Agent-based Privacy Enforcement of Mobile Services

Mortaza S. Bargh, Ronald van Eijk, Peter Ebben and Alfons H. Salden,

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

***Abstract*—Next generation mobile service delivery puts very high demands on context-aware and personalization enabling technologies. Context-awareness and personalization of our Personalized Service Environment (PSE) [7] is achieved by the integration of a complex set of those technologies imposing the constraints of different entities such as wireless and fixed networks, end-devices, services, users and businesses. Our PSE concept heavily hinges on a so-called "brokerage system" that controls and manages the delivery of future mobile services [8]. In this paper we investigate additional brokerage functionality that is needed to enforce privacy of the parties involved in delivering mobile services. To this end, a mobile service can be considered as an ensemble of so-called sub-services, where each sub-service is controlled and managed by a "brokerage sub-system". Our study of middle agents for the Internet [9] yields a specification of a "brokerage sub-system" in terms of how it preserves privacy of the actors involved in requesting and delivering the corresponding sub-service. A well-designed combination of these brokerage sub-systems can enforce context and user depending privacy requirements. We build a context-aware personalized scheduling service for a mobile business-to-employee (B2E) setting, where software agents collectively arrange new meetings at different locations and times keeping in mind the upcoming meetings of the employees. The software agents also simultaneously look after privacy or security policies of the employees or their companies, e.g. with respect to location information, time schedules, personal preferences or business sensitive information. We developed and deployed our scheduling service on the JADE-agent platform using PDA's and small notebooks connected to a server using WLAN and GPRS networks.**

***Index Terms*— privacy, scheduling service, service brokerage, software agents.**

I. Introduction

Mobile services must be very resilient in a highly heterogeneous and dynamic environment that encompasses different and varying networks, terminals, locations, user preferences, etc. Normally the service delivery in this environment is realized by service components distributed over many physical and administrative domains. Concerning mobile service delivery then several issues have to be resolved such as:

- Which sub-services are needed in a mobile service?
- Where can we find providers offering these mobile sub-services?
- How can we schedule these mobile sub-services?
- How can we integrate sub-services, taking into account the existing constraints imposed by devices, networks, requesters and providers?
- How can we adapt a mobile service to the dynamics of the mobile environment?
- How can we learn from current experiences in order to make the future mobile services more efficient?
- How can we react proactively to future mobile service conditions? Or how can we anticipate future brokerage problems?

To settle these issues, mobile service provisioning require a sophisticated and intelligent system to initiate, steer and terminate the service such that it is acceptable to all actors. This is a very challenging problem that the telecommunication research community still has to face and solve.

In this paper we elaborate on a framework in which these issues are settled by a so-called *brokerage system.* In a mobile service environment the service components and resources are spread over different administrative and physical domains. In this environment the brokerage system must gather information about the capabilities and preferences of all the actors that request and provide the sub-services of a mobile service. Each sub-service in the life-cycle of a mobile service, in turn, is enabled by a set of control and management operations carried out by a so-called *brokerage sub-system.* We focus on the privacy protection functionality of these brokerage sub-systems that are primarily based on latest software-agent related technologies and research. These brokerage sub-systems enforce privacy through multi-agent negotiation means in every stage of the brokerage process. Depending on the level to which each actor is willing to share his capability and preference information with others, an appropriate brokerage mechanism can be used to enforce the privacy

concerns. We will illustrate the concept by describing the implementation of a scheduling service in a mobile B2E setting. By implementing these brokerage mechanisms into autonomous, negotiating software agents, a service was built that is capable of protecting employees' privacy for scheduling of meetings.

The main goal of mobile service brokerage is the steering of the required mobile sub-services in such a way that all resources are optimally managed and used. This issue has extensively been studied in the area of Quality of Service (QoS) management for distributed multimedia applications [2][3]. In this and many others areas such as commerce, trade and network bandwidth management brokers are enabling the above optimization. Contrary to those other areas that limit their scope of the broker to a particular service aspect within a particular business, network, and application area, we define the functionality of a service independent of service characteristics. Instead we focus on generic service delivery and how to deal with information flow and privacy aspects.

Our paper is organized as follows. Section II elaborates on the problem of next generation mobile services, summarizes work related to it and points out the contributions of our work. In Section III we categorize brokerage sub-systems from a privacy perspective. Section IV describes in detail a personalized scheduling service in a mobile B2E setting, which settles privacy issues by means of negotiating software agents.

II. Background and Related Work

Nowadays, mobile services should be adapted in order to meet the very dynamic and diverse range of user requirements. In this context the adaptability of generic services is getting recently more and more attention, for example, to ensure their accessibility by all users and to satisfy these users when using such services tailored to their needs. These forms of service adaptation could be performed by systems, by end-users, or by a combination of both. User initiated adaptation (or "customization" according to [15]) has, however, many limitations, especially in the case of mobile services where context changes are considerable and cannot be resolved in time manually. Furthermore, a mobile user has normally very limited (human-computer) interaction capabilities on his end-device to adapt services. Another problem in mobile service provisioning is persistency of the service when many interactions have to take place between user and service provider to define a service request and to define the proper service that matches the request.

We adopt a solution to the above mobile service adaptation problem that is based on communicating the service adaptation (brokerage) issues mainly in a fixed instead of a wireless environment. We use the term "personalization" to denote both user and system initiated service adaptation concepts. Services become personalized when they are (dynamically) tailored to usage contexts. Usage context in this sense consists of many aspects, like the needs of an end-user; location-related aspects (e.g., physical co-ordinates and velocity, ambient conditions); technical aspects (e.g., network bandwidth and capabilities of the terminal); business rules that apply, etc. In addition to physical co-ordinates and velocity, the location related aspects involves environmental and ambient conditions, like for instance indoors/outdoors, humidity, temperature, etc, and activity status, like for instance working, meeting, walking, driving, etc.

In the GigaMobile project [25] we use a Personalized Service Environment (PSE) to adapt mobile services [7]. In our PSE users are allowed to roam between different (wired and wireless) access networks and receive services according to their personal profiles and contexts. In this respect we are merely concerned with delivering mobile services between enterprises and their employees. The employees interact with peer-to-peer as well as client/server services via different terminals. In this B2E setting we identify five types of actors each imposing their own mobile service requirements. These actors are employees (having personal preferences), end-devices (having limited capabilities), network operators and service providers (having limited capabilities) and business or organizational managers (determining or affecting access to resources by enforcing their policies).

In [7] and [8] we discriminated between two planes to deliver personalized mobile services in our PSE, namely, a service plane and a brokerage plane. The latter plane takes into account the constraints imposed by all actors mentioned above (see Figure 1). The distinction between service plane and brokerage plane is also made for signals carried in different OSI layers (see for example [26]), where it is often known as the difference between data plane and control plane. The data plane carries data and supports control signals of higher OSI layers. Here we take a user–centric perspective with respect to the difference between service and broker planes. What is stored and flows in the service plane is what a user expects from the service (e.g., content data, arranging an appointment, etc). The service plane thus contains system components that store, forward and adapt the data units and logic for delivering mobile services to the end-users. The brokerage plane, on the other hand, contains system components that determine a specific launching of service plane components. The brokerage plane obtain requests, profiles and status (evolution) of actors and then issue the storage and flow controls affecting the service plane components (ensuring therewith dynamic service binding) [8].

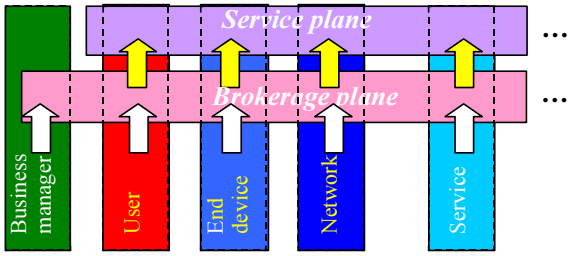


Figure 1: Our reference model for a PSE

A service brokerage system in a PSE carries out the following functions to realize a mobile service delivery (see also [8]):

- Integral representation, retrieval and updating of actor profiles,
- Service initiation by matching and finely tuning the requests and offerings in order to optimize overall system performance.
- Service maintenance by driving deductive or inductive inferential structures to enable a broker to monitor the service, to adapt the service on the fly, and to learn and anticipate.
- Service termination, by releasing the resources.

The PSE brokerage functions during a mobile service *delivery*, i.e., the last three functions mentioned above, correspond to three generic brokerage phases, namely: service initiation, service maintenance and service termination. These phases are based on those of QoS management for distributed multimedia applications. This is abstracted from observing a strong correlation between the QoS management topics for real-time multimedia applications and the brokerage topics involved in next generation mobile services. For example:

- In both areas, the preferences and capabilities of the actors are represented by meta-data or profiles to reach a settlement.
- Like in real-time streaming applications, in a PSE the predefined conditions and constraints do not remain valid through the whole lifecycle of the service due to dynamically changing locations, time, device, network load, user preferences, etc.
- It is necessary in both cases to monitor (the status of) the service during the delivery time and to adapt the service dynamically to rising conditions.

The goal of the brokerage system is to use available resources effectively and efficiently in the sense that all actors are satisfied with the mobile service given their contexts. This optimization problem has not yet been resolved in distributed system theory. Nevertheless, the classical research in the area of QoS management has produced many solutions in terms of frameworks and standards [2], [3]. Centralized and client-server resource management schemes [4] and QoS brokers [6] are common solutions to the optimization problem mentioned above.

Another solution to the optimization problem of mobile and distributed service brokerage can be provided by collective intelligent agent systems [1]. Following [1] we proposed in [8] a functional problem-solving environment for service brokerage based on such an agent paradigm. In distributed multimedia applications such an agent-based approach has also proven to support effectively service brokerage in terms of robustness, flexibility, openness and resource optimization [4]. However, as observed above, our paradigm ensures that novel brokerage mechanisms are put in place to enforce dynamic service binding mechanisms. For example, in order to dynamically enforce privacy issues on the service plane, we could deploy appropriate service brokerage measures in the brokerage plane. For this reason, in the next section we categorize the behavior of agents in brokering sub-services with respect to privacy issues analogous to that categorization process carried out for information-gathering organizations in [9].

In section III we make the roles of the privacy enhancing agents in brokerage sub-systems explicit. In section IV we will describe a detailed implementation of a related scheduling service for a mobile B2E setting. In this setting traveling employees of possibly different companies might share a distributed schedule. The brokerage system of such a scheduling service could control the exchange of schedule information and the rescheduling of meetings according to the wishes of the employees and the policies set by their companies. In contrast to existing scheduling services such as Outlook (or even those based on agent technology [12]), our service aims at protecting privacy or business sensitive information. Instead of revealing schedules and preferences to all actors, and then let these actors determine proper time and location slots for a meeting, we automate the negotiation phase between the actors keeping in mind their privacy concerns. Our actors are not aware of privacy concerns of the others, but their representative agents deal with these issues using multi-agent negotiation strategies. Note that these agents are not aware of the negotiation strategies adopted by the others.

III. PROTECTING PRIVACY

Considering the lifecycle of a mobile service, we can group the activities of its brokerage system in three generic phases, namely: service initiation, service maintenance and service termination, similarly to those in QoS management of distributed multimedia applications. Like most real-time control, communication, and information systems, also our PSE brokerage system is complex in nature. This complexity calls for decomposing a mobile service to a set of sub-services that jointly deliver a privacy-enhanced mobile service. To this end, each sub-service is controlled and managed by a

brokerage sub-system. The brokerage-subsystems corresponding to the sub-services of a service steer the delivery of the mobile service collectively. How to decompose a service to its sub-service combine will be illustrated by a case service in Section IV (a B2E scheduling service). In this section we elaborate on specification of a "brokerage sub-system" in terms of how it preserves privacy of the parties involved in delivering the corresponding sub-service and we explain the strategies for making decisions in "brokerage sub-system".

A. *Brokering Sub-services*

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 the service. This sub-service for example enables communication (messaging, phone calls, etc) or the allocation of resources (network bandwidth, time, facilities). The data or service flows live in the service plane, while control or brokerage flows live in the brokerage plane. In the brokerage plane, 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. This model of a sub-service is illustrated in Figure 2.

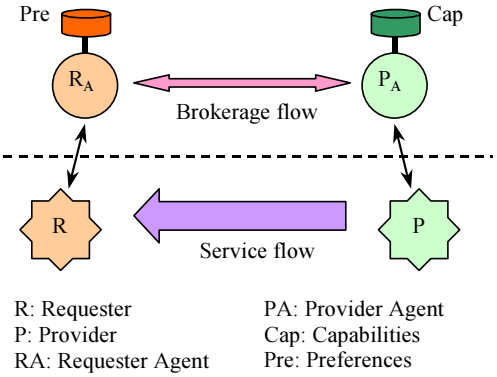


Figure 2: The model of a sub-service.

R_A and P_A are in possession of their own preference and capability information, respectively. To establish the service, the capabilities and preferences should somehow be compared and matched in the brokerage plane. The preference information can for this reason flow from an R_A to a P_A, and the capability information can flow the other way round. Any agent (R_A, P_A or any other 3rd party agent) that is informed of *both* preferences and capabilities is in a position to make a decision on the type of storage and flow in the service plane, i.e., on the service to be delivered. Using this approach, the privacy issues involved in sub-service brokerage can now be solved by ensuring that the preference and capability information is only accessible to entrusted parties.

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). For privacy purposes, the flow of information can be stopped at different points in the brokerage plane by R_A, P_A and M_A as shown in Figure 3, resulting in different organizational structures for solving the brokerage problem of a sub-service. Preference information can initially be kept private to R_A, be revealed to some M_A or be known by P_A. Similar reasoning can be held for capability information.

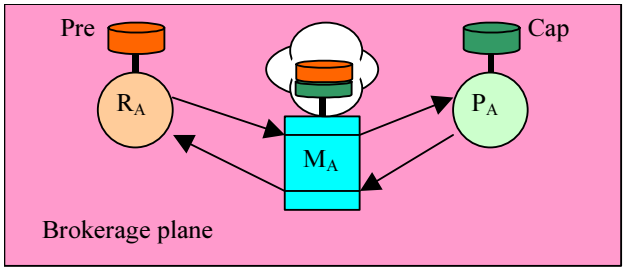


Figure 3: Flow of information in the brokerage plane of a sub-service.

We assume that the preference information is handed over from R_A to M_A and from M_A to P_A, and the capability information is handed over from P_A to M_A and from M_A to R_A, as illustrated in Figure 3, by a *brokerage action*. Depending on agents R_A, M_A and P_A having this information before and after a brokerage action, nine general middle-agent roles can be identified (analogous to the approach of [9] for information-gathering organizations) as illustrated in Figure 4, namely:

1. Where there is no M_A involved, R_A or P_A (or both) could broadcast their preferences or capabilities to the whole community. Consequently, either the requester informs the provider of the preferences or the provider informs the requester of the capabilities. After such a brokerage action, either P_A or R_A (or both), respectively, could be aware of both capabilities and preferences and therewith devise a solution. When R_A or P_A broadcast their knowledge about preferences or capabilities, the whole community can be aware of this information, which may not be desirable from a privacy perspective. To limit the extent of private information dissemination or to hide the identity of R_A or P_A, there is a need of an M_A to facilitate the brokerage process. The following brokerage actions reflect the roles that such an M_A may have.
2. *Front-agent*: P_A informs M_A of its capabilities in order for M_A to deal with R_A on P_A's behalf.
3. *Yellow-pages (matchmaker)*: P_A has already informed M_A of its capabilities. M_A shares this information with R_A in two ways: either R_A asks M_A for it or M_A forwards it to R_A in a deterministic way (if R_A is subscribed).
4. *Anonymiser*: R_A informs M_A of its preferences in order for M_A to deal with P_A on R_A's behalf.
5. *Blackboard*: R_A has already informed M_A of its preferences. M_A shares the information with P_A in two

ways: either P_A asks M_A for it or M_A forwards it to P_A in a deterministic way.

6. *Broker*: Both R_A and P_A inform M_A of their preferences and capabilities, respectively. M_A holds this information for itself and especially does now share preferences and capabilities with P_A and R_A, respectively.
7. *Recommender*: Both R_A and P_A have already informed M_A of their preferences and capabilities, respectively. M_A shares the capabilities with R_A.
8. *Introducer*: Both R_A and P_A have already informed M_A of their preferences and capabilities, respectively. M_A shares the preferences with P_A.
9. *Arbitrator*: Both R_A and P_A have informed M_A of their preferences and capabilities, respectively. M_A shares the capabilities and preferences with R_A and P_A, respectively.

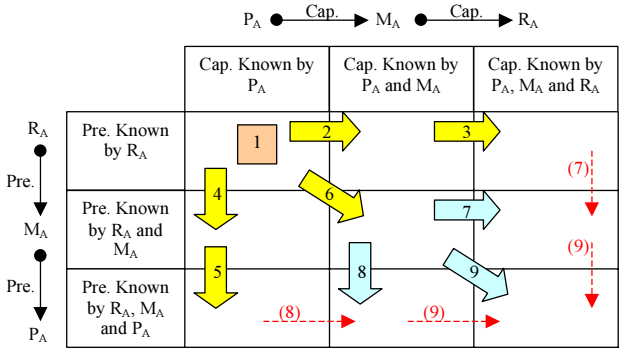


Figure 4: Roles of a middle agent in brokerage plane of a sub-service.

The brokerage actions mentioned above assume that the requester and provider agents are aware of the M_A. For example, in a pure brokered organization all the agents generally know of the whereabouts of an MA. In an open system, however, hybrid brokered organizations use a matchmaker allowing providers and requesters to find an appropriate middle agent. Hybrid organizations can make use of anonymizer or front-agent M_A's to protect both requesters and providers from (security and) privacy infringing agents.

B. Categorization of Decision Making Strategies

A solution to a privacy enhancing service brokerage takes a proper combination of sub-service. One can distinguish two aspects in each brokerage step corresponding to a sub-service. The first aspect concerns the selection of methods applied in decision-making. For example, the inference schemes and criteria on which a "broker middle agent" decides to assign requests to providers come from several areas of research, including load balancing, enterprise resource planning, and market-based economics. The second aspect concerns determining who is entitled to make the decisions. Apparently anyone aware of both preferences and capabilities is in a better position to decide.

With a close investigation of the nine classes mentioned in the previous paragraph (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 four main decision making strategies. The second aspect regarding the best decision-making authority is briefly touched upon in the following strategy categorization and in Figure 5.

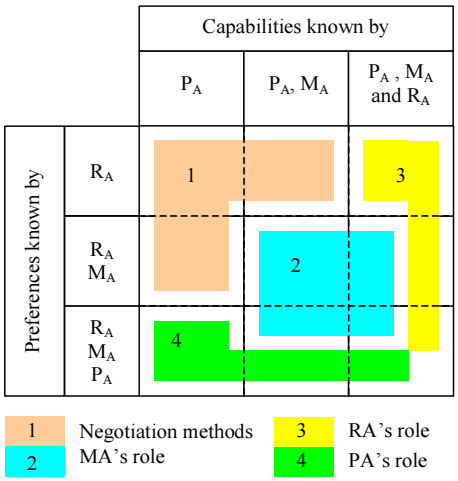


Figure 5: Four categories of strategies of decision-making.

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 anonymiser) can withhold the sensitive information regarding the preferences and capabilities. For example, such sensitive information can be the price range that they are willing to pay or to get 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, and 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.

Depending on its organizational role, an agent makes a decision in favor of the requester, the provider or both (i.e., being fair). This dependency influences the choice of a brokerage action. Another problem to handle is the effect of dynamically changing preferences and capabilities. A capability change over domains implies the entry or exit of requesters/providers. The ability of an organization to quickly adapt to new preferences and capabilities is in addition a function of the distance that the information has to travel, and the costs of keeping that information up-to-date. This may also influence the choice of a brokerage action.

IV. Privacy Enhanced Scheduler Agent System

In this section we outline our activities in designing and implementing a location aware personalized scheduling service using agent technology. The brokerage system of this so-called Scheduler Agent System (SAS) enforces users' privacy, an important issue impeding the success of M-commerce [24]. SAS schedules a meeting for employees keeping in mind the contexts of the attendees. SAS is enabled by software agents for negotiating resources like time and location, which are aware of preferences and schedules of the employees.

In the following paragraphs we describe: a scenario on which SAS is based, how SAS follows the different service brokerage phases, how SAS is implemented using software agents, and what our practical results and experiences are in building SAS. Moreover, we emphasize in particular the different roles of agents, privacy issues and service brokerage actions.

A. Privacy and Security Policies in SAS scenarios

Our SAS was especially designed to realize personalization, device and time-critical aspects, and location-awareness of mobile services in line with a common B2E setting. Let us briefly elaborate on this real-life situation.

Assume that three employees, each from a different company A, B or C, have scheduled a meeting in a city. They all have to drive to this city to attend the meeting and when it is finished, each of them has to drive 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 sometime later.

Now this SA will negotiate with the SA's that represent the other employees over location and time of the new meeting keeping in mind the current locations of their employees on the road, employees' preferences and time constraints for traveling (taking into account the travel time to their homes or some other second location). The employees work for different organizations with different security or privacy policies. Because of these policies one of the employees or corresponding SA may not be willing to share its current location or schedule with the others. After some negotiation rounds, the users receive notifications over the rearranged meeting on their wireless devices.

B. Service brokerage phases of SAS

In Section III we distinguished three brokerage phases in the lifetime of a mobile service, namely service initiation, maintenance and termination. Here we describe these phases of our scenario presented in the previous paragraph.

In the service initiation phase, an employee requests a meeting with a group of people. This user event triggers activation of the SA's (of the attendee and other required resources) in the broker plane. These SA's negotiate over different issues such as the location, the day, the starting time, the ending time (or the duration), taking into account the preferences of the users. When an agreement is reached, the appointment is stored in the scheduling database (reservation of resources, i.e. time and facilities) and a notification is sent to the users. In the service plane, the system delivers the data about a meeting to the corresponding attendees. In the brokerage plane all measures are taken to arrange the meeting.

In the service maintenance phase, the status of the appointments is monitored with respect to the contexts of the attendees by the users themselves or some agents on their behalves). An initially agreed upon appointment might not be acceptable anymore later because of a change in context, e.g. a traffic jam and delay. If a user cannot make the scheduled meeting, the service broker triggers a rescheduling process (initiated by the user or an agent on his behalf). When an agreement is reached, the rescheduled appointment is stored in the scheduling database (resource and time reservation) and a notification is sent to the users.

The service termination phase is not relevant in this service. As soon as a meeting finishes, resources are freed automatically.

Service initiation as well as maintenance can be triggered

and controlled by different types of events. The agents in the control plane responsible and active in those phases can be triggered by user related events (employee manually requests meeting with a group of people or cancels a meeting) or by context related events (changes in circumstances like location unavailable or traffic jam). The position of a member of a contact list of an employee can be monitored and used to establish an ad-hoc meeting when both are in each other's vicinity.

C. *Implementation of SAS*

Our SAS has to be able to reschedule a meeting and notify the traveling employees about the new location and time of the meeting. When we use our scenario as a basis for our SAS, note that one can conceive the Schedule Agents (SA's) as the enablers of a "scheduling service" in the brokerage plane that deliver a service, i.e. arranging a scheduled appointment for the employees. During negotiation in the service initiation and maintenance 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.

1) *System Architecture*

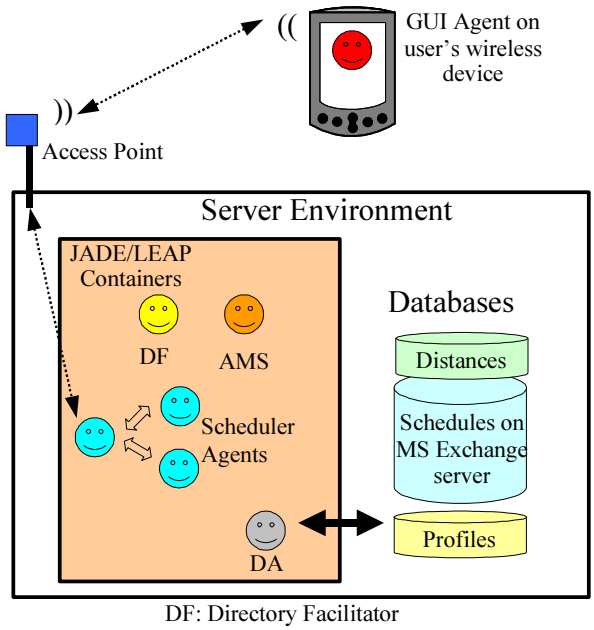
For our implementation we use the JADE-LEAP [20][22] agent platform. Within this Java-based and FIPA [17] compliant agent platform, agents communicate by sending FIPA ACL [18] messages over a TCP/IP connection between different runtime environments. It has a Directory Facilitator (DF) agent where other agents can register and expose their service and functionality. Furthermore, it 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. The JADE-LEAP platform uses agent behavior models for the tasks that an agent might perform. The agents instantiate their behaviors (as threads) according to the needs and capabilities required for performing their task [20].

One or more local servers will host, besides schedule data, and profile information, the containers (runtimes) of the JADE-LEAP platform hosting all SA's. The JADE-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 Figure 6):

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

The DA agent accesses the schedule of a user and his/her profile data. Harmony® [16] for MS Exchange® was used to enable the JAVA based DA agent to login, extract and update appointment information of schedules on an MS Exchange server.

Each client device (notebook, Sony VAIO or Compaq iPAQ) is a portable device that runs a JADE-LEAP peripheral container. The peripheral agent container hosts one single agent, namely the 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 small notebooks connected to the server using WLAN and GPRS networks. The main negotiation functionality has been implemented into the Scheduler Agents. The interaction and negotiation functionality of these agents will be explained in detail in the next sections



DF: Directory Facilitator
AMS: Agent Management
DA: Database Access

Figure 6: Simplified system architecture of SAS.

2) *Basic Interaction Algorithm*

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 not share their strategies and the main part of user preferences directly. Also they do not give this information to an M_A to decide. On

the other hand, user preferences are stored in databases, to be requested from and accessed by the DA's. The DA is a trusted-third party in our implementation and it will refuse any direct request for schedules or locations from outsider agents. 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 Subsection III-B).

Based on the FIPA Iterated Contract Net Interaction Protocol (ICNI protocol, see [11]), 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 (see [11] for definition of ACL performatives), setting the conditions by 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.

In our case, the interaction algorithm has to deal with one important additional condition: both the initiator (who reschedules the meeting) and *all* responders have to agree on the same conditions (settings of the meeting). This implies that at some point in the negotiation, i.e. in response to one specific CFP, all responders have to send the same proposal to the initiator. This will never happen when the initiator just sets general conditions in a CFP. Instead, it has to define a full proposal in the CFP and the responders must have the choice to propose the exact similar proposal. The initiator finally has to send out a CFP that results in the different responders sending back the similar proposal. Based on these conditions, an interaction protocol has been developed that enabled successful negotiations and reaching always agreements in finite time (no infinite loops possible). The steps defined and developed for this protocol will be described in the next paragraphs, for the monitoring and adaptation phase (rescheduling).

3) Definition of a Meeting Proposals

The objective of the negotiation is to reschedule a meeting. 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. The main issue here is the location, because it determines the travel times of the employees. All agents have to agree on what a meeting is and have to construct proposals in the same way (notably not their strategies have to be the same). This proposal is wrapped in an ACL message, so the other agents can extract and understand its content.

4) Information Collection

An SA is programmed to perform a task and it will search for the information it needs to do its task. For network and memory efficiency reasons, an agent will not have all information at all time available. Thus he will only query relevant information at the proper time. The initiator will collect information of its own employee first, namely:

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

In the current implementation the user has to enter his current position on his device manually. The DA agent logins in the MS Exchange server as *one* of the users and extracts information from the corresponding schedule (see Figure 7). Note that it will only access the schedule of the user that corresponds to the SA that made the schedule request (See Figure 6). Also note that the agent will not share this schedule information with other agents. In other words, the DA has a broker middle agent role that preserves privacy.

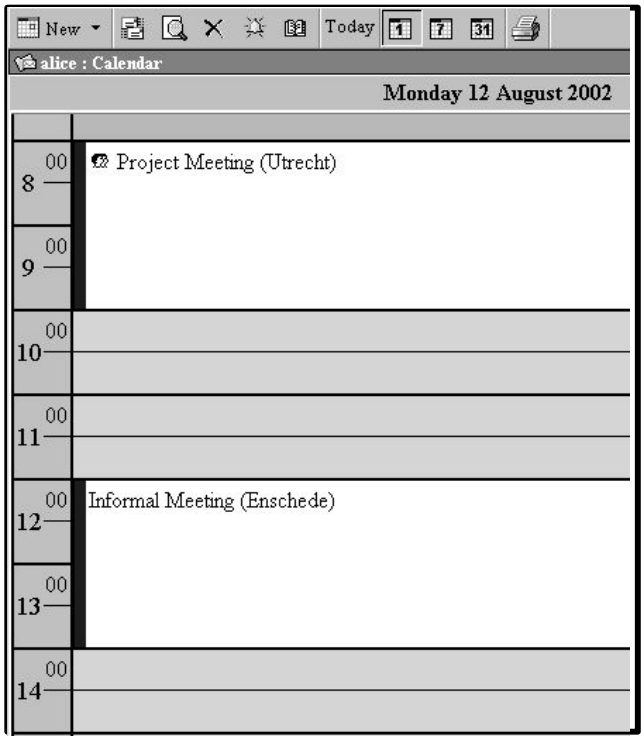


Figure 7: Each Scheduler Agent can request today's schedule of its user from the DA agent. Times and location of a possible second meeting are taken into account when negotiating about rearranging the first meeting.

Instead of asking the complete travel time matrix containing the distances to and from all possible locations, the

DA will request two lists of travel times: one to be generated and based on his current location and one to be generated and based on his second destination (see Figure 8 for an explanation). Together, these two lists contain the travel times to all other relevant locations along the route and the SA can sort this list by travel time. This information is requested from the DA agent according to the FIPA query interaction protocol [19].

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.

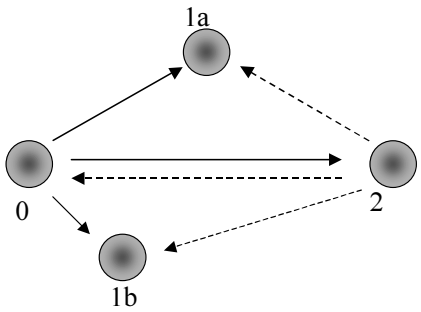


Figure 8: The user is at current location 0 and its final destination is at location 2. To calculate all possible total travel times, the agent of this user only requires the list of travel times from locations 1a, 1b and 2 to current location 0 and a list of travel times from final destination 2 to locations 0, 1a and 1b. When n is number of other locations, the agent has to query 2n travel times and not n^2.

5) 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. This implies that the initiator calls for a possible location and asks the responders to do a proposal. If the initiator has gathered the locations of the responders, it can follow a cooperative strategy by suggesting a central location. However, in order to illustrate privacy aspects during negotiation we assume that those locations are not available. Moreover, we assume that the initiator like the responders will follow a competitive strategy by proposing preferably his current location. For example, when it is close to Amsterdam, it sends a message like `CFP(Amsterdam?)` to all responders.

6) Proposal 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)`. Note that the agents in a responder role are *not* allowed to send `REJECT` or `ACCEPT`. Because the initiator is in control of the interaction, only he has the right and possibility to send `REJECT` or `ACCEPT` messages (see FIPA ICNI protocol [11]).

7) 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. When other issues have to be included, a scoring function for each issue has to be developed and the rating function is a weighed sum of these scores. Matos and Sierra [23] have shown how this approach can be used to define strategies. In our present case, each SA has to be able to determine whether a location as part of a meeting proposal fits its employee's requirements and schedule (user has to be able to arrive at least at a second location after this meeting in time). Therefore, a rating has to be assigned to each possible location, where Rating = 1 for the user's current location and Rating = 0 for each location that implies too much additional traveling time for the user. To assign a rating to each specific location in a proposal, the agent has to have a normalized rating function.

When the negotiation starts, each agent can choose for two tactics. These correspond to two different scenarios with respect to privacy. When agents are *cooperative*, they will share information, i.e. give their location away in the beginning of the interaction. The initiator agent can then base location and times in his proposal on an optimization for all participants and offer a location that probably fits others' requirements. This means he starts to propose a location with a rating value < 1, i.e. he gives way in the negotiation. In the other case, when the agents act in a *competitive way*, the agent starts to bid from the location with rating = 1, i.e. they start to offer its 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. This is called a stubborn strategy.

The initiator has to start the negotiation by sending his proposal inside the first `CFP` message. It depends on the availability of information about other employees' locations whether he will do a cooperative offer by proposing a central location or a competitive offer by proposing his own current location.

8) Reject or Accept and End Negotiation

The initiator will evaluate whether all responders did propose offers similar to the `CFP` or, by coincidence, whether they all sent a similar alternative. Otherwise, the initiator will send a `REJECT` to both, e.g. `REJECT(Amsterdam)` and `REJECT (Utrecht)`. He will *not* accept the Amsterdam proposal yet, because all parties must agree on one and the same offer. Then he will prepare and send a new `CFP`, taking into account his own list of cities and distances, but also taking into account alternative proposals that were sent back. In other words, with every negotiation step the initiator agent obtains more information about preferences of the responders and, therefore, makes a better proposal. 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 further travel time), at some point, most locations will be acceptable. In our example, a second call, `CFP(Utrecht?)`, may be responded by `PROPOSE(Utrecht)` messages from all responders. In that case the initiator sends the `ACCEPT(Utrecht)` message to these responders. This implies that initiator and all responders agree that they have a mutual contract and have to act according to it. In this case they have to update schedules and notify their own users.

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. Or when all responders 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. This implies that all agents agree that they have a mutual contract and have to act according to it. In this case they have to update schedules and notify their own employees. The agents will do this by sending `REQUEST` messages to the database agent to update all necessary information. When this is done successfully, responders have to send an `INFORM` message to the initiator to inform him that the action is performed successfully. After this final message the interaction protocol is finished. When the level of autonomy given to the scheduler agents is high, the employees do not have to confirm this new appointment.

Figure 9: Alice gets a message that location and time of her project meeting have been changed.

9) Results

A SAS has been developed based on software agents that reschedule a meeting scheduled between traveling users. An interaction algorithm has been developed and successfully implemented into agents that were executed on a JADE-LEAP agent platform connected to real wireless devices. Different runs were done on the agent platform, setting different locations, different schedule scenarios and different user preferences. In all cases the agents negotiated successfully and all actors finally agreed on rescheduling or canceling a meeting.

V. CONCLUSION AND FUTURE WORK

In line with our PSE we represented actors involved in service delivery by agents in the brokerage plane. The PSE system reaches a satisfaction level when all those agents, and thus their actors they represent, are pleased with the proposed settings of mobile services in the service plane. In [8] these satisfaction levels are interpreted as “acceptable QoS levels for all actors”. Privacy of preferences and capability information of actors is an important issue in brokering next generation mobile services. Based on our agent paradigm, we presented taxonomy of different privacy enhancing brokerage sub-systems that can be deployed in brokering of sub-services. We implemented and demonstrated a travel time-aware mobile scheduling service for employees on the road that want to adjust their planned meeting. The software agents control in that context service initiation, management as well as termination depending on the privacy policies imposed by their corresponding employees.

In our opinion development and deployment of software agents can be done on different platforms and thus be kept separated [8]. The developer can concentrate on developing the functionality of the agent (its behavior and intelligence). It is important to note that the amount of intelligence required to successfully deliver a mobile service depends on the complexity of this task. The complexity strongly relates to the amount of information available by each agent, i.e. the level of sharing of preferences. Note that in our scenario, the employees can partly regulate this.

Our scheduling service demonstrates the brokerage aspects and the roles of the agents very well. It illustrates the different phases to deliver a service and the different types of brokerage methods that can be involved. It especially illustrates how one can deal with private information in different ways and how agreements and services can be realized, even when information is kept private or secret. Our SAS is flexible and scalable and has the following features:

- User control. This control is enforced as SA represents a specific user, has access to the users schedule and knows its current location. It does *not* have to share its information with the other SA's to find a solution (a rearranged meeting). The more intelligence is used in the strategies of the SA's, the less information they have to reveal to others and the better privacy is preserved.
- Flexibility. Although the users are from different companies, it would still be possible to arrange a meeting, even when they were using different schedule services or applications. The DA agent takes care of translation between schedule information and proposed meetings to be offered to others. This is possible because they share a common language (ACL), a common interaction protocol (the meeting interaction protocol) and a common scheduling ontology (definition of a meeting proposal). A direct interaction between different scheduling applications without the use of intermediates like agents is very cumbersome if not impossible. However this translation is not straightforward and the choices for specific scheduling or database applications may limit the application independent nature of any agent system.
- Resilience or persistence of mobile services. Meetings can be arranged even when connection to user is lost or slow. Current, traditional wireless networks are still hampered by long latency for connection set-up, data flow corruption, disconnection, low available bandwidth and unstable quality. Using agents the data load and flow, and allocated logics on the networks can be dynamically brokered and spread on behalf of the different actors. This persistency due to autonomy of agents does not require constant connection with a home-base to carry out tasks on behalf of an actor.

For the traveling employee scenario only a few parameters were required to describe the employee's preference, namely a parameter describing the extra travel time it is willing to make and a privacy parameter determining whether the user is willing to show information. For scheduling meetings on long-term, more parameters are required, including parameters that describe preferences for specific day in a week or specific time (morning) in a day [8]. A more complex algorithm in which these issues and utility functions are used has already been developed and applied successfully [9]. These functions should be described as utility functions and stored in user profiles. At this moment all functions related to a user's personal preferences have no relation with real user behavior. To develop a function and a proper interaction between real users and our scheduling system, some user behavior experiments need to be conducted. When more profiles are required, e.g. service profiles, service–specific user profiles and terminal profiles, a more elaborate profile management system has to be used and accessed by our agents.

In future work, location-aware functionality will be added to a similar scheduler service as described here, but supporting a Business to Employee (B2E) setting at the site of a company. Such a location-aware service will be enabled by different terminals connected to (wired and wireless) core networks in a business area (W-LAN hot spots) or Bluetooth. Such a location aware scheduling service could automatically arrange short and spontaneous meetings between people as soon as the system spots them as being present in the same building. Or it can monitor the location of the user and notify them when they cannot make it to the location of their scheduled meeting.

By implementing a Scheduler Agent System (SAS) we showed how a mobile distributed scheduling problem can be solved using agent technology. The SAS delivers a service to traveling users with different preferences. Furthermore, our scenario demonstrates personalization, device and time-critical aspects and location-awareness of mobile services and shows how arranging or adjusting meetings can improve B2E workflows in an ad-hoc way. All together, we have shown that the SAS, connecting users equipped with small wireless devices, can bring the right people together at the right time and at the right place.

The approach we have taken here, i.e. using agents to represent users and using negotiation as a way to adapt services and to preserve privacy, can also be used in Business to Customer (B2C) settings. Soon, WLAN hotspots and Location Based Services will be available and will be offered on the fly. Examples are tourist services, and shops offering their services to customers when they are near. Langendoerfer [24] mentions that it seems feasible that low cost services will be successfully deployed in hot spots such as shopping malls, train stations and airports since here airtime will be free or at least extremely cheap. Such services require dynamic binding of business and customer, which can be handled by brokerage sub-systems (e.g. agents) in the same way as was described in

this paper.

REFERENCES

[1] 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.

[2] A. Campbell, C. Aurrecoechea and L. Hauw (1995), "Architectural Perspectives on QoS Management in Distributed Multimedia Systems", PROMS'95, Salzburg, Austria, October 1995, pp. 274-283.

[3] C. Aurrecoechea, A.T. Campbell, and L. Hauw (1998), "A survey of QoS architectures", Multimedia Systems, Springer-Verlag, June 1998, pp.138–151.

[4] L.A. Guedes, P.C. Oliveira, L.F. Faina and E. Cardozo (1997). "QoS Agency: An Agent-based Architecture for Supporting Quality of Service in Distributed Multimedia Systems", IEEE conference on protocols for multimedia systems, multimedia networking (PROMSMmNet'97), Santiago, Chile, Nov. 1997.

[5] A. Hafid, G. von Bochmann and R. Dssouli (1998). "Distributed Multimedia Application and Quality of Service: a Review", Electronic Journal on Networks and Distributed Processing, No. 6, February 1998, pp. 1-50.

[6] K. Nahrstedt and J.M. Smith (1995). "The QoS broker", IEEE Multimedia, spring 1995, pp. 53-67.

[7] 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.

[8] A.H. Salden, R.J. Van Eijk, M.S. Bargh and J. de Heer (2002). "Agent-based brokerage of personalized B2B mobile services SSGRR2002, L'Aquila, Italy, 2002. https://doc.telin.nl/dscgi/ds.py/ViewProps/File-22312.

[9] K. Decker, K. Sycara and M. Williamson (1997). "Middle Agents for the Internet", Proceedings of the 15th International Joint Conference on Artificial Intelligence, Nagoya, Aichi, Japan, August 23-29, 1997.

[10] N.R. Jennings (2001). "An agent-based approach for building complex software systems", Communications of the ACM, April 2001, Volume 44, No. 4, p. 35-41.

[11] FIPA, Foundation for Intelligent Physical Agents, http://www.fipa.org/.

[12] F. Bellifemine, A. Poggi and G. Rimassa. JADE – a FIPA-compliant agent framework, SELT internal technical report. From http://sharon.cselt.it/projects/jade/.

[13] F. Bergenti and A. Poggi (2001). "LEAP: a FIPA Platform for Handheld and Mobile Devices", presented at ATAL 2001, from http://leap.crm-paris.com/.

[14] N. Matos and C. Sierra (1998). "Evolutionary Computing and Negotiation Agents", In AMET98 Workshop on Agent Mediated Electronic Trading, Minneapolis MN, 1998, pp. 91-111.

[15] Page, S.R., T.J. Johnsgard, U. Albert and C.D. Allen (1996). "User customization of a Word processor". In *Proceedings of the conference on Computer Human Interaction,* (pp. 340-346). ACM Press.

[16] Compoze Software Inc, Harmony for MS Exchange, available from, http://www.compoze.com/index.html.

[17] FIPA, Foundation for Intelligent Physical Agents, http://www.fipa.org/.

[18] FIPA ACL, FIPA Agent Communication Language, http://www.fipa.org/repository/aclspecs.html.

[19] FIPA, Interaction Protocol Specifications, http://www.fipa.org/repository/ips.html.

[20] F. Bellifemine, A. Poggi and G. Rimassa. "JADE – A FIPA-compliant agent framework", CSELT internal technical report. From http://sharon.cselt.it/projects/jade/.

[21] N.R. Jennings (2001). "An agent-based approach for building complex software systems", Communications of the ACM, April 2001, Volume 44, No. 4, pp. 35-41.

[22] F. Bergenti and A. Poggi (2001). "LEAP: a FIPA Platform for Handheld and Mobile Devices". Presented at ATAL 2001. From http://leap.crm-paris.com/

[23] N. Matos, and C. Sierra (1998). "Evolutionary Computing and Negotiation Agents", In AMET98 Workshop on Agent Mediated Electronic Trading, Minneapolis MN, 1998, pp. 91-111.

[24] P. Langendoerfer. (2002). "M-Commerce: Why it does not fly (yet?)", SSGRR2002, L'Aquila, Italy, 2002.

[25] GigaMobile project http://www.telin.nl/Middleware/GIGAMOBILE/

[26] B. Ip, "3G Wireless Network Architecture UMTS vs. CDMA2000", ELEN 6951 Wireless and Mobile Networking II, Columbia University.