Agent technology for future mobile networks

Jens Hartmann, Wei Song

Ericsson Eurolab Deutschland GmbH, Ericsson Allee 1, D-52134 Herzogenrath, Germany
E-mail: {eedjhn, eedwso}@eed.ericsson.se

Abstract

This paper describes an implementation of an agent-based homebanking service for a mobile network. Agents offer the prospect for revolutionising the provision of services in telecommunications networks by allowing dynamic distribution of service related data and processing across networks and terminals. At Ericsson Eurolab, Germany, an agent-based solution has been implemented for "services on demand".[1]

Keywords: *Agent Technology, CAMELEON, Mobile Agents, Voyager, Service on Demand, Homebanking, HBCI.*

1. Introduction

From the scientific disciplines Artificial Intelligence (AI), Distributed Systems and Object Orientation (OO) a new scientific working field called Agent Technology has been established. Currently agents are seen as a key to the issue of service provisioning within future telecommunication environments.

The fixed telecommunication networks are now providing not only voice services but also more and more data based services, whereas mobile networks are still intending to develop into this direction. Compared with the traditional voice based networks; future networks need more flexibility to fill the new purposes. Agents, especially Mobile Agents (MA) seem to be best suited because of their characteristics of autonomy, intelligence, mobility, co-ordination and co-operation.

Considering provisioning of new, sophisticated services by a network operator or service provider, the services should be provided in a more direct and flexible way. A user should have the opportunity to access these services anytime and anywhere, independent of the terminal and network technology.

Using agents, services can be easily subscribed, downloaded or migrated to the user's end device. The user needs only to subscribe a certain type or version of agent, who realises the user's preferred "look and feel". Thus, this new service-provisioning paradigm can be identified not only as 'service on demands', but also as 'look and feel on demands'.

Using an agent-enabled system, agents can represent almost any party of the system. Some old application processing scenarios can be executed in a more effective and flexible way. In particular, this is valid for new applications in the well-promising area of electronic commerce, because in the past it was only tried to realise these applications

(1) This work was partly supported by the Commission of the European Community (CEC) as part of the ACTS CAMELEON project (AC341)

in a transaction based manner. With the help of agents, it is possible to build these applications easily in an asynchronous way.

The work described in this paper has been undertaken in the framework of the *CAMELEON* project [CAM98], which develop and trial service roaming applying Agent Technology in a mobile network environment using the Virtual Home Environment (VHE) as a test case.

2. Mobile Telecommunication Networks

The success of the 2nd generation mobile communication standard GSM [Mou92] relies, among other things, on the possibility to roam between networks – and thus between countries – by using a single subscription. This means that the subscriber is reachable using a single number and receives a single bill from his/her home service provider. The three most important topics for future mobile users will therefore be the same as they are today for GSM users:

- Easy handling of the desired telecommunication services, including the opportunity to customise the 'look and feel' of services and get 'services on-demand'
- Global availability and consistent performance of telecommunication services
- Understandable billing with a single point of contact

However, the future telecommunications world will not be homogeneous, and therefore these goals cannot be achieved easily. This has been identified by the telecommunications standardization bodies ITU and ETSI, and measures have been taken to allowing 'service roaming', sometimes referred as service portability. This concept shall be realised by the VHE, which shall enable a visited network to obtain information about the user's Service Provider (SP) during the registration procedure and other information such as the user's personalised service profile and the identification of service capabilities needed for the execution of SP specific services. Although the physical realisation of a service may differ from one network to another, the VHE concept enables the user to access and to use the service in the same way on any network. The VHE is currently being standardised in ETSI SMG 1 and ITU SG 2 for the use in 3rd generation mobile networks such as the Universal Mobile Telecommunication Network (UMTS) and International Mobile Telecommunications 2000 (IMT-2000) [ITU98].

For these third generation networks frequency spectrum has already been allocated in the 2 GHz frequency band. The UMTS air interface will utilise W-CDMA for the wide area environment with a proposed TD/CDMA structure for the unpaired banks. In implementing this solution the ETSI representatives follow the specification of UMTS with the objective to provide:

- low-cost terminals
- harmonisation with GSM
- FDD/TDD dual-mode operation terminals

However, the global standardisation work being conducted by public authorities and the industry is ongoing.

3. Agent Technology

Although Agent Technology is currently ubiquitous in the telecommunication research domain, there is no commonly agreed definition of what an agent is. We propose to use the following definition, which has been developed within the *CAMELEON* project [CAM98]:

*An agent is a **piece of software**, which is able to perform a specific predefined task **autonomously** (on behalf of a user or an application). An agent is either stationary providing the necessary **intelligence**, or **mobile** so that it can move between **distributed** (possibly incompatible) systems to access remote resources or even meet other agents (or activate them). All agents have capabilities to **co-ordinate**, **communicate** and **co-operate** with the system or with other agents.*

From this definition the most important attributes of an agent can be derived:

- **Autonomy**; similar to an agent in the human world, a software agent tries to fulfil the required tasks autonomously according to his rights. The agent knows his mission, and how to accomplish it. There is no need for a permanent connection with the principal;
- **Intelligence** is needed, if the agent should achieve his goals sensible. Moreover, the agent could be able to plan (re-active), and learn (adaptive) from the things he is doing;
- **Mobility** means the capability of an agent to move through a network. Before the agent start to migrate, the program execution should be stopped, and the program code and data should be packed up. After migration, the agent should be able to continue the program execution from the last breakpoint.
- **Distribution** should a priori cause no problem to agent. Agent should know how to interact with physical and logical distributed systems. Furthermore, an agent itself could be called a special kind of a distributed object.
- **Co-ordination**; an agent system should be build up in a way that mechanisms to avoid conflicts are available. Moreover, in the case of a conflict the agent should know how to handle it. Tasks such as synchronisation of jobs, consistency handling, redundancy avoidance and the addition of controlled redundancy belong also the group of co-ordination activities, which should be considered when implementing an agent, respectively an agent system.
- **Communication** plays of course an important role in the agent domain. Methods are needed to describe how to communicate, protocols are necessary to define who is allowed to communicate, and an agreement on the content is essential, if the partners should understand each other.
- **Co-operation** mechanisms appear in the case of problem division, task sharing, resource problems, result sharing, and result synthesis.

From this long list of requirements it can be concluded that there is a need for a classification scheme of agents. From very a high level perspective two major types can be identified [Mag96]:

Mobile Agents, in which the mobility of code, data and state is the most fundamental attribute. This allows software entities to roam autonomously through a network and to perform dedicated tasks at specific network nodes, thereby taking advantage of locality;

Intelligent Agents, who are software entities that are able to perform delegated tasks based on internal knowledge and reasoning, where aspects such as inter-agent communication and negotiation are fundamental. Usually mobility is not considered as an issue.

The focus in this work is on Mobile Agents (MA), which have a shorter history, and are more oriented towards network and communication applications. The combination of MAs and mobile telecommunication system sounds promising, because it seems that MAs could overcome the typical restrictions coming from mobile networks, such as:

- limited bandwidth
- high bit error rate on the air channel
- bounded coverage
- low processing power of the end-systems
- simple user interface

Moreover, the application of MA for service provisioning brings the following advantages:

- asynchronous communication is possible
- agents could work without a permanent network connection
- reduction of network traffic
- very processing power consuming activities could be performed locally
- the reality could be better modelled with agents

4. Mobile Agent Systems

Mobile Agent Systems (MAS) for service provisioning are continuously evolving. Besides providing a system development infrastructure the existing platforms support security, inter-agent communication, agent transport protocols, remote messaging, etc. Several platforms are available but most of these products are in their Beta stages [CAM98]. Some of the JAVA-based mobile frameworks available are: (1) *Voyager* from Object Space, (2) *Aglets Workbench* from IBM, (3) *Concordia* from Mitsubishi Electric, (4) *Agent TCL* from Dartmouth College, (5) *Odyssey* from General Magic, (5) *CyberAgent* from FTP Software, (6) *Grasshopper* from IKV, and (7) *JIAC* from the Technical University of Berlin.

We have decided to build our application on top of the Voyager platform, because several tests have shown that Voyager has the best system performance concerning agent transmission. Voyager is the ObjectSpace product designed to help developers produce high-impact distributed systems quickly [Obj97]. Voyager is now in a phase of Version 2.0. Voyager is implemented in programming language Java [Jav98] and is designed to use the Java language object model. Voyager allows its user to use regular message syntax to construct remote objects, send them messages, and move them between programs and objects.

The root of the Voyager product line is the ObjectSpace Voyager Core Technology. It contains the core features and architecture of the platform, including a full-featured, intuitive object request broker (ORB) with support for mobile objects and autonomous agents. Also in the core package are services for persistence, scalable group communication and basic directory services included.

The key aspects of this framework are as follows:

- Voyager is 100% Java. Voyager applications can be written once and run on a platform supporting Java 1.1 or later. Voyager can remotely construct and communication with Java classes, even third-party libraries without accessing the source code.
- Objects can be created remotely using the regular Java construction syntax, and meanwhile a reference of this object will be keep locally. Static methods can be executed remotely, and remote exceptions are automatically forwarded to caller. A remote-enabled serializable object can be move through the network, even if it is receiving messages.
- Objects can exchange messages locally and remotely using regular Java message syntax. If an object is transported, the messages sent to it will be automatically forwarded to its new location. Voyager supports synchronous messages, one-way messages, future messages, one-way multicast messages and selective multicast messages.
- Using Voyager it is relatively easy to prepare a class for remote programming. With a single Voyager command an object can be enabled remotely.
- A persistent object has a backup copy in a database, so that it can be automatically recovered, if its program is unexpectedly terminated or if it is flushed from the memory to save working space.
- Voyager supports JavaBean.

5. Agent-based Homebanking

Today, most banking service users have to run special software from their bank to access the services offered by the bank. This leads to the fact that the user has various software packets installed on different computers, which he also has to maintain. Moreover, the user does not have an identical ‘look and feel’ of the service, because the user profile and configurations are stored separately on each machine. To integrate the homebanking service directly in a telecommunication network could change this situation, because than the homebanking service provider acts as a universal bridge interface between the user and his banks, see Figure 1. Every subscriber will get his unique user profile and an identical ‘look and feel’ is also supported.

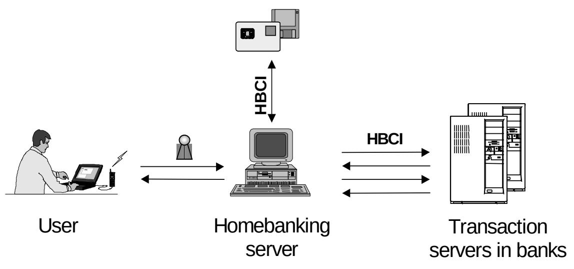

Figure 1: Agent-based homebanking scenario

The transaction servers, who could process the transaction requests from their clients, are running in the domain of a bank. A homebanking server is a homebanking client from the view of a bank and a homebanking service provider from the view of a user. The homebanking server will register itself to a naming service at beginning, so that every *Terminal Agent* could find it. A user could then subscribe the homebanking service by downloading and starting the homebanking client, i.e. the *Homebanking Terminal Agent*, using his *Terminal Agent*. After starting the *Homebanking Terminal Agent*, the user has the possibility to create *Homebanking Agents*, give them specific tasks to process and launch them. After he has been started the *Homebanking Agent,* the agent will move its tasks to server, process them locally and bring the results back when the user is online. As a GSM connection is limited by its narrow bandwidth, a main purpose of this design is to optimise the network usage maximally. Using a *Homebanking Agent*, the communication between the *Terminal Agent* and the server is not transaction-based but asynchronous. In every host, the *Terminal Agent* has to be downloaded only once and can than be reused at the next time. The network connection is necessary mostly to transport *Homebanking Agents*.

The homebanking server communicates with transaction servers in banks over the Homebanking Computer Interface (HBCI) protocol [HBC97, Kel98], that has been recently standardised by German Central Finance Committee (ZKA). HBCI defines the communication between intelligent customer systems and the corresponding computing centres for the exchange of homebanking transactions. As this protocol has been submitted to the European Committee for Banking Standards (ECBS) it looks feasible, that most banks in Europe will support this protocol in the near future.

Generally, an HBCI message consists of message header and trailer, signature header and trailer, and a number of message segments indicating the banking part of several business transactions. Optionally there exist a ciphering header for the ciphering of data and eventually additional signature headers and trailers for multiple signatures. In the first versions, many of the classic business transactions have been defined in HBCI, e.g. balance inquiry, sales statistics, domestic and foreign transfers, single and collective credit/debit note, and exchange rates. More transactions will be added in the next versions, e.g. loading of the German cash card *GeldKarte* [Kel98].

However, the main purpose of the overall service is to benefits from the various advantages of agent characteristics, see Chapter 3. Therefore, the following classes of agent types have been identified:

Terminal Agents

Terminal Agents represent the interface between a user and the system. The user can contact a general *Terminal Agent* in order to enter or leave the system or to start and shutdown a service. A service specific terminal agent provides service specific interactions. Depending on different types of terminal devices, a *Terminal Agent* can have a graphic or text-based user interface.

User Agents

These agents content the full user profile of a user, and represent the preferences and activities of the user. It is also responsible for primitive and advanced migration and consistency of user preference data.

Management Agents

This type of agents manages the system resources. They can be realized as a set of agents, in which every agent or agent group is responsible for one or more subtasks. For example, user access controlling agents are responsible for access controlling of some type or the whole resources.

Communication Agents

These agents are responsible for the management of communication with external components.

Service Agents

A *Service Agent* represents a service to be provided, which can be used by a user or a component. Because the realization of services differs from one to another, *Service Agents* can be a whole service or only a part. The *Homebanking Agent* is an example of a *Service Agent.*

Tool Agents

This category of agents realizes a set of tools for common purposes. For example, an agent wants to inform his owner some messages, but it does not know if his owner is now online. It just starts a Report Agent, who will try to find the user and bring him these messages. Applications might be strongly simplified through the usage of such agents.

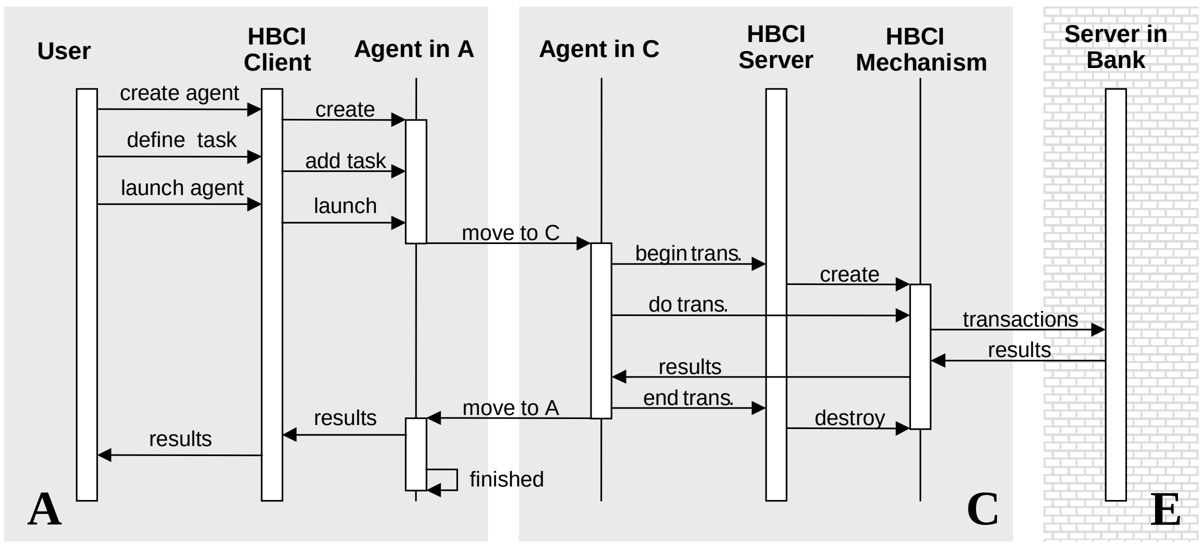

Figure 2: Message Sequence Chart of an FFS transaction

The general sequence of an agent-based HBCI transaction is shown in

Figure **2**. In general, if the *Homebanking Terminal Agent* is already available on the end-system the user is currently using, first the user has to create a least one *Homebanking Agent*. Then the user has to define one or more tasks and delegate them to one specific *Homebanking Agent*. For the different kinds of tasks, which should be supported by the *Homebanking Agent* altered solutions for security handling and data storing were considered.

When talking about security handling, you should have in mind that HBCI supports different kind of security schemes [HBC97]. For example, there are two ways to store the different asymmetric key pairs needed for the HBCI-based communication between the Homebanking service in the domain C and the bank server in domain E, see also

Figure **2**. The first solution is based on the Data Encryption Standard (DES). Here, the keys are stored on the SmartCard, which also performs the calculation of the cryptographic functions. For the second solution uses the asymmetric algorithm of Rivest-Shamir-Adleman (RSA). In this case the keys are normally stored on an ordinary storage media, such as a diskette or a harddisk.

6. Conclusion

Within the framework of the research project *CAMELEON*, which is founded by the European Union, we have implemented an agent-based homebanking service in the commercial mobile agent platform Voyager from ObjectSpace. This service, which has been introduced briefly, allows triggering a huge set of banking transaction over the air-link with the help of Mobile Agents. The main processing of the service is undertaken locally at a bridging homebanking server, which performs the protocol specific communication with bank server based on a new German banking standard called HBCI. A mobile user can access the homebanking service by simply downloading and starting a Homebanking Terminal Agent using his Terminal Agent, sitting on top of the Java-based Voyager platform on any system with a JAVA Virtual Machine.

This homebanking service is a good example how to use agent for enhanced service provisioning. It proves that the application of Agent Technology in future telecommunication networks for end-user services is reasonable. In particular the service shows, that two important challenges for future mobile networks, i.e. the customisation of the 'look and feel' of services and the download of 'services on-demand', could be realised easily with the help of Agent Technology. In the near future it should be also investigated, if agents also bring some benefits for to telecommunication network and management services.

7. References

[CAM98] *CAMELEON* Consortium. *An Open Communication Environment Using Agent Technologies*, ACTS 341 – Technical Annex Part B, March 1998.

[HBC97] Bundesverband deutscher Banken. *HBCI – Schnittstellen-spezifikation*. Version 2.0 (in German), July 1997.

[ITU98] International Telecommunication Union. *IMT-2000*. http://www.itu.int/imt/

[Jav98] Sun Microsystems. *JAVA*. http://java.sun.com/

[Kel98] Keller R., Zavagli G., Hartmann J., Williams F. *Mobile Electronic Commerce: GeldKarte Loading Functionality in Wireless Wallets*. International IFIP/GI Working Conference: Trends in Electronic Commerce, Hamburg, June 1998.

[Mag96] Magedanz T., Popescu-Zeletin R. *Towards "Intelligence on Demand" - On the Impacts of Intelligent Agents on IN*. 4th International Conference on Intelligence in Networks, Bordeaux, France, November 1996.

[Mou92] Mouly M., Pautet M.-B. *The GSM System for Mobile Communications*. Published by the authors, 1992.

[Obj97] ObjectSpace. *Voyager Core Technology User Guide*. Version 2.0 Beta 1, 1997.