privacy in a B2E setting by means of our proposed methodology.

References

1. M.M. Lankhorst, H. van Kranenburg, A. Salden, and A.J.H.Peddemors (2002). "Enabling Technology for Personalising Mobile Services," HICSS-35, January 2002, Hawaii, USA.
2. D. Wolpert and K. Tumer (1999). “An Introduction to Collective Intelligence”, Tech Report NASA-ARC-IC-99-63; In: Jeffrey M. Bradshaw, editor, Handbook of Agent Technology, AAAI Press/MIT Press, 1999.
3. K. Decker, K. Sycara and M. Williamson (1997). “Middle Agents for the Internet”, Proceedings of the 15th International Joint Conference on Artificial Intelligence, Nagoya, Japan, August 23-29, 1997.
4. P. Langendoerfer. (2002). “M-Commerce: Why it does not fly (yet?)”, SSGRR2002, L’Aquila, Italy, 2002.
5. F. Bergenti and A. Poggi (2001). “LEAP: a FIPA Platform for Handheld and Mobile Devices”. Presented at ATAL 2001, http://leap.crm-paris.com/
6. FIPA, Foundation for Intelligent Physical Agents, http://www.fipa.org/.