MOBILE AGENTS IN IMPLEMENTATION OF A VIRTUAL HOME ENVIRONMENT

Kirsi Valtari
M.SC.EE, Leader of Information Networks group in VTT Information Technology, Finland

Key words: Mobile agents, Personal mobility, Service access

Abstract: This paper reports experiences in using mobile agents to implement a virtual home environment. The application environment is an IP-based service network, where the service access is restricted and profiled to those services, which the user has subscribed. The network is envisioned to couple various administrative and technology domains while all services may roam between the domains in order to match the users need to move around and yet connect to his personal communications environment. Agents were applied to provide the user with his personal profile in the visited domain and to negotiate with multiple service providers for the most optimal service offer. Implementation of two agents is studied and the communication trade off is reported. The benefit of using mobile agents within design is discussed.

1. INTRODUCTION

Software agents offer a new programming paradigm, which raises a lot of expectations to tackle problems common in information technology and telecommunications. These challenges include engineering of large distributed systems, optimising communication between computing nodes and supporting mobility of terminals and users.

An agent is a software program, which acts autonomously on behalf of a user (which can be a person or a computer system) or acts to achieve a given goal [10]. The roots of the agent paradigm are in the fields of artificial intelligence and human computer interaction as well as object-oriented

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35581-8_35

T. Yongchareon et al. (eds.), *Intelligence in Networks*

programming [13]. In the field of distributed artificial intelligence, the research on agent collaboration and interaction lead to multi-agent systems and also to the agent communication languages. In the human computer interaction field, the agents were seen as personal assistants of the user, who would learn the preferences and habits of their owner. If an agent has learning capabilities, it is referred to as an intelligent agent.

The concept of a mobile agent combines the characteristics of an agent and code mobility. A mobile agent is an agent that can move in the network performing various tasks [2]. The mobile agents may reduce network load and are claimed to provide asynchronous mode of communication [8]. In addition to this, they might turn out to serve as a more intuitive approach for software engineers currently working with remote procedure call paradigm. The paradigm has been proposed as a successor of client-server in designing distributed systems. The use of mobile agents requires support from the hosting nodes, which has lead to the development of several agent platforms such as Voyager, Aglet and Concordia; a nice overview of mobile agent platforms is presented in [2]. The agent platforms provide typically agent transfer mechanisms, naming and locating services and control of the agent and they could be seen as middleware which enhances Java objects into mobile objects. Voyager [9] is currently the most widely used agent platform and it has also been used in the research work described in this paper.

The agent concept is already being used as a part of the service architecture of next generation networks. In EU/ACTS programme there are roughly ten research projects carrying out prototyping and evaluation of agents within telecommunication applications such as IN, network management and service creation. The projects form a research cluster called CLIMATE [7]. Moreover, the mobile agent concept is also being studied by a number of industry-sponsored projects that investigate the use of mobile agent technology. Those include e.g., NIMA [4] (sponsored by Hitachi and GMD FOKUS) which goal is to design and implement mobile agent based solutions and functions for telecommunications network management, or MAGNA [5] (GMD FOKUS) investigating mobile agent services to be proposed for consideration as part of TINA [6]. However most of the existing projects aim to apply agents in traditional network technologies in contrast to building services framework on top of Internet. In addition concrete experiences and results on performance and applicability of agents are still needed before this new paradigm can be taken into use in operational systems.

As the Internet network keeps spreading from offices into homes and mobile phones, it has become apparent that the IP technology is the platform on top of which future network services will build on. As a next step the

telecommunications infrastructure may also be built on IP-based technology. The low cost and the endless possibilities of open technology will motivate the telecommunications players to solve the problems of security, charging and best-effort packet transfer. As user mobility has become a major requirement for networks, both fixed and mobile networks should support the end user in accessing his services and data. In a situation where all services are built on IP-technology, the user should have a service level roaming possibility over all kinds of networks both in office and in public operator administrative domains regardless of the transfer network technology.

This paper presents research work done in the context of EU/ACTS MONTAGE project [1, 3], that aims at evaluating applicability of mobile intelligent agents in telecommunications applications. The MONTAGE project has used TINA as reference architecture. In this paper the agents are applied to architecture providing service access in Internet. A specific service is envisioned to be restricted to only those who have a subscription for it. The service providers may have agreements, which allow roaming of services over their administrative domains.

2. AN ARCHITECTURE FOR MOBILE SERVICE ACCESS

In order to provide access in foreign service provider domain, a concept of Visited Service Provider (visited SP) is introduced in a similar fashion as in GSM. Mobile agents are used to support the roaming between SP domains. Additionally agents are applied in implementing dynamic service selection among competing offers from different service providers. The potential service providers who could be able to provide an instance of optimal service at a given moment are called Candidate Service Providers. The following roles are given to the service providers in this paper:

Home Service Provider(home SP) is a specific SP, to whom a user has a subscription for one or more services. The Home SP has a user specific database called user profile and a contractual position to bill the user for services used by him. It is assumed that there is a single Home SP per service.

Visited Service Provider is the initial contact point in foreign network environment. It is the service provider offered by the communication network (e.g., ATM, Internet provider or GSM). The Visited SP may also provide the actual service.

Candidate Service Provider is any service provider, who is not the visited SP and with whom the user can access services. This implies that the Candidate SP has a federation or a roaming agreement with the Home SP.

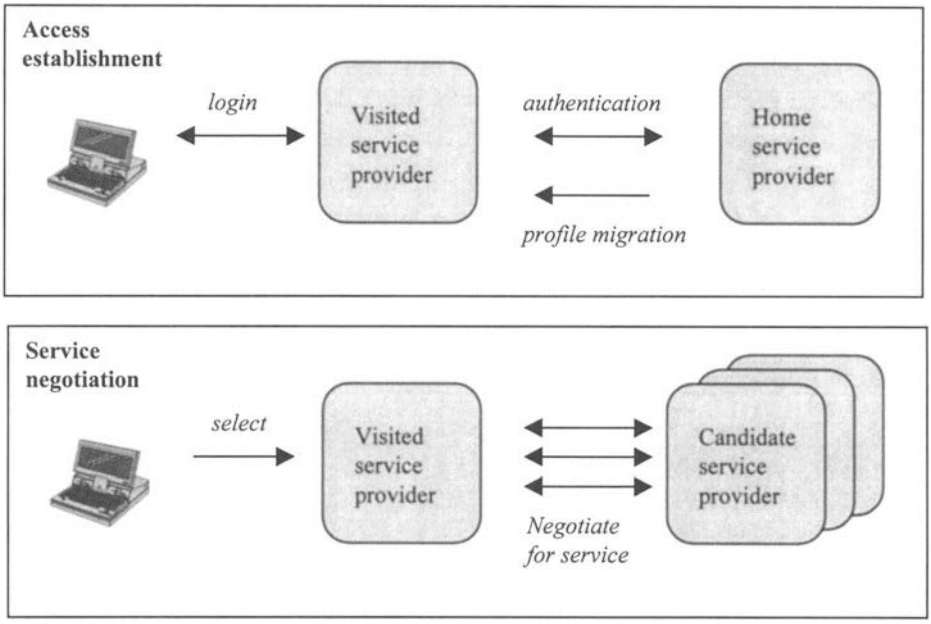

Figure 1. Communication between various participating service providers within access establishment and service selection.

Figure 1 points out the interfaces where communication is active during subsequent phases of service access and selection. The user establishes an access session with a local SP, called visited SP. The user gets authenticated through the Default SP contacting the user's Home SP. In case of positive authentication, a subset of the user's personal profile will be copied to the visited domain. A list of services the user may use in his current location is made available to him. If there are local SP's that have roaming agreements with the user's Home, the user may enjoy personalised service provided locally, while at the same time having the possibility to dynamically select a suitable service offer. Mobile agents are applied in the above scenario to provide a virtual home environment in the visited network and also to implement an efficient negotiation between the SP's to find the most optimal service offer currently available for the user.

A prototype implementation has been made in Java with Voyager [9] as the agent platform. The prototype implements the service access scenario presented in the previous section, including user agent migration and service selection. The current prototype is implemented with the following

technology: SPARC work station, Solaris 2.5.1, Java JDK 1.2, Voyager 3.0. However, in the first version of the prototype Voyager 1.2 was used together with OrbixWeb 3.0.

Figure 2 presents the hardware configuration built for demonstration and testing. It consists of three workstations (representing the SP's and a Content provider), PCs as user terminals and switched Ethernet network. The third workstation hosts a Web server acting as a multimedia content and application provider. A PC implements a user domain with Internet connectivity and a Netscape Web browser as a minimum software requirement. The user terminal does not need to support mobile agents, since they are used only in the communication between service providers. Both SP signalling and service delivery rely on IP for connectivity. The prototype is connected to Mediapoli Pilot network [12] located in Espoo, Finland.

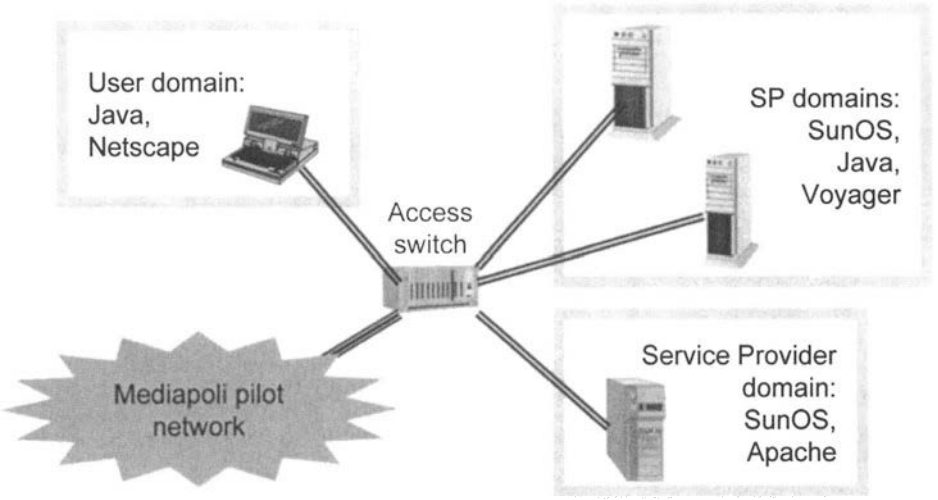

Figure 2. Network configuration used in prototyping

3. AGENT COMMUNICATION

In order to evaluate advantages of mobile agents, a comparison between mobile agent and static object implementation was conducted. A similar functionality was implemented with two different methods of communication: the traditional CORBA communication and agent communication via migration. The mobile agents studied were the User Access Agent (UAA) and the Service User Agent (SUA).

The UAA is specialised in supporting a mobile user by migrating from home domain into the visited domain where it acts on behalf of the user during the access session. The SUA migrates to potential SP's in order to

negotiate an optimal service offer. The traffic caused by the mobile agent implementation was studied and compared with static implementation of comparable functionality. The mobile agent implementation was considered in two cases. In the first case both code and data were transferred with the agent and in the second only the data moved while the code was linked from local library. The second case appears to be typical with Voyager platform, which always checks the availability of the code in target environment prior to transfer. This lead to the following measurement tests:

- Mobile agent UAA migrates to visited domain with code and data attached (mobile agent with code and data).
- Mobile agents UAA and SUA are migrating to default and visited domains carrying data only. A local implementation of the agent is used (mobile agent with data).
- The corresponding functionality is implemented using static objects and CORBA communication (static agent).

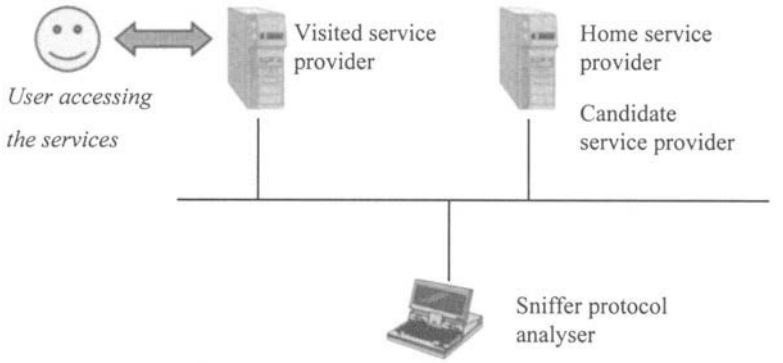

Figure 3. The test configuration used to measure traffic caused by agent migration.

In the tests the SP domains were distributed as shown in figure 3. The SP to which the user has a subscription for services (the Home SP) was located in the same workstation as the candidate to offer services (the Candidate SP). The Default SP that is supporting current access to the user was located in a second workstation. The traffic between the two workstations was recorded with Sniffer protocol analyser. The focus was on data volume of the traffic and also on the traffic pattern (number and size of TCP packets).

In the following sections the communication economics of two mobile agents and corresponding static objects are investigated. Section 3.1 focuses on the user agent UAA, which provides the remote user with his home profile. Section 3.2 gives a comparable study on service user agent SUA,

which negotiates with several SP's in order to find the most optimal service offer.

3.1 User profile migration

In the following text the problem of transferring a user profile into a foreign domain is investigated by comparing three different implementations. The logic of the scenario starts when the user logs into the network in a foreign (visited) domain and asks for an access session to be opened for him. The SP in the visited domain contacts the user's home domain asking for authentication. In case of positive authentication, the user specific profile is sent from home to the visited domain. This action was implemented in three different ways as mentioned in previous section: A) mobile agent carrying code and data, B) mobile agent carrying data, C) static agent.

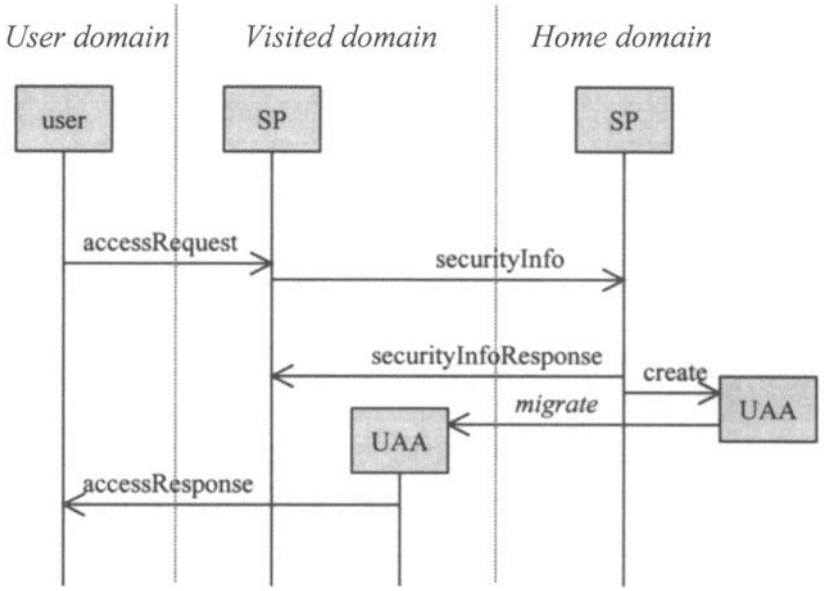

Figure 4. Service access scenario in case when the user is in a foreign (visited) network domain.

Let us first consider the two implementations where mobile agents are involved. The difference between the implementations is that in the first one both code and data migrate with the agent and in the second only data is transferred. Figure 5 presents communication study of implementations made with Voyager 1.1.6 agent platform. Column A) presents measurement

results received in case where the agent carries both data and code. In the beginning the UAA class code does not exist in visited SP domain, which forces the agent to carry code.

The A) scenario starts with securityInfo request with size of 733 bytes (which is equal in all there implementations). The Voyager platforms preparing for agent migration exchange numerous small TCP packets (24 + 19). The wasteful way of using TCP is caused by JDK 1.1 RMI implementation, which is claimed to write header data byte by byte [14]. The migration of the agent generates a TCP payload of 66 365 bytes, divided into 55 packets for transfer. In between each group of packets there is a Voyager level handshake depicted by dotted line. The overall traffic exchanged between agent platforms is 68 484 bytes.

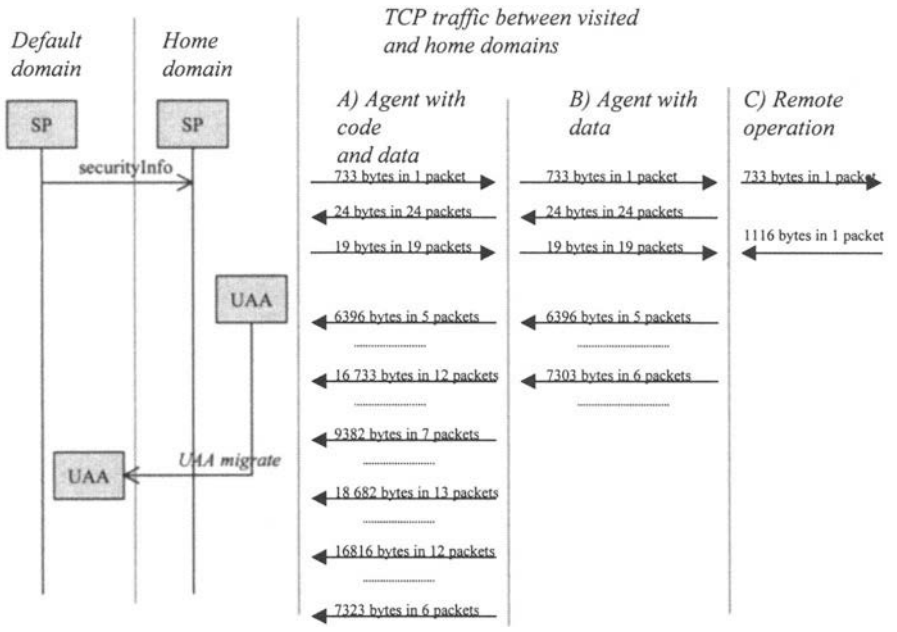

Figure 5. Traffic measurement results with three implementations.

The column B) presents a case when mobile agent UAA migrates from home domain to visited SP domain, but this time the agent carries only data. The scenario starts with securityInfo and Voyager platform specific communication as in the previous case. The migration of the agent generates a TCP payload of 13 719 bytes, splitted into 11 packets for transfer. In between each group of packets there is again a Voyager level handshake. The overall traffic exchanged between agent platforms is 14347 bytes.

When comparing implementation A and B, it is obvious that the transfer of code seems to be inefficient. The traffic volume of the agent carrying code is roughly five times bigger than the agent carrying only data.

Column C in figure 5 reflects a situation when the UAA migration has been replaced by remote operation in the implementation. However exactly the same data, the user profile, is passed as parameter. The UAA agent in visited SP is replaced by a UAA_static object. The scenario starts with securityInfo request with size of 733 bytes as in the former case. This is followed by a securityInfoResponse of size 1116 bytes, which contains the user profile data. The static implementation using remote operation appears to be far more efficient than the agent based implementation. The size of the agent carrying profile data is roughly eleven times bigger than the comparable securityInfoResponse containing user profile.

A second version of the UAA agent was implemented on Voyager 3 platform. An analysis similar to case B (agent with data) showed that the Voyager communication has substantially improved. The total communication between platforms was 8779 bytes (using two TCP connections) and the TCP round-trip number was reduced to eigth.

2.1 Service negotiation

The second agent under study was the negotiating agent, SUA. The task of the agent is to find the best service offer provided by competitive service providers. In the following text the trade off of the negotiation agent is investigated as in the former section, by comparing two different implementations. The logic of the scenario starts when the user has selected a service type with a specific wish to have it delivered by the most optimal service provider. The user agent located at the visited domain will create a number of dedicated negotiating agents (SUA) and send them to potential service provider nodes to negotiate about an optimal service offer.

Figure 6 presents the message sequence in the SUA negotiation when mobile agents are used. In case of static implementation the negotiation is performed using remote CORBA calls. The sequence is designed to use agent replication which allows parallel negotiation with a vast number of candidate SP's although in the figure only two candidate SP's (and SUA agents) are visible. In the following text we will focus on the interface between visited domain and candidate SP domain in order to calculate the traffic volume exchanged during the agent based negotiation.

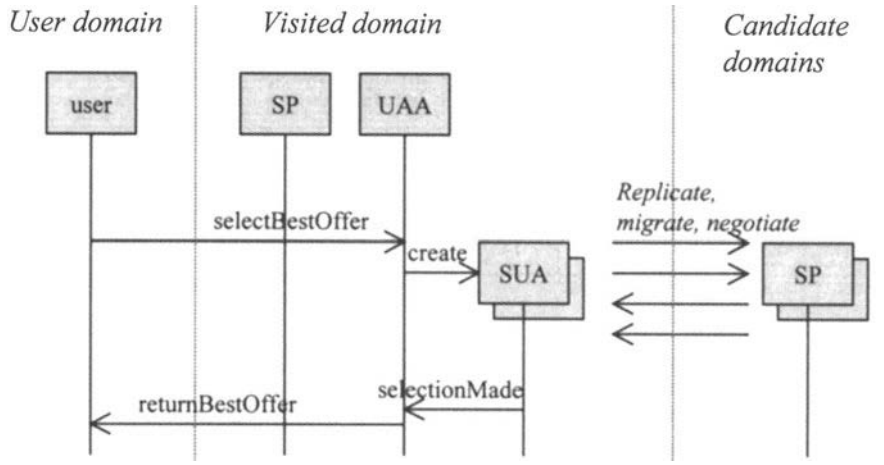

Figure 6. Negotiation for service selection.

In figure 7 column A) presents traffic analysis results when mobile agent SUA migrates from visited domain to candidate SP domain and then locally goes through a negotiation algorithm with an agent representing the retailer. The traffic relating to only one candidate SP negotiation is shown, since the negotiation with each candidate SP results in the same traffic pattern. The SUA agent in this implementation carries only data and no code. This is because the test configuration had only two workstations and home and candidate SP's were run in same computer. In such situation Voyager platform was able to detect the existence of SUA code implementation in the target domain.

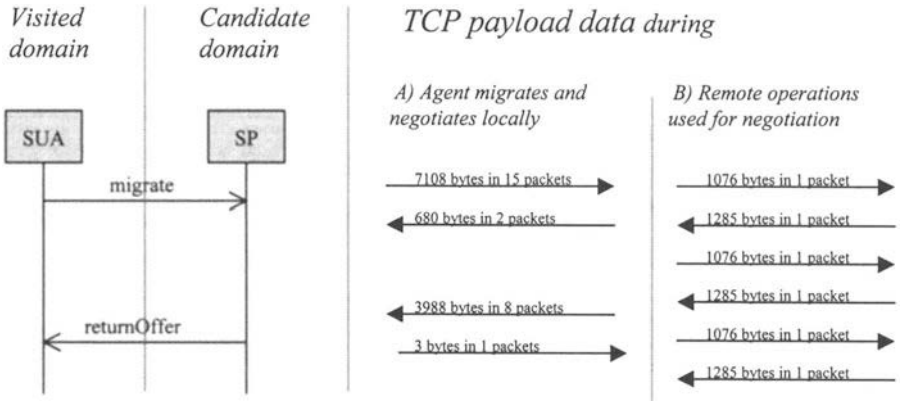

Figure 7. Traffic measurement results of SUA negotiation with one candidate SP.

The migration of the SUA agent causes a transfer of 7108 bytes from visited SP domain to the candidate SP domain. The receiving side agent platform apparently acknowledges the migration. The negotiation reply is passed as an argument in a method call and it causes transfer of 3988 bytes. The large size of the reply is caused by the user profile data, which is transferred in case the negotiation would produce a slightly differently profiled service offer. The overall traffic generated when the SUA agent negotiates with one candidate SP is 11 779 bytes. Two TCP connections are opened one for the agent migration and one for passing back the negotiation result. The overall number of TCP round-trips is 11, not including the opening or the connection. The number of round-trips could be optimised to two, if maximum packet size was used.

In figure 7, column B) depicts the static implementation corresponding to the service selection scenario in case of one SUA agent. The SUA agent is replaced by a SUA_static object, which makes a CORBA method call from visited SP domain to RA in candidate SP domain. The negotiation algorithm requires exchange of three method calls between each SUA and RA pair. In agent implementation of column A) this is implicit and not visible since it is local communication. In static implementation the negotiation leads to three remote calls in order to receive the selection result. Each of the three method calls generate traffic volume of 2361 bytes to be exchanged. It maybe easily calculated that already in case where the negotiation algorithm requires more five method calls, the total traffic volume of static implementation exceeds the volume in case of mobile agent implementation.

2.2 Discussion on measurement results

In previous sections the communication cost of two mobile agents was analysed by comparing the agent implementation with a static implementation of a similar functionality. The following figures show the total communication between the nodes during the user profile migration and service negotiation scenarios.

UAA, user profile migration (implemented with Voyager 1.1.6)

a) mobile agent with code and data, 68 484 bytes
b) mobile agent with data, 14 347 bytes
c) static object implementation, 1 849 bytes

SUA, service negotiation (implemented with Voyager 3)

d) mobile agent with data, 11 779 bytes
e) static object implementation, 7 083 bytes

The task of the UAA agent is to roam the user's home environment into a visited domain and act on his behalf in the network side during the access. In

essence, it provides a virtual home environment for the user. A simple version of the home environment is reached when only the user profile is transferred into the distant domain. The implementation UAA c) corresponds to this case. The limitations are that the SP's must have exactly the same model of the user profile (a standardised interface) and the user must satisfy with the functionality offered by the specific SP in the current location.

A more enhanced version of home environment offers not only the profile data, but also the code and algorithms, which the user prefers. The cost of the mobile agent based implementation would probably situate between the first and second column in the table 1. This stems from the belief that part of the agent code would usually be found in the target location. Only such pieces of code, which require update or contain personalised features, would be normally migrated with the agent. This kind of behaviour is also supported by the agent platforms. Allthough the difference in communication overhead between mobile agent and static implementation seems big, it may become insignificant when compared with the volumes of the application data (eg. multimedia communication). Also, the overhead is not a big trade off when the availability of a preferred functionality needs to be assured. In the access scenario example, the SUA negotiating agent is not necessarily available in foreign domain unless it is brought there within the UAA agent.

The SUA agent demonstrates the benefits of mobile code in a nice way. In our test setup the SUA migrates to a candidate SP node where it negotiates locally by exchanging three request/reply pairs. This is compared with a static SUA issuing three method calls to a remotely located object. Already if the negotiation algorithm would involve five request/reply pairs, the use of mobile agents would lead to a more efficient implementation than the use of static objects from the communication economics point of view. The performance of the SUA mobile agent negotiation is optimal from the user's point of view, since all SUA replicas process simultaneously in different SP nodes and only the negotiation result is passed back to the visited SP node.

The scalability of the agent implementation studied is estimated to be reasonably good. The UAA migration consumed communication resource, but since it is not a frequent event in the network it is seen as acceptable overhead. The SUA migration scales well since by nature it converts part of the remote communication into local communication in the target node. Both agents of course require memory space and processor resources in the SP domains, which support the user mobility. A strong benefit of mobile agents is the new way to model a distributed system and especially asynchronous communication.

3. CONCLUSIONS AND FUTURE WORK

This paper has presented an implementation of service access architecture which exploits mobile agents. The agents were used to provide a virtual home environment for a nomadic user and to negotiate about an optimal service offer. The service profile of the user was carried to the visited network domain by a migrating user agent.

The communication cost of a mobile agent based implementation has been analysed and compared with a static implementation. The agent migration was reported less economical than remote operation call in all studied examples, but it was found that the migration of the negotiating agent pays off allready when the agent makes five or more local method calls in the visited node. The transfer of code was found very consuming, however it is seen as a worthwhile price to pay whenever the target environment is lacking the services customised to the user. The agent platforms are still heavily evolving and this is reflected also in the implementation experiences with two platform versions.

The MONTAGE project continues to enhance the agent based service software. A user trial will be carried out in Medipoli pilot network later this year in order to evaluate the benefits of mobile agents. The security aspect of the agent based design remains for future study.

REFERENCES

[1] Montage www page, http://montage.ccrle.nec.de/

[2] Vu AnhPham, Ahmed Karmouch, Mobile Software Agents: An Overview, IEEE Communications Magazine, July 1998.

[3] Henryka Jormakka, Kirsi Valtari, Andreas Kind, Didoe Prevedourou, Kimmo Raatikainen, Agent-based TINA Access Session Supporting Retailer Selection in Personal Mobility Context, TINA-99 Conference, April 1999.

[4] http://www.fokus.gmd.de/research/cc/ima/projects/

[5] http://www.fokus.gmd.de/ice/projects/magna.html

[6] TINA-C, Homepage, http://www.tinac.com 1998

[7] http://www.fokus.gmd.de/cc/ima/climate/

[8] Andreas Kind, Mitiades E. Anagnostou, Malamati D. Louta, Jan Nicklisch, Juan Tous, "Towards the seamless integration of mobile agents into service creation practice", IS&N Conference proceedings, April 1999

[9] http://www.objectspace.com/products/index.html

[10] S. Albayrak et al., "Intelligent agents for telecommunications applications", Proceedings of IATA 98' Workshop, IOS Press, June 1998

[11] Michael S. Greenberg and Jennifer C. Byington, Thephany Holding, David G. Harper, "Mobile Agents and Security", IEEE Communications Magazine, July 1998

[12] http://www.mediapoli.fi

[13] Nicholas R. Jennings, Katia Sycara, Michael Wooldridge, "A Roadmap of Agent Research and Developement", Autonolous Agents and Multi-Agent Systems, 1, 7-38 1998

[14] Stefano Campadello, Heikki Helin, Oskari Koskimies, Kimmo Raatikainen, "Optimizing Java 2 RMI for Slow Wireless Links", submitted for publication in Mobile Computing and Communications Review, Volume1, Number 2, 1999