INNOVATIVE TECHNOLOGY SERIES

INFORMATION SYSTEMS AND NETWORKS

Intelligent Agents for Telecommunication Environments

edited by
Dominique Gaïti & Olli Martikainen

HPS
HERMES PENTON SCIENCE

Intelligent Agents for Telecommunication Environments

This page intentionally left blank

INNOVATIVE TECHNOLOGY SERIES
INFORMATION SYSTEMS AND NETWORKS

Intelligent Agents for Telecommunication Environments

edited by
Dominique Gaïti
Olli Martikainen

HPS
HERMES PENTON SCIENCE

First published in 2000 by Hermes Science Publications, Paris
First published in 2002 by Hermes Penton Ltd
Derived from *Networking and Information Systems Journal, Intelligent Agents for Telecommunication Environments*, Vol. 3, No. 3–4.

Hermes Penton Science
120 Pentonville Road
London N1 9JN

British Library Cataloguing in Publication Data

A CIP record for this book is available from the British Library.

ISBN 1 9039 9629 5

Printed and bound in Great Britain by Biddles Ltd, Guildford and King's Lynn
www.biddles.co.uk

Contents

This page intentionally left blank

Foreword

This publication deals with agents and services in a telecommunications environment and the contributions included are based on those which were submitted to the Networking 2000 conference and especially to the IATE (Intelligent Agents in Telecommunication Environments) mini-conference.

Service and network providers must be able to satisfy the demands for new services, improve the quality of service, reduce the cost of network service operations and maintenance, control performance and adapt to user demands. In other words, it seems essential to investigate new approaches under consideration for performing such tasks.

Telecommunication systems become more and more complex and dynamic with the introduction, for example, of new services, mobility and active networks. The use of artificial intelligence and intelligent agents, integrating reasoning, learning, cooperating and mobility capabilities to provide a predictive control seems promising in such environment. Different areas around telecommunication systems have to be investigated: performance, flow and congestion control, intelligent control environment, security, service creation and deployment, mobility of users, terminals and services, to mention a few.

New approaches under consideration are the following:

– Introducing intelligence in nodes and terminal equipment in order to manage and control the protocols. One way to make the network "intelligent" is to use a distributed multi-agent system capable of managing and controlling network resources. Using intelligent agents and distributed artificial intelligence concepts are interesting ways of supporting shared intelligence.

– Introducing intelligence mobility in the global network. Very soon, users will be able to define their own services based on their own needs and ask for the associated quality of service. The aim of such tools is to provide the quality of service and adapt the existing infrastructure to be able to handle the new services' functions and to achieve a necessary cooperation between nodes.

This publication addresses some of the problems presented above, including control, services, distributed intelligence, mobile and intelligent agents and security, and investigates new trends.

The area of intelligence in networks will provide stimulating challenges for years to come. In this publication the authors share their answers to these questions and provide some direction about new available approaches.

Dominique Gaïti
Olli Martikainen

Chapter 1

Mobile agents and security

Fritz Hohl
Sony International (Europe) GmbH, Stuttgart, Germany

1. Introduction

Mobile agents are program instances (or processes) capable of moving within the network under their own control. They consist of three parts: code, data state, and execution state. Migration of mobile agents takes place between hosts. These hosts execute the agents and provide functionality to them. This functionality includes communication with other agents and users, migration to other hosts, and other services such as persistency (i.e. storing data or agent snapshots), services lookup (i.e. finding agents that offer a service), location of agents (either locally or on a larger scale), integration of legacy systems such as databases, and the like.

Being the technical basis for applications, mobile agents offer a number of advantages. These include the saving of network bandwidth and increase in the overall performance by allowing the application to process data on or near the source of data (e.g. a database), asynchronous processing, i.e. the possibility to fulfil a task without the need to have a permanent connection from the client to a host, achieving true parallel computation by employing a number of agents working on different nodes, the replacement of a fault model where network failures can interrupt every phase of the computation by one where network failures can influence only the migration of an agent (as the rest is then done locally on the same node), and so on. Additionally, mobile agents "inherit" the advantages of mobile code systems, especially the possibility to transport functionality automatically to nodes where it has not been installed before.

For mobile agents, security is a very important aspect since neither the provider of a host or an agent-based service, nor the owner of an agent wants to be harmed by employing this technology. This is a non-trivial requirement in mobile agent systems, because firstly, the executing party has no vital interest in executing a program correctly, and secondly, the employer of a program has to give away the control over its execution.

This paper presents some of the problems and approaches in the area of security for mobile agents, especially those focusing on the attacks between mobile agents and hosts. Therefore, in Section 2, a categorisation of the area of security in the context of mobile agents is presented. Section 3 concentrates on approaches to protect hosts from attacks by malicious agents, whereas Section 4 focuses on approaches to protect mobile agents from attacks by malicious hosts.

2. Categorisation

Security in the area of mobile agents can be categorized according to the possible conflicts of interests between the parties involved, and their corresponding components, respectively.

These components and the associated parties are:

– agents (or their owners, resp);

– hosts (or their operators, resp.).

Having these two parties, three different conflicts of interests can be distinguished.

2.1. *Agent–agent*

This category includes attacks between two agents. These attacks can be divided into organizational and technical attacks. Organizational attacks, which include masking of hosts, and denial-of-service attacks on the level of agents, are a general problem of interacting partners; they are not problems specific to mobile agents. Therefore, the same solutions to prevent such attacks can be used as in applications that do not use agents. A good example is protocols that allow electronic. Here, different versions of different security levels exist according to the anonymity needed and the protection level. Technical attacks use security holes in the implementation of hosts. They are a problem if two agents are executed on the same host. Depending on these security holes, an attacking agent may e.g. access the attacked agent by using unprotected memory areas, thus reading or manipulating private agent data. To solve this problem, programming languages are often used for hosts that (at least conceptually) allow isolation of agents from each other, e.g. Java.

2.2. *Host–host*

This category includes attacks between different hosts. The most important, non application-specific attack here is masking, i.e. pretending to be another host. This attack can be prevented easily using existing authentication mechanisms, as hosts match the roles of autonomous, independent parties that are used in traditional security schemes.

2.3. *Agent–host*

This category is specific for mobile agents, as the main problem, the separation of the executing party and the one that owns the executed agent, results from the ability of an agent to migrate. The category can be divided into two subcategories. The first, the area of protection of hosts from attacks by malicious agents, will be examined in the next section. The second subcategory, the area of protection of mobile agents from attacks by malicious hosts, will be presented in Section 4.

3. Malicious agents

The problem of malicious agents occurs when mobile agents attack the host that executes the agent and/or the computer system on which the host operates. A very similar problem occurs also at other mobile code systems, e.g. in the case of Java applets executed in WWW browsers. The difference from applets is that such entities are executed on behalf of the operator of the executing environment. The conflict of interest therefore exists between the applet executor and the programmer of the applet. Thus, attacks by applets are normally directed against the computer system of the executor, whereas attacks by malicious mobile agents may also be directed against the host as a part of the agent system. So the problem of the secure execution of applets constitutes a part of the problem of the secure execution of mobile agents. Therefore, some of the approaches for executing applets can also be used in the area of mobile agents.

3.1. *Attack classes*

Possible attacks of a malicious agent against the host can be divided in two areas; attacks against the host as a part of the agent system, and attacks against the computer system a host is located on.

3.1.1. *Attacks against the host*

Attacks against the host as a part of an agent system cause problems only inside, but not outside the host. Such attacks include:

3.1.1.1. Unauthorized usage of resources

Hosts offer resources like computation, memory, and communications means to agents. Mobile agents consume these resources to different degrees. The question of which consumption is acceptable for which agent, and which consumption is not, depends on criteria defined by the host. These criteria may be defined in an explicit manner, or they may exist only implicitly.

3.1.1.2. Denial-of-service attacks against the host

The aim of this attack is to prevent the execution of existing or arriving agents by the host, e.g. because an attacker wants to exclude concurrent usage of its own host. This can be done e.g. by one or more agents that try to consume all resources of the attacked host.

3.1.1.3. Denial-of-service attacks against other agents

Not only hosts, but also other agents may be the target of denial-of-service attacks. Such attacks typically try to overload attacked agents or try to crash them exploiting known implementation bugs.

3.1.1.4. Read attacks against other agents

As agents may carry valuable goods like electronic money, attacking agents might try to read these data.

3.1.2. *Attacks against the computer system*

Apart from attacks against the host as a part of the agent system, there might be also attacks against the computer system the host is located on. In this case, the mobile agent system is used only as a possibility to attack something outside the agent system. The prevention of these attacks is especially important as it makes sure that the operation of a host does not compromise the computer system employed.

The possible attacks reflect all the attacks described above (like unauthorized usage of resources, denial-of-service, etc.), but now directed not against the host, but the computer system used. They also include all attacks between different computer systems, and between users and their computer system that are generally possible in any network.

3.2. *Existing approaches*

Today, there are numerous approaches that prevent at least some of the attacks by malicious agents. Therefore, a complete description of all of these approaches is outside the scope of this paper. To illustrate at least some of the approaches used, some mechanisms will now be described.

3.2.1. *Isolation of agents*

This approach ensures that agents cannot directly access other agents and the host data. To achieve that, mechanisms of the implementation language, or of the operating system are used. The first method is used in all Java-based systems like Aglets [LO 98] or Mole [BHR 98]. The latter method is used in the ARA system

[PS 97]. Here, every agent is executed using an own interpreter process, thus using the process isolation mechanisms to isolate agents from the outside.

3.2.2. *Authentication of agents*

To ensure that the host knows the owner of an agent a host has to authenticate arriving agents. This authentication can be done in one of two ways. Either the agent proves the knowledge of a secret known only to the owner (like in [RÖL 99], where a X.509 mechanism is used), or the agent is digitally signed by the owner. Then authentication takes place by verifying the signature. This second mechanism was proposed by [CGH 95] and implemented e.g. by [KT 99].

3.2.3. *Resource control*

To ensure that agents can access resources only in a controlled way, in mobile agent systems the sandbox mechanism is often used, implemented e.g. in Java. To achieve that, all possibilities to access resources (apart from computation and main memory usage) are directed to a central component (which is called SecurityManager in Java). If no security holes exist in the implementation of this mechanism, all accesses can then be controlled.

Resources can be controlled either qualitatively or quantitatively. A qualitative control allows the host to determine whether a certain mobile agent may access a resource. Resources that have to be controlled qualitatively are mainly security sensitive services like direct access to the operating system, to the file system, and to host management. Quantitative control allows a host to determine how much resources an agent may use. Quantitatively controlled resources include e.g. computation, memory usage, and communication.

There are different ways to control access to and usage of resources. Policies allow one to formulate rule systems that determine which agents may access certain portions of certain resources. An example for the application of such policies in mobile agent systems is presented in [KLO 98]. Another approach uses the early mobile agent system Telescript, where tickets are used instead of policies (see [TV 96]). Tickets are issued by the agent owners and allow restriction of the maximum usage of resources. A third class of approaches use money-like elements to control the quantitative usage of resources. In [GM 94] the authors describe an idea of having units that are called Teleclicks. These units are used by agents to pay for resource consumption. Teleclicks in turn have to be bought by real money. Artificial money is used by [BKR98]. Here the idea is to give every agent the same amount of an artificial currency and to let the host set the prices for resource consumption, thus achieving load balancing.

3.2.4. *Proof-carrying code*

In order to prove that actions of an agent conforms to a security policy defined by the host operator before the execution takes place, [NL 98] present an approach that they call “proof-carrying code”. In this approach, the owner or the programmer of an agent creates a digital proof of the results and consequences of the execution of a specific agent. This proof is transported with the agent. It can be verified by the host operator automatically.

After having examined the area of security between hosts and agents, the next section will describe the protection of mobile agents from attacks by malicious hosts.

4. Malicious hosts

In this section, the problem of malicious hosts is introduced. Further, approaches that tackle this problem are outlined roughly. Two more detailed analysis of the problem of malicious hosts can be found e.g. in [ST 98] and [HOH 98].

The fact that the runtime environment (the host) may attack the program (the agent), hardly plays a role in existing computer systems. Normally, the party that maintains the hosts also employs the program. But in the area of open mobile agents systems, an agent is in most cases operated by another party, the agent owner. This environment leads to a problem that is vital for the usage of mobile agents in open systems: the problem of malicious hosts. A malicious host can be defined informally as a party that is able to execute an agent that belongs to another party and that tries to attack that agent in some way.

To illustrate this problem we will use a small purchase agent as an example. The central procedure startAgent, which is called after the agent is initialised, could look like this:

```
public void startAgent() {
  if (shoplist == null) {
    shoplist = getTrader().
     getProvidersOf("BuyFlowers");
    go(shoplist[1]);
    break;
  }
  if (shoplist[shoplistindex].
   askprice(flowers) < bestprice) {
    bestprice = shoplist[shoplistindex].
                  askprice(flowers);
    bestshop = shoplist[shoplistindex];
  }
  if (shoplistindex >= (shoplist.length - 1)) {
    // remote buy
    buy(bestshop,flowers,wallet);
    // go home and deliver wallet
    go(home);
```

```
      if (location.getAddress() = home) {
        location.put(wallet);
      }
    }
    go(shoplist[++shoplistindex]);
  }}
```

Out of the number of possible attacks by a malicious host, two will be described:

"Stealing" the electronic money

If the value of the electronic money "coin" in a variable wallet is defined by the knowledge of the bitstring representation of it, the attacker can simply copy the bitstring and spend the money. The agent cannot see the attack when it happens; it may notice it when the agent tries to spend the money some migrations later.

Suppress other offers

There are two possible ways to reach this attack goal. First, the attacker can simply alter the value of bestshop to one that is preferred by the attacker. When it now also alters the value of shoplistindex, the agent will buy at the preferred shop the next time startAgent() is called. The second way is to manipulate the way the statements of the program are executed. If the if-statements of lines 9 and 15 always yield true, then the same effect is reached. Note that this manipulation does not leave traces as the code is not altered, just the way it is executed.

Existing approaches

One way to protect agents is to follow an organisational approach, i.e. to make sure that only trustworthy parties execute an agent. This can be realised either by maintaining the whole agent platform by only one institution, or by disallowing migration to unknown agent platforms. Currently, the first approach does not seem to be feasible since such a system would require a large number of vendors and clients in order to be worthwhile. The problem of the second approach is twofold: on one hand, trust is not a immutable attribute, but may change depending e.g. on the tasks an agent has to fulfil (although an airline as a big company is trustworthy, one does not want to depend on the goodwill of the company's host when comparing different flight prices). On the other hand, not all resources needed for the execution of a certain agent may be available on trusted hosts.

Another way to protect agents is to use special, trusted, tamper-free hardware (see e.g. [WSB 99]). To use them in the near future, at least two things are necessary: the need for such devices by platform providers and a manufacturer who builds these devices. Currently, no commercial application fosters this need. Therefore, today, there is no manufacturer thatproduces these devices. Another problem is whether trusted hardware allows the cost-effective execution by agents, but this aspect will not be discussed here.

If neither organisational mechanisms nor special hardware can be used, mobile agents have to be protected by software means only. Currently, there are two approaches that try to protect an agent from all major attacks. The first approach, which is called Mobile Cryptography [ST 98], aims at converting agents into programs that work on encrypted data (i.e. the operations use encrypted parameters and return encrypted results without the need to decrypt these data during execution). There are three problems which have to be solved before this approach can be used. First, currently, only rational polynomial functions can be used as input "agents" (recently, there are plans evolving to remove this restriction). Secondly, the used agent model does not allow agents that are protected by this approach to return plain text data to untrusted hosts (as this could lead to security problems). Thirdly, the efficiency of this approach is unknown (there is no implementation yet). The second approach based completely on software is called Time-limited Blackbox Protection [HOH 98]. Here, the agent code is obfuscated using techniques that are hard to analyse by programs. Since such an obfuscation can be broken by a human attacker given enough time, the agent bears an expiration date, after which the agent gets invalid. Successful attacks before this expiration date are impossible. In this approach, the input may be any agent, but there are problems that seem to prevent its employment in the near future. First, it is not known yet whether there are obfuscation techniques that are "hard" enough. Second, it is unclear whether the expiration date can be computed. Third, the efficiency of this approach is currently unknown (but it seems sure that at least the size of the agent increases significantly).

5. Conclusion

Security is an important aspect of using open mobile agent systems, especially in the area of electronic commerce. While other problems seem to be soluble today, the protection of mobile agents from attacks by their hosts is still not completely solved if only software means can be used. Readers interested in single aspects of mobile agents and security will find in [SecBib] a bibliography of papers dealing with this topic.

REFERENCES

[BHR 98] BAUMANN JOACHIM, HOHL FRITZ, ROTHERMEL KURT, STRAßER MARKUS, "Mole – Concepts of a Mobile Agent System", *World Wide Web*, Vol. 1, Nr. 3, 1998, p. 123–137.

[BKR 98] BREDIN JONATHAN, KOTZ DAVID, RUS DANIELA, "Market-based Resource Control for Mobile Agents", In *Proceedings of Autonomous Agents*, ACM, 1998, p. 197–204.

[CGH 95] CHESS DAVID, GROSOF BENJAMIN, HARRISON COLIN, LEVINE DAVID, PARIS COLIN, TSUDIK GENE, Itinerant agents for mobile computing. IBM Research Report RC 20010, IBM, March 1995. http://www.research.ibm.com/massive/rc20010.ps

[GM 94] GENERAL MAGIC, "Telescript Technology: The Foundation for the Electronic Marketplace", *General Magic White Paper*, 1994.

[HOH 98] HOHL FRITZ, "Time Limited Blackbox Security: Protecting Mobile Agents From Malicious Hosts", in Giovanni Vigna (Ed.) *Mobile Agents and Security*, Springer Verlag, 1998, p. 92–113.

[KLO 98] KARJOTH GÜNTER, LANGE DANNY B., OSHIMA MITSURU, "A Security Model for Aglets", in Giovanni Vigna (Ed.) *Mobile Agents and Security*, Springer Verlag, 1998, p. 1–14.

[KT 99] KARNIK NEERAN, TRIPATHI ANAND, Security in the Ajanta Mobile Agent System, Technical Report, Department of Computer Science, University of Minnesota, May 1999.

[LO 98] LANGE DANNY, OSHIMA MITSURU, *Programming and Deploying Java Mobile Agents with Aglets*. Addison-Wesley, 1998.

[NL 98] NECULA GEORGE C., LEE PETER, "Safe, Untrusted Agents Using Proof-Carrying Code", in Giovanni Vigna (Ed.) *Mobile Agents and Security*, Springer Verlag, 1998, p. 61–91.

[PS 97] PEINE HOLGER, STOLPMANN TORSTEN, "The Architecture of the Ara Platform for Mobile Agents", In Kurt Rothermel, Radu Popescu-Zeletin (Eds.): *Mobile Agents, Proc. of the First International Workshop on Mobile Agents MA'97,* Lecture Notes in Computer Science No. 1219, Springer Verlag, 1997.

[RÖL 99] RÖLLE HARALD, Authentisierung und Autorisierung für das Java/CORBA-Agentensystem MASA. Diplomarbeit, Institut für Informatik, TU München, 1999 http://wwwhegering.informatik.tu-muenchen.de/common/Literatur/MNMPub/Diplomarbeiten/roel99/HTML-Version/roel99.html

[SecBib] *Security in Mobile Agent Systems*. Online Bibliography. http://mole.informatik.uni-stuttgart.de/security.html

[ST 98] SANDER TOMAS, TSCHUDIN CHRISTIAN F., "Protecting Mobile Agents Against Malicious Hosts", in Giovanni Vigna (Ed.) *Mobile Agents and Security*, Springer Verlag, 1998, p. 44–60.

[TV 96] TARDO JOSEPH, VALENTE LUIS, "Mobile Agent Security and Telescript", In *IEEE Proceedings of COMPCON '96*, 1996.

[WSB 99] WILHELM U.G., STAAMANN S., BUTTYÀN L., "Introducing trusted third parties to the mobile agent paradigm", in: J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Mobile and Distributed Objects*, Lecture Notes in Computer Science, Springer Verlag, 1999, p. 471–491.

This page intentionally left blank

Chapter 2

Agents: the future of intelligent communication

Robert Ghanea-Hercock
British Telecom Laboratories, Ipswich, UK

1. Introduction

Within two years widespread access to third generation (3G) mobile phone networks, combined with advances in computing hardware, will transform the cellular phone into a multi-function computing and communication device. According to the Symbian consortium: *"By 2003 there will be 1 billion of these Wireless Information Devices on the Market"* [QRD 00].

This paper considers how intelligent, mobile software agents can enhance the support and application software required by wireless computing devices, in particular how the proactive, reactive and autonomous capabilities of mobile agents can improve the robustness and flexibility of a mobile computing and communication system. It is argued that future 3G mobile services will require smart, autonomous and proactive support software, which can seamlessly connect the mobile user with their home and work information services. In order to understand the application demands of 3G systems we first need a working definition of a wireless information device (WID) or communicator: *"A communicator is an information centric device with voice capability"* [QRD 00].

Already in Japan mobile phones outnumber fixed line installations and accessing the Internet via a mobile phone has become extremely popular. The incorporation of a simplified user interface in WID systems is also a factor in encouraging users with limited computer experience to adopt this technology. The services provided by such devices will include:

- Voice activation.
- The ability to run third party software.
- Full telephony and data/fax services.
- Hand writing recognition.
- Integrated Web browser, email and calendar applications.

The intense bidding for 3G licences, which occurred in the UK during April 2000, [CRA00], shows further evidence of the anticipated commercial potential of WID's. A good example of a current prototype device is the Symbian Quartz 3G platform [WUN 00].

A projection for growth is illustrated in Figure 1, showing the number of WID's for the next two years, indicating their rapid adoption as a link to Internet services.

The paper first reviews the design and features of mobile agent systems in Section 2, and then considers some specific applications to mobile wireless systems, in Section 3. Section 4 presents some preliminary results, with ideas for future work in Section 5. Section 6 presents our conclusions to the proposed development of mobile agent systems in WID's.

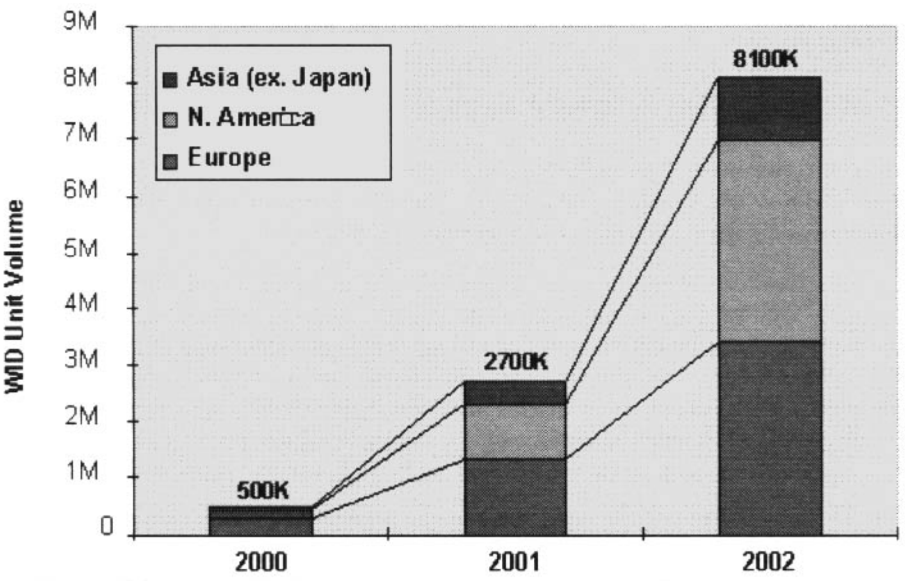


Figure 1. *Projected WID market growth, courtesy PSION/MOTOROLA [WUN 00]*

2. Mobile agents: an overview

Mobile agents represent one class of software agent, which can be defined by the following: "a component of software and/or hardware which is capable of acting exactingly in order to accomplish tasks on behalf of its user" [NDU 97].

Or alternatively: "a self-contained program capable of controlling its own decision-making and acting, based on its perception of its environment, in pursuit of one or more objectives" [JEN 96].

Nwana [NWA 96] defines the key properties of an agent as:

– *Autonomy*: the ability to function largely independent of human intervention;

– *Social ability*: the ability to interact 'intelligently' and constructively with other agents and/or humans;

– *Responsiveness*: the ability to perceive the environment and respond in a timely fashion to events occurring in it;

– *Proactiveness*: the ability to take the initiative whenever the situation demands.

A mobile agent incorporates these general agent properties and includes the ability to move between hosts within a computer network. To execute remotely, mobile agents move to a machine running a mobile agent server, which provides an interface to the underlying host machine, and facilities to transmit and receive agents. Several mobile agent packages currently provide this functionality, e.g. Voyager [OBJ 97], AgentTcl [KOT 97], Mole [STR 96] and Aglets [CLE 97]. The core of an agent server is the virtual machine, onto which mobile agents are loaded and executed. This helps hide from the developer the complexities of moving between, and executing on differently configured remote hosts. Mobile agent packages also provide facilities for high-level messaging (for inter-agent communication), and high-level methods for controlling agent behaviour, (for instance, moving the agent to a new host). One property of mobility, which is of potential benefit, is co-operation between agents. This concept is often closely associated with the agent metaphor, as it involves team activity in achieving a complex task through co-operation. Mobile agents have been proposed as a means for reducing network load and providing a dynamic and flexible platform for a wide range of applications e.g.

– Network management [BIE 98].

– Distributed processing [GHA 99].

For example, Lange in [LAN 99] has proposed several arguments in favour of mobile agent technology, such as:

– They reduce the network load.

– They overcome network latency.

– They encapsulate protocols.

– They execute asynchronously and autonomously.

– They adapt dynamically.

Unfortunately, significant problems have limited their commercial deployment beyond the research stage. In particular the problems include:

– Lack of agreed inter-agent protocols and standards.

– Slow emergence of suitable software and hardware.

– Poor security for mobile code.

This final point has been the principal obstacle to the application of mobile agent technology, i.e. a valid fear of weak security in such systems ([CHE 98] and [SAN 98]).

Design patterns

There are several principal design patterns that have been adopted by researchers in mobile agent applications. Examples of these are:

– Single homogenous agents that transport all code and data, and move between simple host servers.

– Two part systems with some functionality resident in the mobile agents and a set of essential and common services resident in each host server, [HOF 98].

– Complex heterogeneous systems with specialised mobile agents, and multiple services available at different servers, [GHA 99].

In the first example each mobile agent carries all of the classes and data it requires to perform the assigned tasks. This approach offers simplicity in design and reduces the load placed on the host servers and complexity of the host software. However, it suffers from major security problems, as the hosts are vulnerable to malicious agent code, depending on the restrictions the host attempts to place on the agents.

The third design case offers advantages in terms of allowing a very flexible distribution of agents, and minimises the footprint of each agent as they only carry classes essential to their specific task. It also allows for new agent types to be added to the system as required. The disadvantages arise in terms of increased complexity of the total system, and more complex management of the interactions between the agents. Similarly, it also increases the complexity of any security services and the process of managing them.

The second design case, of a two-part system appears to offer the best compromise between specialisation and simplicity, and has been adopted by several research groups (e.g. [HOF 98], [SAH 97]). In the following section we review how mobile agents may be applied to support applications in the mobile computing domain.

3. Wireless applications of mobile agents

Since the most frequently proposed application area for mobile agents has been portable computing systems [KOT 97], we need to review what features of mobile agents are of value in this context. One example study using mobile agents to deliver data to mobile computers demonstrated specific advantages of mobile agents for this application, i.e.:

– *Task continuation*: An agent can migrate to a host server to continue a processing task while the user is disconnected from the network.

– *Minimal connection use*: the agent can pre-process information at either the server or mobile device, in order to reduce the bandwidth required to transfer data [KOT 97].

An additional feature, which is well suited to the needs of wireless networked devices, is the ability of mobile agents to operate asynchronously, [HAR 95]. For example, mobile agent systems often include flexible messaging protocols such as asynchronous messaging, (Voyager: Objectspace [OBJ 97]). This can be summarised as providing a 'fire and forget' capability, i.e. the ability to launch information or processing agents which will return results when available.

In addition, even if high bandwidth is readily available in the near future, battery life on mobile devices is likely to be restricted. Hence the ability to relocate the processing element of an application to an available machine is still potentially useful. Secondly, wireless bandwidth is likely to remain expensive and hence worth using efficiently (increasingly likely given the cost of recovering the expense of acquiring 3G licences in the UK).

Example applications

Mobile device network support

A strong candidate application for mobile agents is mobile computer network management, ([SAH 97], and [MIN 98]). In this example, the agents can provide robust and decentralised network management architecture. Mobile agents can provide intelligent information processing and data filtering within distributed networks composed of transient connections. This can include services for automatic network management, as current static management systems (e.g. SNMP) are poorly suited to such mobile systems.

Mobile e-commerce

A second application domain for agents is mobile E-commerce. The emerging market for smarter mobile phones and integration of computing with telecommunication functionality has opened the possibility of mobile device based Ecommerce. People will use smart phone/pocket computers for online trade and service requests. For example a report in the Ecommerce Times highlights such trade as a major aspect of business to business trade in China [Eco 00]. France Telecom has also recently introduced a mobile e-commerce service [FRA 00]. Mobile agents can support such systems in the following ways:

Allow asynchronous message transfer of trading requests

A user can make a request to a mobile agent service via their mobile device. The agent then transfers into the Internet and queries multiple online vendors for the required service or goods. When the user reconnects to the network the agent automatically locates them and transfers the required data to their device. The option also exists of transferring the agent itself back to the device if the user wishes to interact with the agent. In this scenario the agent can perform work for the user while they are disconnected from the network.

Data intensive processing

Mobile agents can allow mobile users to access large online databases and perform asynchronous search and processing of data. This would enable remote product ordering and business to business transactions.

Dynamic network creation

Future mobile device networks based on Blue-tooth and 3G cellular protocols will also enable the formation of very fluid and dynamic networks of co-operating users over relatively short time scales. In this context mobile agents can be used to provide lookup, profiling, and user matching services. In such a domain the client-server model is no longer a useful solution.

A further application example is in co-operative multi-user applications, e.g. creating automatic calendar management and meeting schedules. A mobile agent can move between machines in a work group automatically updating users schedule data.

4. Implementation example

In order to demonstrate the potential bandwidth efficiency of mobile agents, a series of experiments were conducted to determine the relative advantages of using mobile agents to retrieve information from a remotely sited database, compared with using static remote method calls to the same database. (A comparison can be made with the work by Brewington [BRE 99], in which the time to transfer increasing numbers of documents was tested).

4.1. *Method*

The experiment was performed using a local area network of desktop Unix workstations running Solaris 5.5.1, with Java JDK 1.1.2. Using the Voyager library to create mobile objects a database consisting of N strings of ASCII data was constructed, each of random length, at one server in the network. Two types of experiments were performed, the first was to make remote method calls to the remote database and search for all strings of length n or less (where n<=N), which would then be returned to the local machine. In the second case a mobile agent was launched to the remote server and instructed to perform the same search operation, and then return with the data. In both cases an automated routine increased the length of string to be searched and restarted the experiment to test the effect of increasing data size on each process. A series of experiments were performed using different sizes of database, ranging in size from 200 Kbytes up to 16 Mbytes of data. The principal means for comparing the two methods was to measure the total time taken to recover the required data and return to the local machine, in milliseconds. The following figures show the effect on total time as a function of increasing both the size of requested data and the size of the remote database.

4.2. *Results*

From Figure 2 a clear distinction can be seen in the operational effect between using remote calls and performing a local search using mobile agents. The upper horizontal curve depicts that the transfer time using remote calls is independent of the size of the block of data being transferred.

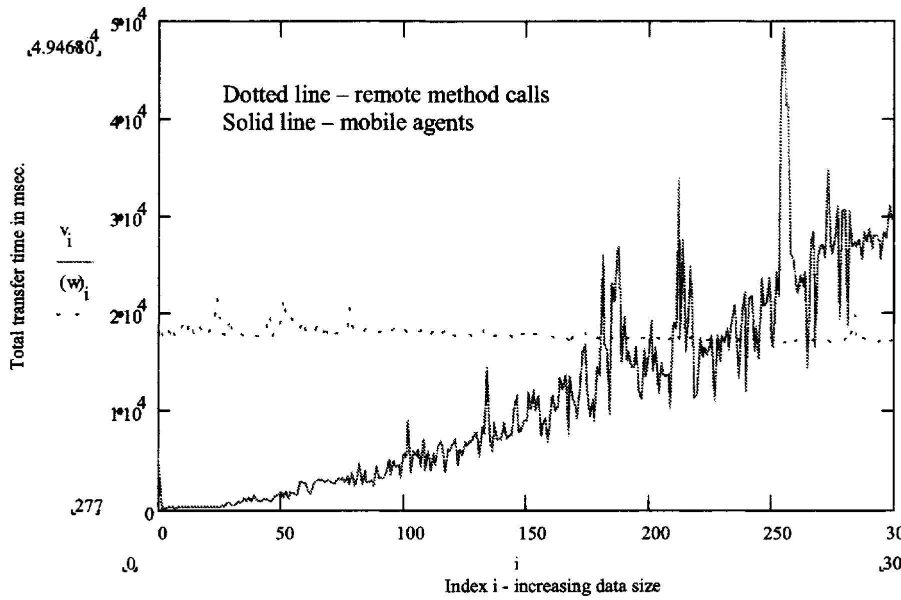


Figure 2. *Graph showing total transfer time as a function of increasing size of requested data, to a remote database containing 2Mbytes of data*

4.3. *Analysis*

From these results several conclusions can be drawn regarding the benefits of mobile agents in search operations. The principal result is that mobile agents are more efficient in performing remote searches than a remote method call up to a particular size of data transfer, which is a function of the size of the remote database. For transferring relatively smaller blocks of data (typically <100Kbytes), the advantage can be up to 40 times greater than a remote call method. The important point is that the advantage is greater for larger remote data sets, e.g. between the experiments for 2Mbyte and 8Mbyte of remote data the difference increases from 16 to 40 times.

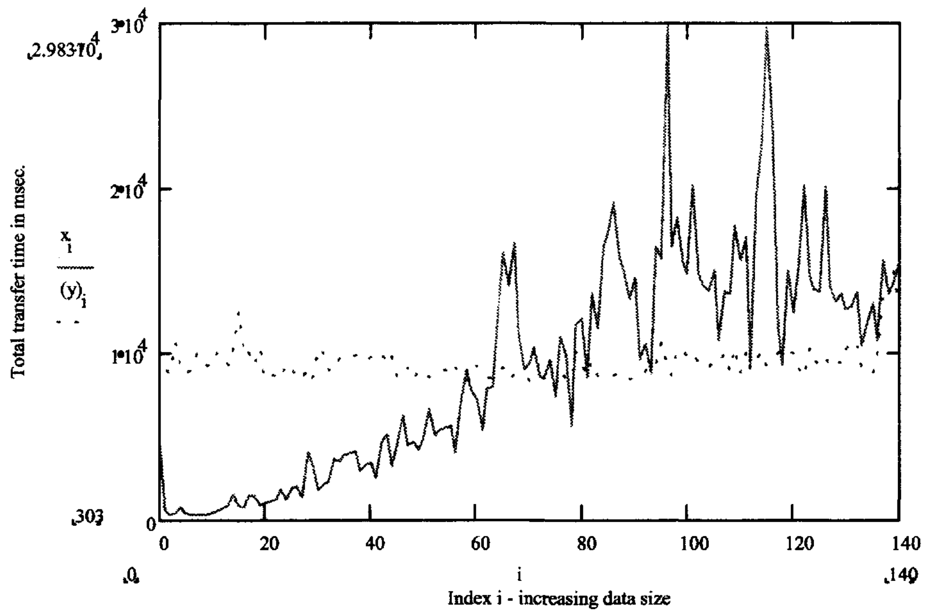


Figure 3. *Graph showing total transfer time as a function of increasing size of requested data, to a remote database containing 8Mbytes of data*

These results are supported by the conclusions of other groups such as [HUR 97] who also conclude that mobile agents are most productive when:

– The Remote server contains large volumes of data to be processed.

– There are hard real-time constraints.

– Communication is costly.

– The recipient of the mobile agent is located on a mobile terminal.

– The processing capability of the host is limited.

4.4. *Obstacles to mobile agents*

4.4.1. *Security*

The issue of securing host servers against malicious agents is still a subject of hot debate. However, the adoption of extended encryption and certification protocols can reduce the risk to an acceptable level for commercial transactions, as the real question in most security issues is one of risk management.

4.4.2. *Migration overhead*

The problem of additional processor load, especially due to the use of interpreted languages (Java and TCL), should be resolved through improvements in mobile hardware resources. In addition we can expect the emergence of custom hardware to support Java in native mode, which will greatly improve the performance benefit to be gained by local agent processing versus the cost of serialising and migrating the code and data.

5. Future directions

Mobile agents are clearly not a panacea for all mobile computing network problems. Serious issues regarding infrastructure and security remain unresolved. However, several factors point to the realisation of mobile agent based services in the wireless-computing domain.

– Pocket size computing devices can now support Java and possess the processing power to host software agents.

– Many 3G device users will lack computing experience and hence will require smart software to assist them.

– Wireless bandwidth is likely to remain an expensive commodity and hence services that can optimise its use will be of value.

6. Conclusions

By deploying intelligent mobile code in a wireless network environment, we can gain major advantages over the standard client-server architecture. The principal benefit is the ability to seamlessly integrate data and services between large numbers of users in a dynamic and robust manner. In addition the wireless mobile device becomes less dependent on the reliability of the network connection, even where high-bandwidth is available.

Secondly, the argument that unlimited bandwidth is a solution to all mobile network and application problems, does not reduce the benefits which mobile agents offer. No more than unlimited numbers of roads solves the transport problem in traffic systems. The high cost of accessing such bandwidth in the near future also makes efficient usage a priority in designing network processing software.

We have yet to realise an integrated network between palm devices, a home server, a work machine, and the Internet. Intelligent mobile code however, will assist in realising this goal and enable the formation of personalised smart sub-networks, within which groups of users can share data and computing services.

As with biological systems, in order for a particular species to flourish it requires a suitable ecological niche, the emerging third generation WID/Communicator market will provide just such a niche for mobile agents.

REFERENCES

[BIE 98] BIESZCZAD A., PAGUREK B., WHITE T., "Mobile Agents for Network Management", *IEEE Communication Surveys*, Vol. 1, No 1, p. 2–9, September 1998.

[BRE 99] BREWINGTON B., GRAY R., MOIZUMI K., KOTZ D., CYBENKO G., RUS D., "Mobile agents for distributed information retrieval", In Matthias Klusch, Editor, *Intelligent Information Agents*, Chapter 15, p. 355–395. Springer Verlag, 1999.

[CAS 98] CASTILLO A., KAWAGUCHI M., PACIOREK N., WONG D., "Concordia as Enabling Technology for Co-operative Information Gathering", *Japanese Society for Artificial Intelligence Conference* in Tokyo, Japan on June 17–18 1998.

[CHE 98] CHESS D.M., "Security Issues in Mobile Code Systems", p. 1–14, In *Mobile Agents and Security*, ed. G. Vigna, Springer Verlag, 1998.

[CLE 97] CLEMENTS P. E., PAPAIOANNOU T., EDWARDS J.M., "Aglets: Enabling the Virtual Enterprise", published in *ME-SELA '97*, available on-line from: http://www.trl.ibm.co.jp/aglets/index.html

[CRA 00] CRAIG A., "3G Licence bidding in UK": http://www.vnunet.com/News/601662 April 2000.

[ECO 00] Ecommerce Times report at: www.ecommercetimes.com/news/articles2000/000121-4.shtml, Feb 2000.

[FRA 00] France Telecom mobile commerce: www.techweb.com/wire/story/TWB19990813S0007, Feb. 2000.

[GHA 99] GHANEA-HERCOCK R., COLLIS J.C., NDUMU D.T., "Co-operating Mobile Agents for Distributed Parallel Processing", *Autonomous Agents Conference*, Seattle, USA 1999.

[GHA & BAR 99] GHANEA-HERCOCK R., BARNES D., "Disturbed Behaviour in Co-operating Autonomous robots", *Autonomous Agents Conference*, Seattle, USA 1999.

[HAR 95] HARRISON G. C., CHESS D. M., KERSHENBAUM A., "Mobile Agents: Are they a good idea?", *IBM Internal Research Report*, T. J. Watson Research Center, 1995.

[HOF 98] HOFMANN M., MCGOVERN A., WHITEBREAD K.R., "Mobile Agents on the Digital Battlefield", *Autonomous Agents Conference*, p. 219–215, Minneapolis, USA, 1998.

[HUR 97] HURST L., CUNNINGHAM P., SOMERS F., "Mobile Agents – Smart Messengers", *Lecture Notes in Computer Science*, Proc. First Int. Workshop, MA `97, Berlin April 1997, p. 111–122.

[NDU 97] Ndumu D., Nwana H., "Research and Development Challenges for Agent-Based Systems", *IEEE Proceedings on Software Engineering*, 1997, Vol. 144, No. 01, January 1997.

[JEN 96] JENNINGS N.R., WOOLDRIDGE M., "Software Agents", *IEEE Review*, January 17–20, 1996.

[KOT 97] KOTZ D. ET AL., "Agent TCL: Targeting the needs of Mobile Computers", *IEEE Computing Online*: http://computer.org/internet/icl1997/w4toc.htm, 1997.

[LAN 99] LANGE D., OSHIMA M., "Seven Good Reasons for Mobile Agents", *Communications of the ACM*, March 1999.

[MIN 98] MINAR N., KRAMER K., MAES P., "Cooperating Mobile Agents for Mapping Networks", Submitted to *Cooperative Information Agents* 1998, available from, lcs.www.media.mit.edu/people/nelson/research/routes-coopagents/.

[NWA 96] Nwana H., "Software Agents: An Overview", *The Knowledge Engineering Review*, 11, (3) p. 205–244, 1996.

[OBJ 97] "ObjectSpace Voyager and Agent Platforms Comparison", public white-paper from http://www.objectspace.com/Voyager/voyager.html Oct 1997.

[QUA 00] Quartz Reference design, http://www.symbian.com/news/featured/quartz_feat.html. June 2000.

[SAH 97] SAHAI, A., MORIN, C., BILLIART S., "Intelligent agents for a Mobile Network Manager (MNM)", In *Proceedings of the IFIP/IEEE International Conference on Intelligent Networks and Intelligence in Networks* (2IN'97), September 1997, Paris, France.

[SAN 98] SANDER T., TSCHUDIN C. F., "Protecting Mobile Agents Against Malicious Hosts, Security Issues in Mobile Code Systems", p. 44–60. In: *Mobile Agents and Security*, ed. G. Vigna, Springer Verlag 1998.

[STR 96] STRASSER M., BAUMANN J., HOHL F., "Mole – A Java based Mobile Agent System", *Proceedings of the 2nd ECOOP Workshop on Mobile Object Systems*, Austria, 1996.

[WUN 00] WUNKER S., "Quartz Presentation to Developers" Conference and Exhibition in Santa Clara, California, February 2000, available at: http://www.epocworld.com/events/US2000/UStalksmenu.htm

Chapter 3

Agents in telecommunication-based services

Mihhail Matskin
Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway

1. Introduction

In agent technology there is great interest in different application areas related to networking. This interest is based on a practical reason: there is no yet widely accepted paradigm for developing software in a distributed decentralized environment. The problem is that most existing and well-developed programming or system design paradigms were not designed for distributed system development, and they do not consider an environment to be open. When we refer to an open environment we assume that the computing environment, where the system operates, can not be described completely, and it may be dynamically modified. In the case of decentralized and distributed information sources in the network, an environment cannot be specified completely in many cases because it is problematic (if possible at all) to describe all available information sources in the network and to specify precisely and completely all possible actions and changes. The distributed environment is not closed any more, and in many cases there are no restrictions to affecting the environment and for being affected by the environment. By 'environment' we assume both computing resources and participants of a distributed work. There have been attempts to extend former programming and design paradigms in order to cover dynamics and modifications. However, most of the attempts are hardly practically applicable, and they are often outside the main streams of former paradigms.

We think that in order to have a solution for the distributed environment a new agent-based paradigm can be constructively exploited [BRA 97; JEN 97]. The basic idea behind agents can be formulated as follows. The approach to work in an open environment can be similar to the human mode of behavior in an open physical world. This approach is based on perceiving the world, having a model of the world and behavior and having intentions/motivations to be fulfilled by implementing corresponding goals. This is, of course, a schematic view of human behavior; but it emphasizes the autonomous, pro-active and distributed nature of behavior in the case of open world, and this approach is employed in agent-based systems.

Development of agent technology is very much based on research in artificial intelligence and distributed computing. However, attention and practical interest in agents was initiated by the development of network technology. Thus networking is an essential source for developing interest in agents and it is a vehicle for them. At the same time networking itself is an attractive area for agents. The potential of such applications is very high but there are not too many practically successful applications, and the benefits of agents still need to be demonstrated or justified.

In order to contribute to such justification we try to consider what the essential agent characteristics are; we try to rectify agent features and to see how they can be useful for networking related purposes. In particular we consider an application of agent technology in telecom-based services.

The outline of this paper is as follows. First we consider software agent characteristics and dimensions (perspectives). After that we discuss a general way of agent utilization in telecom-based services. Finally, we consider some particular examples of possible usage of agents in telecom services both from individual and group agent perspectives.

2. Software agents: basic properties

The term "agent" is overloaded and it is often used differently in different research communities. However, there is better agreement of what is understood by the term "software agents". Software agents are usually explained by presenting their basic characteristics. Here we summarize some of these characteristics by adopting definitions from [WOO 95; NWA 96] as follows:

– Software agents operate on behalf of their creator. This means that they represent a human or another agent in a computer environment.

– Software agents can operate without intervention from outside, and by doing so employ autonomous behaviour.

– Software agents react to changes in the environment. This means that they can perceive and affect the environment.

– Software agents employ pro-active behaviour. This means that they can take initiative in order to achieve their goals.

– Software agents can communicate with other agents using an agent communication language, and by doing so can perform negotiation and co-ordination.

The last four properties are also referred to as *weak intelligent agent properties*. There are some other properties usually attached to agents. From our point of view among them mobility and learning are the most important ones. However, we agree with those who consider these properties to be strong agent properties and not exclusive to agents but also to non-agent systems.

The border between mobile agents and intelligent agents is quite artificial, and it often exists due to the different backgrounds of corresponding research communities rather than due to different basic foundations. We admit that mobile agent platforms are better developed. However, we think that underlining mobility as the sole basic agent characteristic is a simplification of the whole agent paradigm, and this may be misleading for selecting potential applications.

Learning is also referred to as an intelligent ability and it is desirable to have it in agent systems. However, we think that agents may be not very intelligent (do not employ advanced adaptive abilities) but even in this case they may allow new approaches to distributed and decentralized computing.

The following two dimensions are usually considered in software agent research:

– Individual perspective – considers individual agents which do not cooperate, coordinate or communicate with each other (the agents may not know of the existence of other agents).

– Group perspective – multi-agent systems that are concerned with the behavior of a collection of individual agents aiming at solving a given problem.

Considering individual perspective, agents employ autonomy, pro-activity and reactivity but they do not use communication as a means for common problem solving. In the group perspective agents also employ cooperative agent behavior.

In the next sections we consider the use of agents in telecommunication services from these two perspectives.

3. Agents in networks

To utilize the agent characteristics mentioned in Section 2 in telecommunication-based services let us first consider individual perspective.

The individual agent perspective can be directly applied to telecom-based services as follows: *Agents can be autonomous (decision making) components representing a customer in the network.*

A customer of a network can be, for example, a user of mobile (cellular) handset. A good association in this case is a personal assistant that helps a user to be aware of important events or who helps to keep his/her profile dynamically updated.

A basic difference between personal assistants and the agent for mobile device customers with customer profile databases is that the agent is an active entity rather than a passive record in the database to be used by some processing software. At the same time this differs (has a special characteristics) from the traditional image of computer personal assistant/agent, because it represents the customer off-line and uses only a short time interval for connection to the customer. In this case autonomy and pro-activity should play very important roles. Intelligent abilities of such agents (or in other words decision-making abilities) may vary quite a lot from simple pre-programmed actions to highly adoptive behavior via learning.

The agent group perspective can be applied as follows: *Customers and service providers can be represented by their individual agents who communicate, negotiate and coordinate their activities in order to reach their own goals.*

This perspective assumes flexible adjustment of an agent's preferences and demands in order to find an acceptable (with respect to some criteria) solution. A communicative aspect of agency becomes very important and plays a central role in this situation (of course, it does not deny other features which are essential to individual agents). In other words, in addition to operating in a resource-based environment and communication only to the creator (human), socially communicating agents exchange information with other similar agents.

We can summarize the above assumptions about usefulness of agents in network services as follows:

– Application of the individual agent perspective in network services assumes development of software which operates autonomously and represents an active customer profile. By 'active profile' we assume ability of such software to perceive the computing environment, to react to events in the environment and to predict customers' needs by employing pro-active reasoning.

– Application of the group agent perspective in the network services assumes existence of a community of individual agents in the network. These agents represent entities (customer, service provider or someone else) in the network environment. It also assumes that software programs, which implement individual agents, employ some model of communication or in other words can understand each other's messages.

In the next sections we illustrate possible applications of agents in telecommunication-based services by some examples of our prototype systems.

4. Individual agents (WAP-based services)

As an example of individual agents in telecom services we consider agents for support of WAP (Wireless Application Protocol)-based services.

WAP [WAP 00] is a rapidly growing area of mobile communications. More and more companies support WAP/WML-pages for their customers. It is a new field and previous solutions (as for example WWW-based solutions) cannot be copied directly to WAP. The basic difference with previous solutions is that the users of WAP are not assumed to surf through WML-pages, not knowing what they are looking for but rather will surf in order to cover their needs. It is also possible that WAP users will not want to surf over the Internet at all but just to have direct access to a specified information. An extremely important requirement of telecom services from the

WAP perspective is a high level of *personalization* in WAP-available information. We list the following reasons:

– high cost of mobile communication,

– poor experience of most of WAP users in Internet usage,

– limited expressive capability of WAP devices.

It is likely that some limitations of WAP devices will be relaxed in the future, but at the moment we should take them all into consideration. A consequence of the first reason is that the mobile customer connection time to the network should be minimized, and that as much as possible work for surfing should be done off-line. Another consequence of the first reason is a requirement for a high precision of searched information because displaying a large amount of information (usually provided by WWW search engines) on a small WAP device screen may be very inefficient. In other words compactness and focusing of information are very desirable.

The way WAP-services are now supported by service providers is based on proposing some predefined set of services and on making them available via a WAP-handset menu. A typical set of services includes news, banking and finance, newspapers, travel information, currency exchange rates etc. These services are a quite reasonable selection but they are not personalized enough, and they, of course, cannot cover needs of all customers. Our concern is to make services personalized, adaptable and pro-active. In order to justify our solution to the problem we consider some possible scenarios of providing WAP-based services.

The most constructive way of adopting and customizing services is to deliver up-to-date information to the customer in a convenient place and time. This means that in the case of customer's communication with a WAP device a WML-deck/card (as a basic means for WAP-based interaction) should always reflect what the user wants to see at that precise moment. In this way it should be as easy as possible to retrieve the information without typing advanced queries or long links in order to get to a certain destination. This can be illustrated as follows.

If the user always reads the news on his way to work in the morning then the best way would be to present an overview of all the latest (and still not read) news from a favorite news agencies as either a starting card, or a card which is only on the next level in the hierarchy. During the day user preferences or requirements for information could be changed to, for example, more business-oriented information in daytime or TV-programs in the evening on the way home. Then, the starting WML-card should reflect such a shift of focus of interest, and it should allow access to preferable information to be as easy as possible, while the previous focus of interests (which could be not as important anymore) should not correlate with the current focus.

Another situation in the usage of WAP may reflect a need for permanent awareness of important events and newly available information of great importance for the customer. As an example of important information we can consider quotas from the stock market, and as an important event we can consider changes in particular stock quotas beyond a predefined threshold. As a summary of the scenario requirements we can say that ability to facilitate dynamic shift of focus of required information and the dynamic reorganization the WML-cards would be of a great interest.

Each of the above-mentioned scenarios could be specialized for different customers and situations. It seems to be impossible to create a WAP portal which accommodates all the different needs customers might have. It may also be very hard to manually create many portals which are meant to be updated regularly and which reflect different customer needs. From the customers' point of view, the best way to reach the documents of interest as easily as possible could be for them to be available through own personalized portal. Since the customers are interested in updating their portals infrequently, there should be somebody who keeps it constantly up-to-date. If we take into account agent properties from Section 2 then we can see that such job is just perfect for software agents.

An agent can be a constantly running piece of software. When nothing happens, it will hibernate but when a change in the source for the portal or some event occurs, the agent wakes and pro-actively updates the information on the WAP portal. When the user connects to the personal portal s/he will get required information with no waiting for completion of queries. If we refer to agent properties discussed in Section 2, autonomy, reactivity and pro-activity will play an important role in implementing such agents. A customer profile presented in a system should be a key source for implementing pro-activity. In a more complex case the agent may have the ability to learn by monitoring the customer's actions. Such monitoring may be a source for updating the customer's profile and for deciding on a preferred time for availability of particular information.

In order to test our ideas we developed a JAFA (Java based Adaptive Filtering WML-Agent) prototype system. We use JAFA for illustration of utilization of agent technology in WAP-based communication. The main idea behind JAFA is to create an intelligent agent which is

– highly adaptive to extracting the interesting information from various commonly updated WML-pages,

– able to locate, collect and present interesting information in a WAP device-oriented view,

– able to reorganize an order and priority of information representation,

– able to simple pro-activate behavior,

– able to off-line process user requests.

JAFA inhabits the Internet, collects information from different sources in the Internet and provides a WML-deck which the customer can access via the WAP gateway as shown in Figure 1.

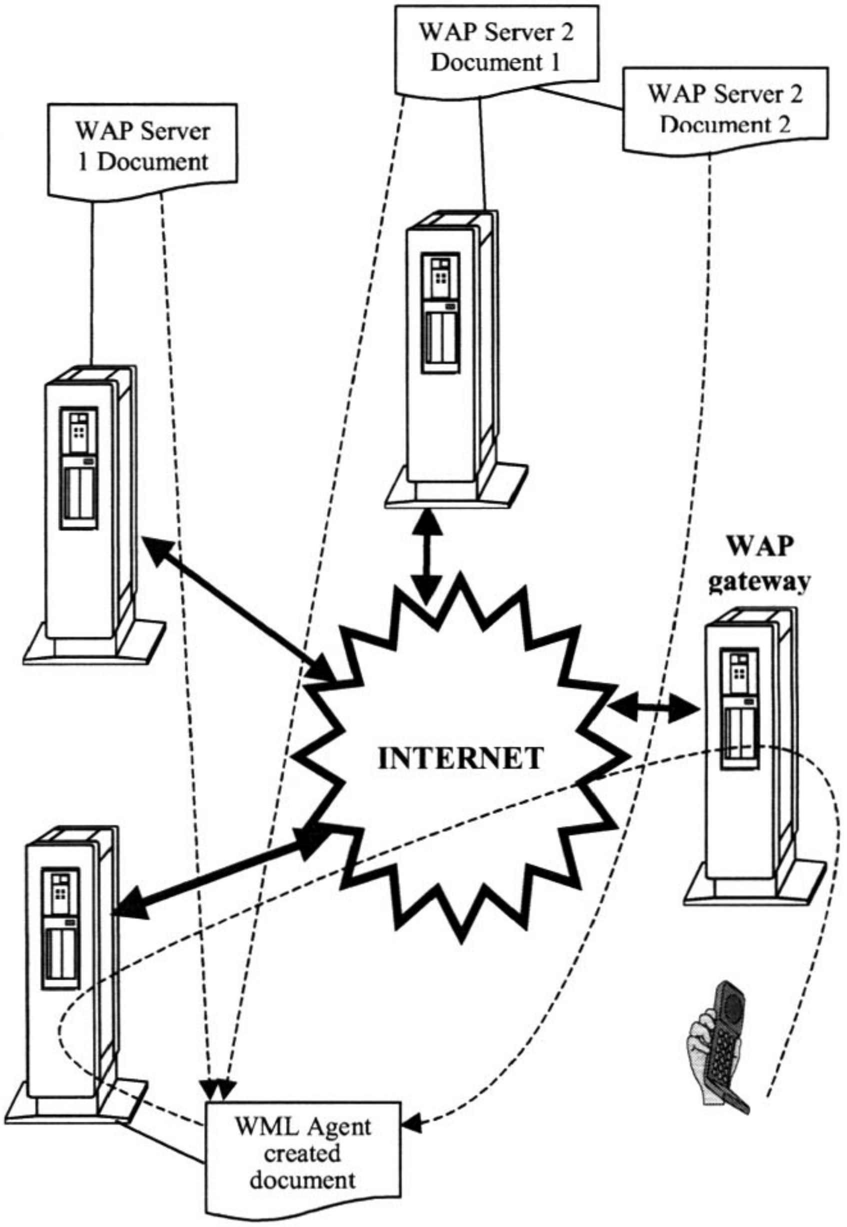


Figure 1. *Personal agent in the network*

Customers should be able to design their starting pages telling JAFA what information they would like to have and where to find it. Since we assume that many customers are inexperienced in WAP, they probably may find this a hard and time-consuming task. In order to help the customer in creating personal starting cards, it should be possible for a service provider to use JAFA for creation of certain decks targeted to a large group of users. This could be, for example, creation of decks with the target group being customers interested in fashion and music. Some other decks would cover target group of peoples interested, for instance, in technology and economics. By combination of these decks it would be possible to create a profile for customers interested in music and technology but not in economics and fashion.

The JAFA system may be divided into two parts:

– Graphical User Interface (GUI).

– Processing part.

The GUI part provides means for creating, updating and customizing user cards and requirements.

The processing part carries out search, adaptation and creation of WML decks/cards. In other words, the processing part of JAFA allows one to display the required information according to customers needs, to process from the customer off-line and to present results of proceeding in the form of WML-cards.

Another information retrieval feature that is supported by the JAFA agent is information filtering. The user can describe what particular kind of information from predefined WML-cards and WWW-pages s/he is interested in. This can be described by a regular expression, and the agent will filter updated information in the pages according to the expression. Such filtering allows a more personalized and selective search that may decrease the amount of deliverable text. This is especially useful with a limited size of screen of WAP devices.

The JAFA system employs basic agent features such as autonomy, reactivity and a simple pro-activity. In the current implementation the pro-activity is restricted to ability to a simple reorganizing of WML-decks according to results of processing (for example, by putting the hottest news on the top). However, further development of this feature is our prioritized work in a new JAFA system version.

5. Support of group agent work

The next step in applying agents in network services (the group perspective) is to allow them to communicate with each other. For example, agents may represent customers and service providers of telecom services. Each of the agents personalizes some interests and has goals to be achieved. In particular, the customer is interested in the comparison of available services and choosing one of them according to price-quality relation, and the service provider can be interested in involving a greater number of customers into its service network and to make a reasonable profit. In spite of the fact that these interests may be antagonistic (non-cooperative) the agents may reach mutually acceptable solution via information exchange and negotiation (proposals and counter-proposals). In this case the process of service selection may be treated as a trading process. We consider how such trading process between agents can be supported. As an example we use a virtual shopping mall case where agents represent buyers and sellers.

We develop a multi-agent platform for cooperative work support which was used for modeling trading between agents (in particular, it was used for virtual shopping mall modeling).

The general idea of cooperative agent work support is based on a concept of Agora [MAT 98; MAT 99] – a facilitator of cooperative work. Agora can be considered as a node where agents can register their own skills and tasks as well as required skills and tasks. Agora is responsible for managing a context for cooperative work of registered agents, matchmaking requests and offers, constraining communication between agents and supporting negotiation and coordination among agents. In other words Agora can be considered as an infrastructure where agents can get help in common work. In the case of multi-agent activity we can present a group agent work as a network of Agoras created for some particular purpose (see, for example, Figure 2).

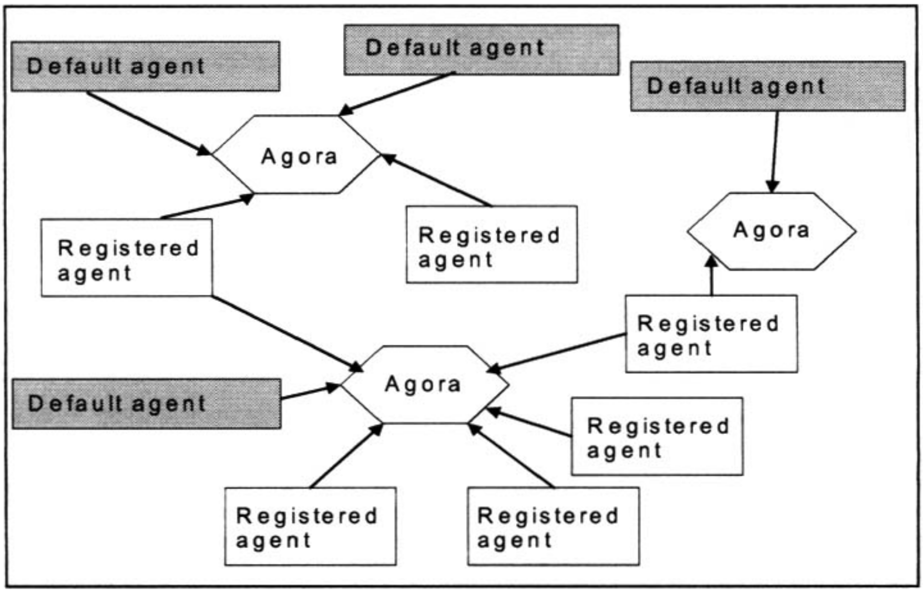


Figure 2. *Agoras network*

Using our approach, the shopping mall scenario will include agents and Agoras (cooperative points) to support common work. First we consider possible agents in the shopping activity. They are as follows:

– Customer Agent represents customer. A customer can have more than one Customer Agent. In this case there has to be an Agora where these agents can coordinate their work.

– Shop Agent represents a seller (or service provider). A shop can have more than one seller (Shop Agent). In this case there has to be an Agora where these agents can coordinate their activities.

– Mall Agent represents a mall manager. This agent can be considered as an information desk manager that may keep an overview of customers and shops participating in the virtual mall. Mall Agent can also perform customer registration service.

– Negotiation Agent represents a negotiation manager (for example, auctioneer in auction or manager in the Contract Net Protocol [SMI 80]). It manages the negotiation process between customer and shop agents in a neutral way.

– Marketing Agent represents a marketing manager. This agent can register itself at Customer Agoras and perform active marketing towards the Customer Agents locally. This can be especially useful in the case when Customer Agents do not know which shop they should visit in order to find a particular product.

– Bulk Purchase and Multiproduct Agents represent mangers of agent coalitions. Agents can form different types of coalitions. In particular, coalitions can be created for a product information exchange or for uniting buying resources in order to get a better price for a particular product or multi-products. These particular coalitions are coordinated at Bulk Purchase and Multiproduct Agoras. Shops can also create different coalitions, in particular, for coordination of price policies or for providing better information to customers.

Cooperative points for agents' work are represented by the following Agoras:

– Customer Agora supports coordination of different Customer Agents which represent different customer profiles and preferences. Marketing Agents may be also registered at this Agora.

– Shop Agora supports group work between Customer Agents and Shop Agents in a particular virtual shop. Each shop is represented by Shop Agora with a set of Shop Agents registered at it. The Shop Agents provide registered Customer Agents with information about products and try to sell their shop's products through negotiation with the Customer Agents. In addition, the Shop Agents can also be registered at the Multiproduct or Bulk Purchase Agoras, and they may enter into agreement with other agents and other shops registered at these Agoras.

– Mall Agora supports group work of the shopping mall.

– Bulk Purchase and Multiproduct Agoras support group work of corresponding agent coalitions.

Figure 3 presents a simplified example of a Virtual Shopping Mall as a network of interrelated and connected agents and Agoras.

The above-considered shopping scenario is a more advanced scenario than it seems to be as practically used in trading of telecommunication services. However, a telecom scenario can be easily derived from the above agents and Agoras. In particularly we can relate Shop Agoras to service providers, Shop Agents to particular services and Mall Agora to a set of available service providers.

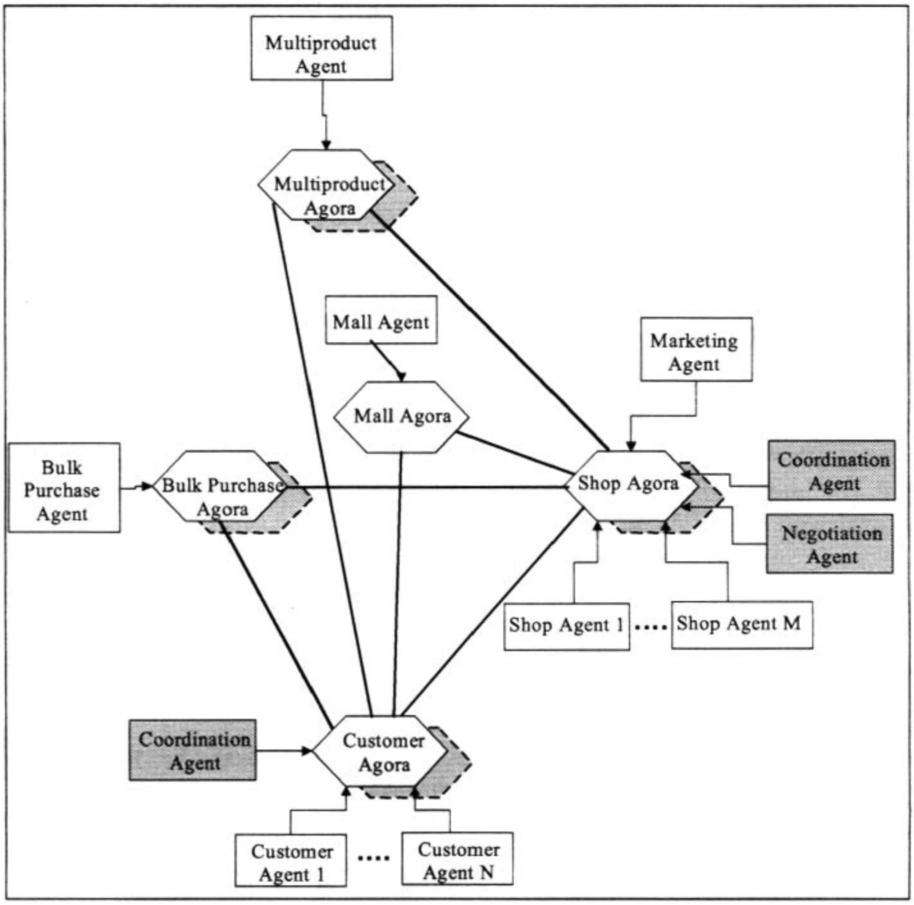


Figure 3. *Agents and Agoras in a Virtual Shopping Mall*

6. Conclusions

We have tried to demonstrate some possible means of application of agents in telecommunication-based services both from individual and group perspectives. Basic benefits of such application can be summarized as follows:

– Agents allow better personalization of services. Because agents always represent their creator in a computational environment, they inherit his/her goals, tasks and profiles, and this enforces creation of highly personalized services.

– Agents may lead to better fairness, quality and higher user satisfaction of services because of a higher quality of discrepancy resolution and more detailed and personalized decisions.

– Agoras give practical support for negotiation and coordination in the group agent system by providing an infrastructure and templates for cooperative points. Agoras provide support for particular cooperative work as well as accumulation and re-use of knowledge about such support.

– Agents may introduce higher levels of adaptation, dynamics and awareness of telecom services.

Acknowledgements

The work described in this paper is partially supported by the Norwegian Research Foundation in the framework of the Distributed Information Technology Systems (DITS) program and the *ElComAg* project. I also would like to thank Runar Bell, Ole Jørgen Kirkeluten, Svenn Bjarte Krossnes and Øystein Sæle for their work on implementing JAFA and Agora systems.

REFERENCES

[BRA 97] BRADSHOW J.M. (Ed.), *Software Agents*, Menlo Park, CA, AAAI Press/The MIT Press, 1997.

[JEN 97] JENNINGS N.R., WOOLDRIGE M.J. (Eds.), *Agent Technology: Foundations, Applications and Markets*, Springer Verlag, 1998.

[MAT 99] MATSKIN M., Multi-Agent Support for Modeling Cooperative Work, *In: T. Yongchareon, F.A. Aagesen, V. Wuwongse (Eds.) Intelligence in Networks, The Fifth International Conference SMARTNET'99*, 22–26, Thailand, Kluwer Academic Publishers, 1999, p. 419–432.

[MAT 98] MATSKIN M., DIVITINI M., PETERSEN S.A., "An Architecture for Multi-Agent Support in a Distributed Information Technology Application", *International Workshop on Intelligent Agents in Information and Process Management on the*

22nd German Annual Conference on Artificial Intelligence in Bremen (KI-98), TZI-Bericht Nr. 9, Germany, September 15–17, 1998, p. 47–58.

[NWA 96] NWANA H.S., "Software Agents: An Overview", *The Knowledge Engineering Review*, 11(3), 1996, p. 205–244.

[SMI 80] SMITH R., "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", *IEEE Transactions of Computer Science*, 29(12), 1980, p. 1104–1113.

[WAP 00] WAP Forum Specifications [Online]. Available at: http://www.wapforum.org/what/ technical.htm

[WOO 95] WOOLDRIGE M.J., JENNINGS N.R., "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review*, 1995. 10(2), p. 115–152.

This page intentionally left blank

Chapter 4

Mobile agents in mobility support and service selection

Kirsi Valtari
Oy Radiolinja Ab, Helsinki, Finland

1. Introduction

Mobile agents are computer programs, which are capable of moving in the network and acting on behalf of someone else. They are controlled by an authority, which can be a user or a computer program. 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 [ANH 98]. While moving, the agent may maintain its state and while visiting remote locations it may communicate with its owner, with other agents or its environment in general.

Mobile agents are expected to bring advantages within networked applications by reducing communication by migrating to the remote node instead of making a series of remote communication calls. This model of communication supports mobile computing, since the agent can move to target domain whenever connectivity with sufficient bandwidth is available. Agents also enable customisation of service before delivery through the network, which is a scenario valuable in information filtering type of tasks. In addition to the optimised performance gains, the use of mobile agents offers a new paradigm to model distributed software engineering and some authors see it as a successor to the client-server paradigm in networked computing. A mobile agent with learning capabilities is often called a *mobile intelligent agent.*

Two major efforts targeting at standardisation have been initiated addressing the interoperability of agents. The organisations active in this area are the Object Management Group (OMG) and the Foundation of Physical Intelligent Agents (FIPA) [OMG, FIP]. OMG is focusing on agent mobility while FIPA is focusing on agent intelligence. The specifications developed so far are not complete, requiring validation, further enhancements and most importantly integration.

The weaknesses of mobile agent technology relate to the security problems and the lack of standards. The security of mobile agents is discussed in [GRE 98] and [CHE 98], where it is pointed out that in addition to traditional threats of a distributed system, new risks arise. In addition to potential misuse of the host by the agents, a severe problem is how to protect the agent from a malicious host as described in [SAN 98]. The research on security problems has started, but pragmatic solutions are still to be developed.

The telecommunications sector is a potential application area for mobile and intelligent agents, which are often envisioned to optimise and add dynamics to the large distributed systems of IN and TMN as described in [ALB 98]. In relatively early work within GMD Fokus agents were proposed to be used in IN to dynamically download customised service scripts from service control points to the service switching points [KRA 96]. The research centre is currently active with projects focusing on management of ATM and SDH networks [IMA]. Both TMN and TINA use object-oriented modelling and use a concept of agent in their architecture, which suggest that the use of mobile agent technologies should be a natural path to follow.

In the past two years the agent paradigm and emerging agent technologies have been considered a key for the implementation of flexible and scalable solutions for an open services market within European research. In ACTS programme 14 projects were launched in Spring 1997 to explore the usage of agent technologies in the areas of service control in fixed and mobile networks, telecommunications management, electronic commerce, multimedia applications. The collaboration forum of the projects is called CLIMATE [CLI] and its web pages provide a good summary of the projects and their work. In the following the projects most relevant to this work, in addition to MONTAGE [MON], are briefly described.

The CAMELEON project has applied agent technology in the area of mobile networks, by providing a unique service set to the user, referred as Virtual Home Environment (VHE), and by enabling users to roam and use the same services in the same way on any Universal Mobile Telecommunications System (UMTS) network [CAM]. A mobile VHE agent has been introduced, which corresponds to the functionality of visited and home location registers of GSM [HAG]. Although the VHE agent has been designed for UMTS and IN based environments, it bears a clear resemblance to the user profile agent introduced in this paper. Two of the CLIMATE projects, MIAMI and FACTS have focused on the agent platforms and the validation and contributions concerning the evolving agent standards. MARINE project has enhanced IN via distributing intelligence in the network by applying distributed object and agent technologies to the network environment. In particular, the project has considered agent-based deployment of IN service logic offering services to both fixed and mobile users. The SCARAB project has focused on the synergy between smart card and agent technology in an open, distributed and secure service architecture.

The current keen interest in applying agents within telecommunications is reflected by a number of agent projects within EURESCOM arena as well. According to the web pages [CLI] EURESCOM Project P712 shares the interest area of this work with the focus on intelligent and mobile agents and their applicability to service and network management.

Principles similar to this work have been recently presented in [LLO 99], which discusses the possibility of implementing UMTS VHE with agents. There it is reasoned that personalised VHE could be realised with two approaches: 1) building on GSM and later UMTS or 2) assuming that seamless mobile internet will become a reality and service access and roaming will be built on top of IP. This articledraws a distinction between network operators and service providers. The work is however in its initial stage and no specific designs are presented. In [THA 99] a notion of an ideal user mobility is presented and a design with mobile agents is suggested where a user agent or a group of dedicated user agents may move to the visited domain.

2. An architecture supporting mobility

In the following, the service architecture together with the mobility context envisioned in this paper are described. Figure 1 presents the business model, which is a simplified modification of the TINA business model. The consumer accesses network services through a retailer, who has the capability to deliver services and the infrastructure to manage end user accounting and charging. The market is competitive; once the user has been authenticated as a valid customer, the user may dynamically choose a service offer from a number of retailers. Each retailer may be promoting content material and services originating from a number of content providers.

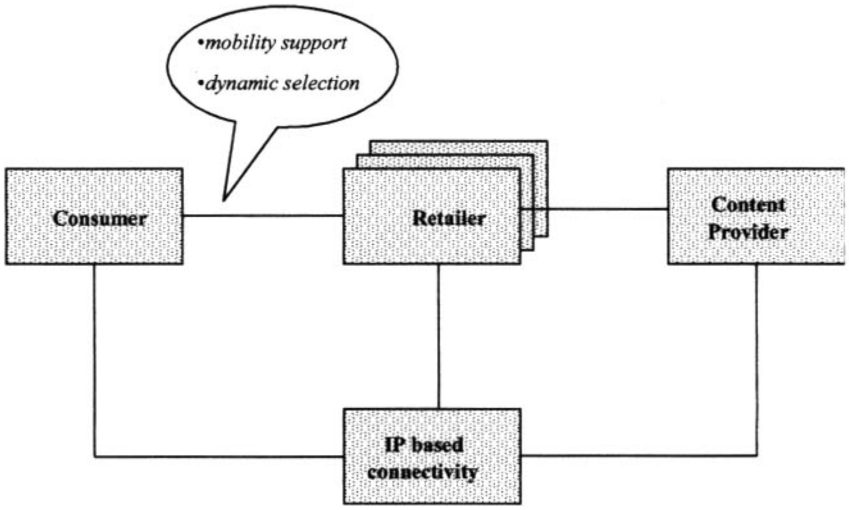


Figure 1. *The business model used in this work with the remark highlighting the mobility and selection focus*

Communication between the stakeholders is enabled by IP connectivity and TINA communication management is not used. However, when it comes to real-time stream communications, a more controlled communication scheme would be needed eg. Session Initiation Protocol (SIP) by IETF workgroup on Multiparty Multimedia Session Control [MMU].

Personal mobility can be considered in terms of federating stakeholders (in particular retailers). In order to provide access in foreign retailer domain, a concept of visited domain is used in a similar fashion as in Universal Personal Telecommunications (UPT) and Universal Mobile Telecommunications System (UMTS) [UPT, UMTS]. A visited retailer within TINA architecture has been previously introduced by ACTS DOLMEN project [DOL 98]. In this work the visited retailer concept is used and the design is further developed to take advantage of code mobility and to allow more flexibility in terms of choosing a retailer. Mobile agents are used to support roaming between retailer domains. Additionally, agents are applied in implementing dynamic service selection among competing offers from different retailers. The potential retailers who could be able to provide an instance of optimal service at a given moment are called candidate retailers. The following roles are given to the retailers in this work:

Home retailer: a specific retailer, to whom a user has a subscription for one or more services. The home retailer 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 retailer per service.

Visited retailer: the initial access session level contact point in foreign network environment. It is the retailer offered by the communication network (e.g., ATM, Internet or TINA kernel Transport Network – kTN). The visited retailer may also provide the service (thus acting as a candidate retailer). In the published papers (listed in Appendix A) the visited retailer is also called a default retailer.

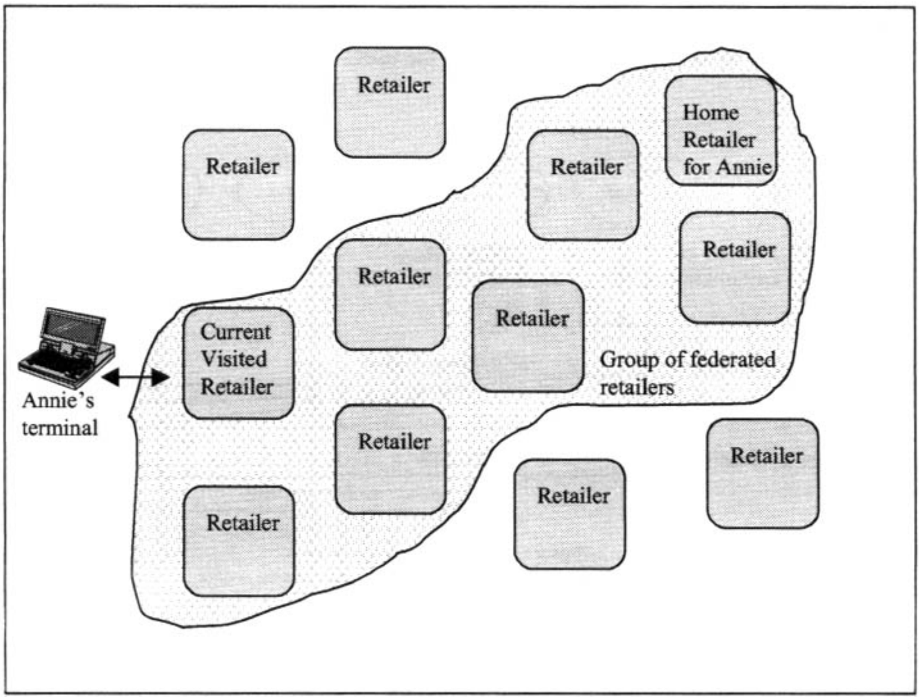


Figure 2. *Retailers in different roles as seen from the mobile user's point of view*

Candidate Retailer: any retailer, who is not the home retailer and with whom the user can establish a service session. This implies that the Candidate Retailer has a federation with the home retailer.

Selected Retailer: the retailer whose service offer is chosen as a result of a dynamic selection process.

Figure 3 depicts the overall design of the personal mobility scenario considered and points out the interfaces where communication is active during subsequent phases of service access, selection and usage. The user establishes an access session with a local retailer (called visited retailer) in the visited network. The user gets authenticated through the visited retailer contacting the user's home retailer. In the case of a 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, together with references to retailers that have a federation agreement with the user's home. If indeed there are local retailers that have federation agreements with the user's home retailer, the user has the choice to purchase personalized service delivered in the local domain. He will also have the capability of dynamically selecting the most beneficial service offering and retailer. Mobile agents are applied in the above scenario to provide a mobile user agent and also to implement an efficient negotiation between the retailers to find the most optimal service offer currently available for the user.

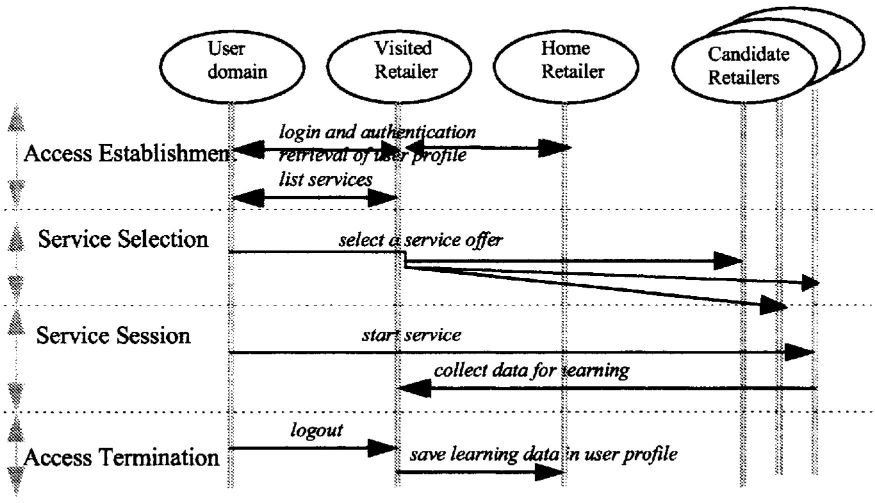


Figure 3. *Access and service session scenarios modified to incorporate mobility and dynamic selection*

The logical phases in service provision are summarized in the following (a more detailed description may be found in [JOR]):

– Access establishment sequence during which the user requests access to the services and his identity is verified.

– Service selection sequence during which a specialized user agent will negotiate with federated retailers who are listed as candidates for offering the service.

– Service session sequence during which a service session is established between the user and the Selected Retailer and the service is offered.

– Access termination sequence during which also the data collected about user's preferences is sent to the home retailer to be saved in the user profile.

The retailer selection introduces a new sequence of actions, which is not present in TINA scenarios so far. The selection is triggered as soon as a user has indicated the service he wishes to use and it ends with a notification, coming from the agent that represents the user in the retailer selection process, informing the user about the selected retailer (for details of the selection, see [PRE 99]). Both mobile agents and stationary objects have been designed and implemented for realizing the scenario presented. In the following chapters the introduced mobile agents are defined and the interactions between them are described.

3. Mobile user agents

The TINA access session maintains the state of a user's attachment to a system and his involvement in services. A user can attach to a system in order to use services and can be involved in many service sessions at the same time; the access session maintains the state of this involvement. To define the access model, a concept of *User Agent* – a stationary computational object representing the user in the retailer domain has been used. A User Agent (UA) together with two other objects: *Initial Agent* (IA) in the retailer domain and *Provider Agent* (PA) in the consumer domain, are the main components of the TINA access session. DOLMEN [DOL 98] extended the concept of user agent for personal mobility by dividing the TINA user agent into *User Agent Home* (UAH) and *User Agent Visited.* The UAH represents the permanent aspects of the TINA UA that resides in the home domain. Being an access point to the user's part of the retailer database, UAH provides information of subscriptions made by the user as well as his preferences. The UAV represents the run-time aspects of the TINA UA that reside in the visited domain.

In this work the access model has been further developed and the UAV is divided into parts: User Agent Access (UAA), Subscribed User Agent (SUA) and User Agent Selected (UAS). The three mobile agents replace UAV in the access, service selection and service session phases correspondingly. The reasoning behind the division was, that the range of functionality of user agent visited was very broad and it was natural to divide it into "small" mobile agents, each responsible for one task only. The agents replacing the TINA user agent will be described successively in the text.

UAH is a stationary, service independent component representing a user in his home domain. Being an access point to the user's part of the retailer's database, UAH (in collaboration with the IA) is responsible for authentication of the user, and it provides information of subscriptions made by the user, as well as of his preferences. It has also capabilities of updating the user profile and creation the UAA.

UAA is a mobile agent created by UAH (in case of positive authentication of the user) in the user's home domain from which it migrates to the domain of the visited retailer. UAA represents the user during the access session in the visited area. It contains part of the user profile relevant to the list of the services the user may make use of in his present domain, user's preferences on the services as well as a list of local retailers that may be considered for service provision (candidate retailers). After the required services are delivered to the user, the UAA returns back to the home domain. It delivers to the UAH information about user's preferences, collected during the access and service sessions. During the phases of service selection and service session, the UAA creates SUA and UAS.

SUA is a mobile agent created by UAA in the visited domain. The SUA's information of the user profile is limited to the information of user's preferences on the service currently selected by the user. Its main role is the role of negotiator with retailers on behalf of the user. In order to perform the task, SUA in the default domain sends its own replicas to all visited retailers that offer the service requested by the user. After the negotiations are accomplished, the master SUA at the visited domain makes decision on the choice of the best offer from the user point of view.

Similarly as UAA, UAS is a mobile agent. It is created by UAA (in the domain contacted by the user) after the service selection has been completed and it contains data relevant only to the requested service. It migrates to the selected retailer domain where it manages the user's preferences on service execution. It is provided with a reference to the PA component in the default domain to enable communication during the service session. After delivering the service to the user, UAS migrates back to the default domain, with the information of user's preferences obtained during the session. After passing the information to UAA, UAS is terminated. Together with UAA and UAH it corresponds to TINA UA.

In addition to the above-mentioned mobile agents, a new static agent is introduced to act as a contact point for the travelling SUA agent. Retailer Agent (RA) is a stationary component residing at each retailer. Its main task is to promote the retailer offers to the visiting SUAs. Its life cycle spans the life cycle of the retailer. The components: IA, PA and User Application (UAP) follow TINA specifications [TIN 97]. In the following sections, the designs of the mobile agents UAA, SUA and UAS are presented.

3.1. *Design of the access agent UAA*

As described earlier, the motivation of the UAA is to support retailer level roaming by moving to the domain where the mobile user is attached. The interface of the agent is presented below in a descriptive form.

```
interface UAAInterface {
    oneway void userContext (in paramList);
    StringList getServiceList();
    ServiceProfile getServiceProfile (in index);
    oneway void updateProfile (in userInfo, in service);
    oneway void setUserProfile (in user_data);
    oneway void setServiceProfile (in service);
    UserData getProfile();
    oneway void selectRetailer (in requestId, in
    serviceName, in profile);
    oneway void selectionMade (in offers,out selection);
    oneway void startService (in paramList, in ip_address);
    oneway void terminateAccessSession();
};
```

Figure 4 presents the behavior of the UAA in relation to components during the access session. After authentication, a UAA is created and it migrates to the visited retailer domain. At the end of the access session, UAA migrates back to the home domain, where it delivers the changes to the user profile based on user choices during the session.

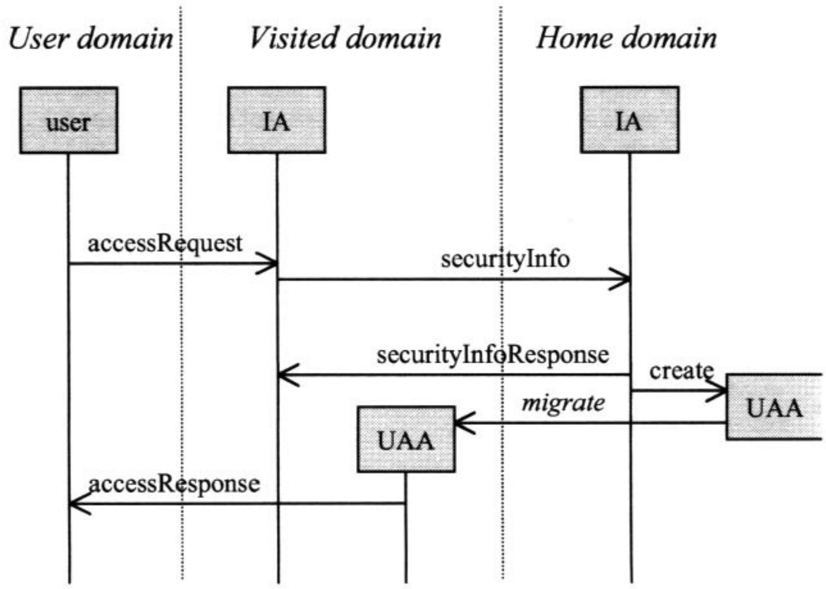


Figure 4. *Interactions between UAA other software modules during the access session*

When the optimal service offering has been selected and the user is willing to start the service, a UAS agent is created and sent to the selected retailer domain (Figure 6). There it plays the user agent role within service session and migrates back after that. The details of interest for future service profiling will be delivered to UAA before UAS is destroyed.

3.2. *Design of the negotiating agent SUA*

In the following, the components active during the retailer selection phase, as well as the actions performed by them is considered. The purpose of the selection phase is to choose the retailer who provides the best offer for the service requested by the user. It is assumed that offers are issued by retailers active in the area that the user currently visits. The phase starts after the user has chosen a preferred service from the list of subscribed services and optionally, expressed requirements on the service. His requirements may relate to service charge, quality of service or preferred language for example.

As mentioned earlier, during the service selection phase the SUA agent represents the user. A SUA inherits from the creating UAA information about the user profile, but only limited to the service selected by the user. Its main task is to negotiate with retailers on behalf of the user. In order to perform this task, the SUA in the visited domain sends its own replicas to visit all Candidate Retailers that offer the service requested by the user. The replicas locally negotiate the best offers, using the information obtained from the user profile. After the negotiations are completed, the SUA at the visited domain collects the results sent by its replicas and makes a decision on the best offer from the user's point of view. The decision is passed to the UAA and the SUA terminates. The interactions during service selection are depicted in Figure 5 while the structure of the SUA agent is presented below in a descriptive form.

```
interface SUAInterface{
    void findBestVisitedRetailer(in RAs, in userProfile, in
    PAref);
    void takeOffer(in serviceProfile, in IA, in utility, in
    netBenefit, in offerId);
    // Used in master SUA
    void suaInit(in master_sua);
    void Goto(in address, in ret);
};
```

The contract partner of SUA in the negotiations is the RA, a stationary agent residing at each retailer. It maintains the current prices of services offered and keeps records of users' choices to consequently plan for and advertise the most "personalized" and desirable offers. As was mentioned before, SUA and RA negotiate on the basis of user requirements and service offerings. The negotiation capabilities are a vital part of a complete SUA and RA design and implementation. The negotiation algorithms designed for the described architecture are presented in [PRE 00].

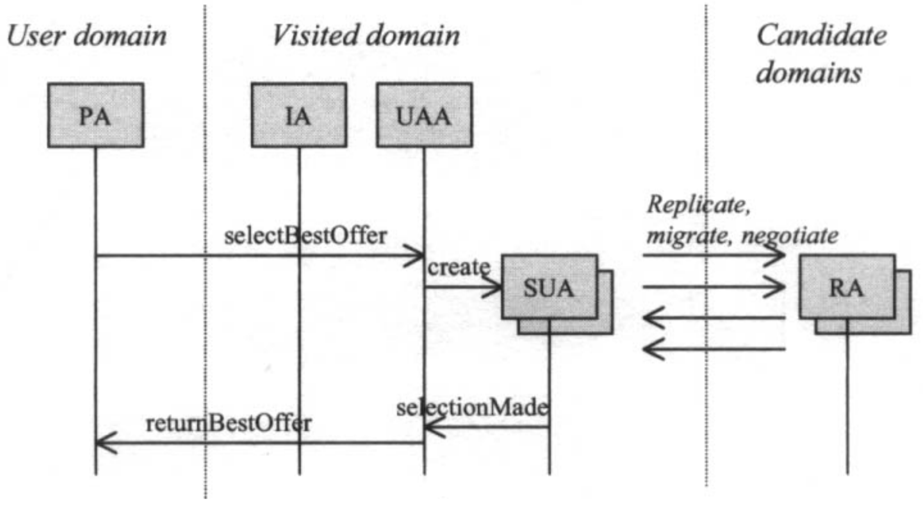


Figure 5. *Operation of the SUA agent by means of interaction and migration*

Based on the value of a utility function for a specific service combination and the charge estimates given by the RAs, SUA may select the retailer that maximizes the net benefit. The advantage of employing a mobile agent (SUA) is that the negotiations with RAs are done locally at the visited retailer nodes. The SUA replicas allow the negotiations to be carried out in parallel thus increasing the efficiency of the retailer selection process.

3.3. *Design of the service user agent UAS*

After the selection phase ends, the UAA creates a UAS, which is a mobile agent migrating to the Selected Retailer where it carries out the User Agent role in service session and manages the user's preferences on service execution. It is provided with a reference to the PA component in the user domain to enable communication during the service session. After delivering the service to the user, the UAS migrates back to the default domain, with information on user's preferences obtained during the session. After passing the information to the UAA, the UAS is terminated. The interactions related to UAS migration are depicted in Figure 6, but the service specific logic within the selected retailer domain is omitted. The structure of the UAS agent is presented below in a descriptive form.

```
interface UASInterface {
    oneway void serviceSessionRequest(in requestId, out
    qosParam);
    UserData getUserData();
```

```
        oneway void updateUserData(in userData);
        ServiceProfile getServiceProfile();
        oneway void updateServiceProfile(service);
        oneway void getUserLogEntries(in userId, in data, in
        sessionId, in start_time, in end_time, out bill);
        oneway void terminateServiceSession (in requestId, in
        sessId);
        boolean updateSection(in serviceId, in sectionId, in
        sectFilter, in sectionInterest);
        boolean refinePreferences(in serviceId);
        string refreshServiceSession (in session_id);
        FilterPreferences getUserContentPreferences (in
        serviceId);
    };
```

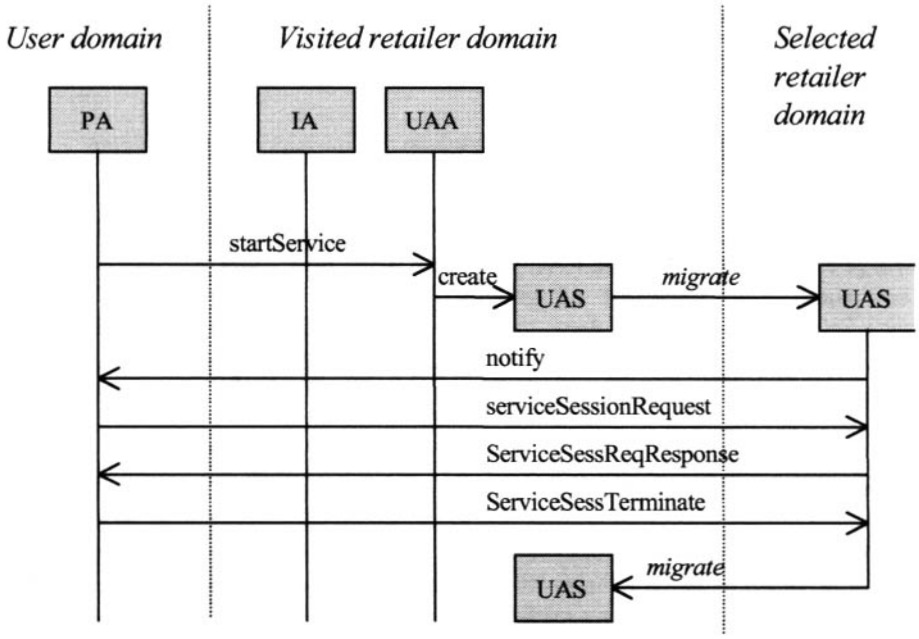


Figure 6. *Service Session sequence with a dynamically selected Retailer*

4. Experiences and conclusions

The previous chapter described how mobile agents have been used in a service architecture by describing a design where user profile roaming and service selection were achieved by means of mobile agents. This chapter concludes with experiences of using mobile agents in respect of performance and software engineering approach.

4.1. *Prototype implementation*

A prototype implementation was made using Java with Voyager [VOY] as the agent platform as described in [SMA]. 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. The interoperability problems encountered with the two platforms are reported and discussed in [JUS 98].

Figure 7 presents the hardware configuration built for demonstration and testing. It consists of three workstations (representing the retailers and a service 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 retailers. Both signalling and service delivery rely on IP for connectivity. The prototype is connected to Mediapoli pilot network [MED] located in Espoo, and it is going to be evaluated in a field trial.

The experiences of the software engineers involved in the pilot development were collected via a questionnaire. The majority considered the agent technology easy to use. The Voyager platform was acknowledged to increase the modularity of the implementation. The attitude was generally very positive towards a new technology, which was to be expected among software developers in research organizations.

4.2. *Performance analysis*

In order to evaluate the impact of mobile agents, a comparison between mobile agent and static object implementation was conducted [VAL 99b] and [VAL 99a]. 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 UAA and the SUA.

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 for the 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).

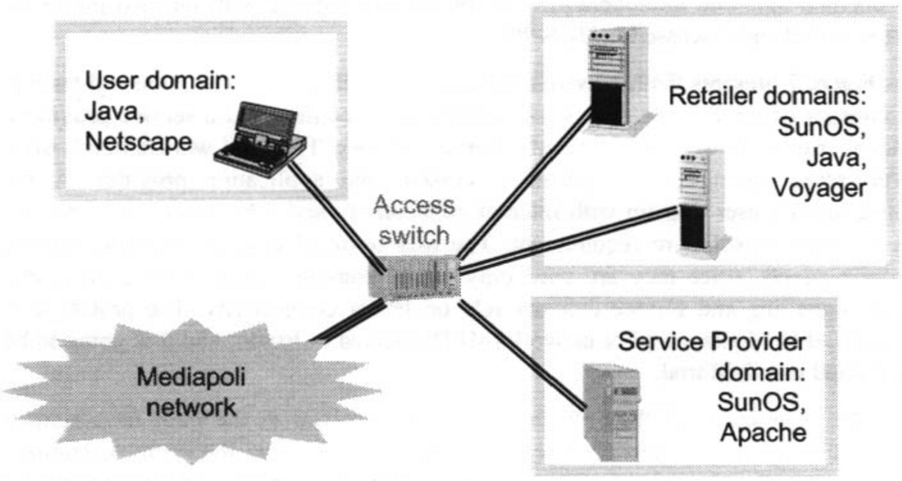


Figure 7. *Network configuration used in prototyping*

The results revealed that the agent migration is a very bandwidth-consuming operation with the current agent platform. The traffic generated by UAA migration was almost forty times bigger that the traffic due to moving the profile as a parameter of a remote procedure call. However, the agent platforms are still heavily evolving and the communication efficiency has already improved which suggests that this result should be interpreted as a worst case performance. The SUA migration gave a hint of a performance gain expected from mobile agents. If the negotiation algorithm would involve five request/reply pairs, the use of mobile agents would already lead to a more efficient implementation than the use of static objects when considering the total bandwidth used. The delay caused by the two approaches was not recorded, but the author believes that whenever the platforms reach a mature state, the timing performance will be a significant advantage of the agent migration. 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 Retailer nodes and only the negotiation result is passed back to the visited Retailer node.

4.3. *Discussion on results*

The goal of this work was to apply mobile agent technology in the development of telecommunication service software and to evaluate the feasibility of the approach. The reported work consisted of design, specification and implementation of a prototype system in which the roaming of user profile into visited domains and a dynamic service selection process were realized with mobile agents.

The design and specification experiences showed that mobile agent paradigm offers an intuitive way to model distributed communication. It also proved the mobile agent concept easy to add in object modelling methods such as class diagrams and message sequence charts. The implementation experiences showed that the agent paradigm was easy to use and that the platforms used offer a good development environment, which in addition integrates well with other platforms (OrbixWeb). However, the implementation of the agent migration should still be optimized in order to compare with remote communication approach. This is expected to happen while the agent platforms are still rapidly developing.

The key innovations within applying agents to TINA service architecture are the implementation of specialized user agents and the extension of access session with retailer selection sequence. Both access and service sessions were enhanced to support personal mobility by means of mobile agents, which appeared to be a natural way to extend and enhance TINA agents. In addition to introducing support for personal mobility, retailer selection on behalf of the user was found to be a natural application for mobile agents and it has been included in the scenario. The dynamic configuration of services and also network related aspects such as bandwidth and quality of services is seen by the author as a promising area to exploit mobile agents.

Mobility has grown into a basic requirement for all network services. Both mobility of users, terminals and services should be supported by core and access networks and the service providers. In order to respond to this requirement, the paradigm of mobile agent should be used in addition to o-o and client server approach in service creation.

4.4. *Future work*

This work has roots in telecommunications service architectures where object modeling and the use of CORBA middleware are key concepts. However, during the work the importance of IP-based networks as a service platform has significantly grown. Although current Internet offers mostly unsecured service with no quality of service guarantees, it seems likely that the future commercial network services will be built on IP. However, the IP based networks will have to be enhanced with middleware infrastructure which allows differentiating of the service levels and charging on usage basis. This suggests focusing the future work towards Internet protocols and especially its mobility solutions.

Another strong trend is the development of the third generation mobile network. Although user mobility and terminal mobility will be essential elements of the UMTS environment, they will only be a starting point. Competition in the UMTS environment will be mostly played out at the service level, with multiple providers offering services over a variety of networks and end systems. In this perspective the concept of personalized service portability across network boundaries called VHE has recently emerged. The ETSI-related Third Generation Partnership Project (3GPP) has defined the VHE as a system concept for personalized service portability across network boundaries and between terminals. It means that UMTS users are consistently presented with the same personalized features, user interface capabilities and services, in whatever network and whatever terminal, wherever the user may be located. The work presented in this text is planned to continue by applying the results in a VHE concept.

A new and interesting network model called Active Network has captured attention of research field only this year. It combines the concepts of programmable switches and mobile code resulting in a flexible architecture where the store-forward principle of packet switching is extended into store-compute-forward [COV 99]. The research on mobility support mechanisms within active network environment is a topic for long-term work.

REFERENCES

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

[ANH 98] V. ANHPHAM, A. KARMOUCH, "Mobile Software Agents: An Overview", *IEEE Communications Magazine*, July 1998.

[CAM] http://www.comnets.rwth-aachen.de/~cameleon/

[CHE 98] D. CHESS, "Security Issues in Mobile Code Systems", p. 1–13 in *Mobile Agents and Security*, (ed.) G. Vigna, Springer 1998.

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

[COC 97] W. COCKAYNE, M. ZYDA, *Mobile Agents*, Manning Publications, Greenwich, USA, 1997.

[COV 99] S. COVACI, "Active Networks", *Proceedings of First International Working Conference Conference*, IWAN'99, June/July 1999.

[DOL 98] ACTS DOLMEN Deliverable ASD4, "Open Services Architecture for Mobile and Fixed Network Environment (OSAM)", April 1998, www page, http://www.fub.it/dolmen/delpages/asd4.htm

[EUR] http://www.eurescom.de/

[FIP] Foundation for Intelligent Physical Agents, http://drogo.cselt.stet.it/fipa/

[GRE 98] M.S. GREENBERG, J.C. BYINGTON, T. HOLDING, D.G. HARPER, "Mobile Agents and Security", *IEEE Communications Magazine*, July 1998.

[HAG 99] L. HAGEN, P. FARJAMI, C. GÖRG, R. LEMLE, "Agent-based Virtual Home Environment: Concept and Performance Aspects", *Proceedings of 2nd International ACTS Workshop*, Singapore, September 1999.

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

[JUS 98] T. JUSSILA, Applying agent technologies in developing telecommunications services – a case study, A Master's thesis, Helsinki University of Technology, 1998.

[KRA 96] S. KRAUSE, T. MAGEDANZ, "Mobile Service Agents enabling Intelligence on Demand in Telecommunications", *Proc. IEEE GLOBCOM'96*, 1996.

[LLO 99] S. LLOYD, R. HADINGHAM, A. PEARMAIN, "Virtual Home Environment' Negotiated by Agents", *Proceedings of 2nd International ACTS Workshop*, Singapore, September 1999.

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

[MON] http://montage.ccrle.nec.de/

[OMG] http://www.omg.org/

[PRE 99] D. PREVEDOUROU *et al.*, "Use of agent technology in service and retailer selection in a personal mobility context", *Computer Networks 31 2079–2098*, 1999.

[PRE 00] D. PREVEDOUROU, K. VALTARI, G. STAMOULIS, A. KIND, M. ANAGNOSTOU, K. RAATIKAINEN, "Agents in TINA to Support Service and Retailer Selection for Mobile Users", accepted to published as a chapter in a book, *On the Way to the Information Society – 5 Years of European ACTS IS&N Research*, edited by Sebastiano Trigila, IOS Press 2000.

[SAN 98] T. SANDER, C. TSCHUDIN, "Protecting Mobile Agents Against Malicious Hosts", p. 44–60 in *Mobile Agents and Security*, (ed.) G.Vigna, Springer, 1998.

[THA 99] D. VAN THANH, S. STEENSEN, J. AUDESTAD, "Realising User Mobility with Mobile Agents", *Proceedings of 2nd International ACTS Workshop*, Singapore, September 1999.

[TIN 97] TINA-C, TINA Business Model and Reference Points, Version 4.0, May 1997.

[TIN 97] TINA-C, TINA Service Architecture, Version 5.0, June 1997.

[TIN 99] H. JORMAKKA, K. VALTARI, D. PREVEDOUROU, A. KIND, K. RAATIKAINEN, "Agent-based TINA Access Session Supporting Retailer Selection in Personal Mobility Context", *Proceedings of TINA'99 Conference*, April 1999.

[VAL 99a] K. VALTARI, H. JORMAKKA, T. JUSSILA, "Experiences with Mobile Agents in Mobility Support and Service Selection", *Proceedings of 2nd International ACTS Workshop*, Singapore, September 1999.

[VAL 99b] K. VALTARI, "Mobile Agents in Implementation of a Virtual Home Environment", Accepted to be published as a contribution to *Smartnet'99 Conference*, Bangkok, November 1999.

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

Chapter 5

WTA-based mobile service management

Mika Andersson
Trio Network Solutions Ltd, Finland

1. Introduction

Wireless Application Protocol (WAP) is dedicated to the goal of enabling sophisticated telephony and information services on hand-held wireless devices. This paper concentrates on how WAP can be utilized to provide new applications and functionality for telecommunications service management. WAP is applied in this paper in the context of Mobile Virtual Private Network (M-VPN). The functionality of M-VPN can be thought of as a Private Branch Exchange, whose extensions are mobile and which is implemented as part of the mobile telecommunications network infrastructure.

WAP itself is an extensive application framework and initially the most attention it received was concerning its capability to provide Internet content for wireless terminals. However, even though that functionality is by no means of less importance, from the viewpoint of a M-VPN vendor its possibilities for enhanced M-VPN service provisioning and control are more significant.

In essence, two aspects of the usage of WAP in an M-VPN environment are considered. Those are how to introduce new functionality into the service and how to make use of the capabilities of WAP Push for application delivery. To reach those goals, the WTA and Push parts of the WAP framework are first outlined, and then some examples of M-VPN WTA applications and their delivery to terminals are considered.

2. Wireless Application Protocol

2.1. *Introduction*

WAP is not only intended to provide Internet content and advanced data services for mobile users but also to integrate the voice call capabilities of mobile phones with new possibilities offered by advancing technology. Wireless Telephony Application (WTA) framework of WAP addresses the requirements of telephony services. WTA offers methods to implement more advanced user interfaces for existing services or, combined with the other elements of Wireless Application Environment (WAE) [1], entirely new applications.

WAP Push specifies a service to push content to mobile devices via the WAP architecture. So far, content delivery has mostly been user initiated, but using Push this can be revised. The network can now initiate a transaction without action from the user. This opens up some interesting opportunities for service enhancement.

The next two sub-chapters introduce the WTA and Push components of WAP. The presentation of the components is not exhaustive. For more information, the reader should refer to the WAP specifications freely available on the World Wide Web (WWW) at the homepage of the WAP Forum (http: //www.wapforum.org).

2.2. *WTA architecture*

WTA architecture adds to the WAP architecture two components. The new components are a WTA User Agent on a client and a WTA server in a network. Figure 1 illustrates one of the possible WTA architectures.

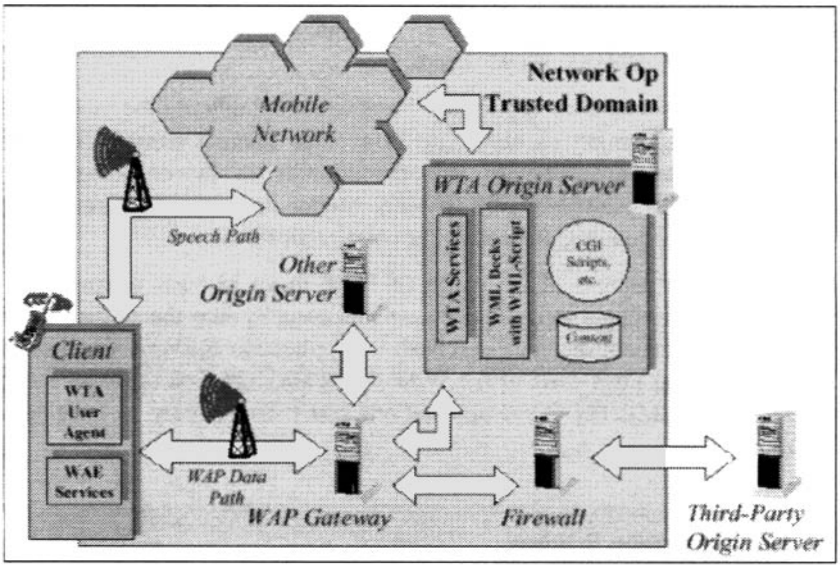


Figure 1. *WAP/WTA Architecture [2]*

As can be seen from Figure 1, a client can access the voice call services of a mobile network. A client accesses WAP and WTA servers through a gateway. A

WTA server may have access to the components of a mobile network, e.g. a switch or a voice mail. [3]

2.3. *WTA User Agent*

A WTA UA on a client is the another fundamental part of the WTA framework, in addition to a WTA server. A WTA UA can be considered similar to a WML UA because it also interprets WML and WMLScript content. [4][5] In addition, a WTA UA can access a repository and execute WTAI functions. WTAI is a function library that contains a set of functions to access the functionality of terminals. [6] In Figure 2, WTA UA and its relation to the other WAP frameworks is illustrated.

The components outside of the shaded area in the figure are a part of the general WAP/WTA architecture. The components within the shaded area are located with a mobile client. Man Machine Interface (MMI) is implementation specific and not discussed any further here.

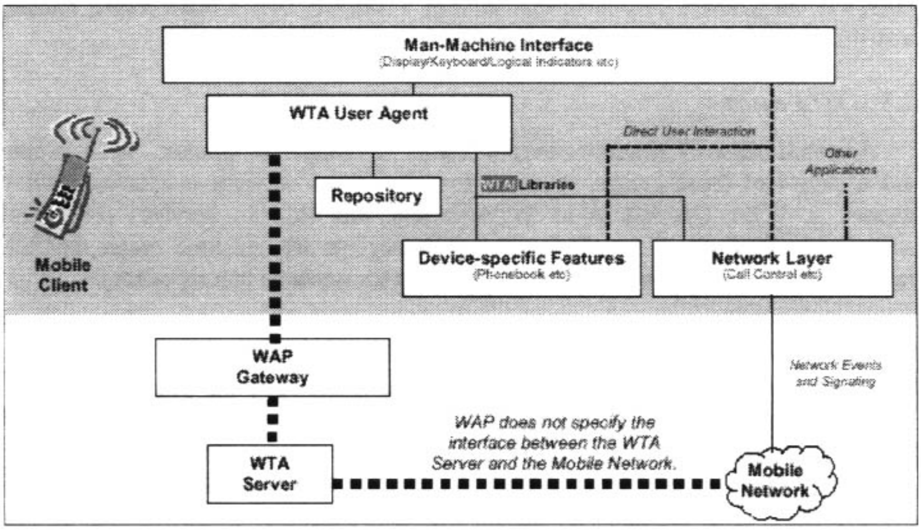


Figure 2. *WTA Framework [3]*

As can be seen from the framework presented in Figure 2, a WTA UA can access a repository, device specific features, and a network layer. A WTA UA uses device specific features and network services through WTAI function calls.

A WTA UA uses URLs to reference content on a WTA server in a similar way, as does a WML UA for WML content. Connections drawn with thin dashed lines are normal, non-WAP specific, client operations. [3]

2.3.1. *Repository*

A repository is an integral part of WTA UA. A repository is a memory on a client where WTA content can be persistently stored. A content in a repository is a resource (e.g. WML deck). In a repository, there can be multiple resources. Each resource must be associated with one or more channels. Channels are used to access resources in a repository, and to form services.

Channels are either pushed or loaded into a repository. All resources indicated by a channel must be loaded, or already exist in a repository, before the channel may be activated.

After all resources have been successfully loaded into a repository, a channel is ready to be activated. A WTA UA requests a 'success' URL, which is defined by the channel, from a WTA server. When the server receives this request, it is informed that the client has received the resources of the channel, and sends a 'success' response to the client (i.e. the content indicated by the 'success' URL). After receiving this response, the client activates the channel. If a channel installation fails, a client requests a resource indicated by a 'failure' URL from a server. Hence, a WTA server knows which channels are active on a client. [3]

2.3.2. *WTA events*

A mobile network generates events related to changes in its state. WTA events are a subset of those events. A client transforms the network events into WTA events. A WTA UA listens to WTA events, and invokes services (channels) associated with them, if any. Because network events are real time events, content involved with the handling of WTA events must be available in a repository.

When a WTA event (e.g. incoming call) occurs, a WTA UA first checks if there exists a 'Temporary Binding' within the current WTA context for that WTA event. If a temporary binding was found, the content associated with the event is executed. If no temporary bindings (i.e. WML 'onevent') existed, and WTA context is not protected from an interruption by global bindings, 'Global Bindings' are checked. Global bindings are the ones defined by channels in the repository. If a global binding was found, a content associated with it is executed. If the previous checks do not resolve any bindings, then a WTA event is handled by MMI. [3]

2.4. *WAP Push*

In WAP Push, a server is the initiator of a transaction, not a client. WAP Push adds two new protocols to the WAP stack, and functional requirements set for a WAP gateway increase. Because of the new functionality, a dedicated gateway (Push Proxy Gateway (PPG)) has been defined for WAP Push.

An origin server (Push Initiator (PI)) and a PPG communicate using Push Access Protocol (PAP). PAP enables a PI to request a result notification for a push-message from a PPG. A PI may ask push cancellation at any time during a push-message delivery. Push Over-the-Air protocol (OTA) is defined for communication between a client and a PPG. The push-message delivery over OTA may be reliable or unreliable. [7][8][9][10]

3. WAP and WTA for service management

3.1. *Service environment*

WAP and WTA for service management and provisioning are described in the context of a mobile switchboard operator service called Mobicentrex. Mobicentrex brings the features of a normal PBX to subscribers' mobile handsets. Unlike traditional PBX services, the Mobicentrex service does not require the subscribing company to invest in switching hardware. All hardware and software are part of the operator's systems.

Mobicentrex offers users of mobile handsets the same call handling features as traditional wireline switching systems, e.g. call transfers. Personal mobility is handled by extension profiles. Each user has several profiles, which they can control via a Web browser interface or WAP-compatible phone. The users can define linked hunting chains to match appropriate situations; for example one for normal office hours, one for meetings and one for free time.

Mobicentrex is implemented as a Service Node (SN) in an IN network. It communicates with SSPs using ISUP. In principle, Mobicentrex SN has a programmable switch, a service control sub-system controlling the switch, and an intelligent peripheral for interactive services. Thus, the functions separated into different entities in IN are combined into one network node.

3.2. *WTA application for call management*

The implementation of the call transfer features can be changed from the SN to WAP/WTA capable mobile terminals. That provides two major benefits.

Firstly, a device independent UI with a familiar look and feel from the WWW can be provided for call management. This way the time required to learn to use the service can be reduced and it is more probable that the features of the service will be more fully utilised.

Secondly, changing some call management operations from a SN onto a terminal simplifies the implementation of the SN and the control signalling between a terminal and the SN required for call transfer is no longer needed.

The drawback of this modification is that the requirements for the mobile network where the service operates increase. The mobile network must provide the call transfer services previously implemented on the SN. The required services are described in [11][12][13][14] for GSM. The WTA application described in this paper operates in a network that uses GSM technology.

If the network does not provide the services needed, similar functionality as presented below can be implemented so that a WTA server has some control over the SN. When an incoming call reaches the SN, the WTA server queries the terminal which should be done for the call. Then, depending on the user's action conveyed to the WTA server, the call is e.g. either connected or terminated on the SN. This model of operation however requires signalling between a terminal and a WTA server while an incoming call is pending or during an active call. Therefore, the requirements set for the data transfer are high.

3.3. *Call management on a mobile terminal*

The WTA application is stored and executed on a terminal. It provides for a user a graphical user interface to control the call transfer features of the service. The operations allowed depend on the states of the other calls on the terminal. The call transfer options are the same as commonly available for an incoming or an active call in a GSM network.

After the WTA application for call handling and its delivery to terminals is described, some ideas of how WTA can be used to add functionality and to some extent, autonomy, to the service is presented.

WTA application

The WTA application is composed of cards dynamically generated by the WTA server using Mobicentrex's subscriber data. When a new user (i.e. an extension) is added to Mobicentrex or user's attributes have changed, the WTA application, with relevant user specific attribute values, is pushed into the user's terminal. Each WTA application is thus unique and created for an individual user.

On the terminal, the service is stored on a repository and bound to WTA events. When a WTA event occurs, its handling is controlled by the WTA application. Each card of a deck is an entity shown to the user at a given time. Calls on the terminal define the states of the system and only a subset of all operations is possible on a given state. WTA events are only responded when the WTA application is on a stable state. A stable state is a state where a card is shown to the user, and neither a content processing nor a state transfer is going on. (The current WAP specification is not fully exact with regards to handling of events. This leaves room for differing interpretations of the process that a WTA user agent must follow regarding WTA events. See 'Conclusions') Transfer from state to state depends on user choices or network events. Two types of user choices can be defined: navigational choices and choices which result network operations to be performed. The navigational choices do not change the state of the system, only a card shown to the user changes. If a network operation, as a response to a choice or an event, fails, an initial state is resumed. The initial state is a state where the system was before the execution of the failed operation began. A successful operation changes the state of system. The following list summarises the essential points of the WTA application:

– Calls on the system define its state;

– Each state has a defined set of possible user choices (operations);

– A user's choice or a network event initiates a transition from one state into a next state;

– Network events are only handled when the application is on a stable state.

The system can be in distinctive states. The states represent possible call states on the terminal. Only one of those states is studied here to give a general idea of how the WTA application reacts to WTA events and performs tasks bound with those events. The operation of the other states follows the same structure and are straightforward modifications to that described here.

Case: Call state 'active'

The state 'active' represents a state where there is an active call on a terminal. The GSM technology allows two operations for an active call when no other calls are present. It can be terminated by either of the parties at any time or put on hold. Figure 3 illustrates the operations, WTA events and resulting state transitions.

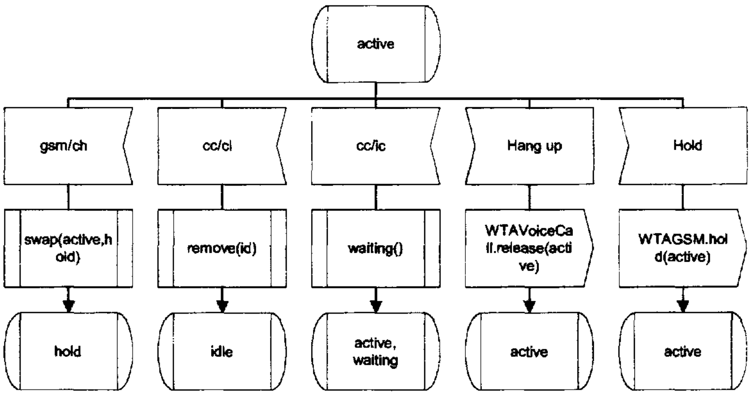


Figure 3. *Call state 'active'*

In the state 'active', the WTA application has temporary bindings with three WTA events. These are 'gsm/ch' generated when a call has been put on hold, 'cc/cl' generated when a connecting/connected call has been disconnected, and 'cc/ic' generated when an incoming call has reached the terminal. The other call-related

WTA events are bound with local bindings as well, but no operation ('noop' [WML]) is performed. Non call-related events are propagated to MMI. It is assumed that the terminal is able to resume an interrupted WTA context after the interrupting event has been handled.

The functions available to a user are to terminate the call ('Hang up') or to put the call on hold ('Hold'). These operations use WTAI functions WTAVoiceCall.release(id) and WTAGSM.hold(id), respectively. The functions use the terminal's functionality to either terminate a call or put it on hold. When the network has successfully disconnected a call, it indicates that to the terminal ('cc/cl'). The same event is also received if the other party of a call disconnects. After receiving the 'cc/cl' event, the state of the WTA application must be updated. The state of the application is stored on state variables representing calls on the system. The call that has been disconnected is removed from the system using WMLScript 'remove(id)'. 'Remove(id)' takes a call to be removed as an argument and performs necessary state updating. Since now there are no calls on the system, a state 'idle' is entered.

The response to the other WTA events ('gsm/ch' and 'cc/ic') is in principle similar to that described above. After receiving an event, the WTA application performs its state managing operations, implemented as WMLScripts, and then enters a new state.

3.4. *WTA application on a terminal*

3.4.1. *WTA application is a channel*

To ensure the timely handling of network events, the WTA application must reside on a terminal. On a terminal, the WTA application is stored into a repository. The WTA application is bound to the 'cc/ic' and the 'cc/co' events with global bindings. Because one channel can have only one associated event (i.e. a global binding), two channels are required. The channels define the WTA event handling for the 'cc/ic' and 'cc/co' events when the WTA application is not executing. When the WTA UA receives either of these events, it loads the associated channel from the repository and begins executing it. Now the WTA application is running and responsible for providing call handling features for the user as described previously.

After a channel is successfully loaded into a repository, a resource indicating the successful loading of the channel is requested from the WTA server. This resource is dynamically named using a unique identifier of the destination terminal. This way, the WTA server can keep track of the terminals that have installed the WTA application successfully.

3.4.2. *Transfer of the WTA application into a client*

When a new extension is added to Mobicentrex or an existing extension has changed, the management part of the service starts generating a new WTA application for the extension. The WTA application is dynamically generated on the WTA server using subscriber's data from Mobicentrex's user DB. Since the WTA application is tailored for each terminal, channels with terminal specific success and failure resource values must also be generated. After the content and the channels have been generated, the WTA server begins a push-submission to deliver the channels to a PPG.

A unique identifier of the destination terminal is used to identify the push-message. The WTA server requests the PPG to inform the outcome of the push-submission. If the push-submission is initially accepted for delivery, but cannot be delivered, the WTA server redoes the push submission. If the push-message is not accepted for delivery at all, resending is not performed, and the service is informed of the error.

Once the PPG has accepted the push-message for delivery, it establishes a push session with the client. The client confirms the successful receipt of the content. This notification is relayed to the WTA server.

The WTA UA processes the pushed content. The content processing results the pushed channels to be installed and the resources for those channels are loaded from the WTA server into the repository. When both channels are activated, the WTA application is ready to be executed.

3.5. ***Introducing autonomy using WTA***

Because the WTA application is dynamically generated for each terminal, it enables user specific features to be implemented as well. Here a few ideas of possible functions are presented.

3.5.1. *Barring of incoming calls*

The Mobicentrex user database may contain entries for each extension defining numbers from which incoming calls are not allowed. Barring can be refined to be time sensitive, i.e. barring for a specific number is effective only at some points of time. This feature is achieved by making the management system of Mobicentrex generate a new WTA application each time the barring is needed to be modified. This new WTA application is then pushed to the terminal overriding the previous one.

Barring is implemented as part of the 'waiting()' function (see Figure 3). After an incoming call event is detected, the WTA application searches the list of barred numbers and if a match is found, the incoming call is terminated. The barring of outgoing calls can be done applying the same scheme.

3.5.2. *Changing the phonebook of a terminal*

The user may have defined some numbers to be automatically updated by the management system of Mobicentrex. Once a followed to number has been changed in the phonebook of a company, a Service Loading [15] is pushed to the terminal indicating an URI where a document containing the phonebook update resides. This document uses WTAI functions to update the phonebook on the terminal.

3.5.3. *Making profiles location dependent*

As explained before, the personal mobility features of Mobicentrex are defined as profiles. The location dependency can be added to profiles by firstly adding the information of the location where the profile should be active into profile definitions. Then, into the WTA application is added a feature to periodically deliver the location information of the terminal (WTAGSM.location() [16]) to the management system of Mobicentrex. When the management system notices that the terminal is on a specific area and no other rule of the profile conflicts, the profile defined for that area at a given time is activated.

4. Conclusions

Terminals supporting WTA are coming to market and at the same time WTA is becoming a working technology in the networks. Since the mobile terminals are quite limited at the moment with their ability to present Internet content, it can be estimated that WTA applications will be a relatively important field of all WAP applications for telecom operators.

However, the event handling of the WTA sets some limits for the usability of the WTA. Events in WTA are asynchronous and must be acted upon in a timely manner. This causes problems if a WTA UA is engaged in some other activity when a WTA event occurs. What should be done for the event? If the current activity is interrupted, it may leave the system in an unstable state. The WTA specification suggests this real-time model. On the other hand, the WMLScript specification states that the WMLScript compilation unit is atomic. This is ambiguous. Nevertheless, disabling an activity interruption is not a solution either, since the network events require a timely handling. Therefore, the event-handling model of WAE must be clarified before the WTA is fully applicable.

Because of the dynamic nature of the WTA application presented, it requires multiple content downloads. In order to achieve high use experience, the bearer used to transfer the application must be fast enough. Thus, the WTA applications like the one presented here will be commercially feasible when bearers from GPRS and up are available for WAP, providing that their pricing is reasonable.

REFERENCES

[1] WAP Forum, Wireless Application Environment Specification, November 1999.

[2] WAP Forum, Wireless Application Environment Overview, June 1999.

[3] WAP Forum, Wireless Telephony Application Specification, November 1999.

[4] WAP Forum, Wireless Markup Language Specification, November 1999.

[5] WAP Forum, WMLScript Language Specification, November 1999.

[6] WAP Forum, WAP Wireless Telephony Application Interface Specification, November 1999.

[7] WAP Forum, WAP Push Architectural Overview, November 1999.

[8] WAP Forum, WAP Push Access Protocol Specification, November 1999.

[9] WAP Forum, WAP Push Proxy Gateway Service Specification, August 1999.

[10] WAP Forum, WAP Push OTA Protocol Specification, November 1999.

[11] GSM 03.83 version 6.0.0 Release 1997, Digital cellular telecommunications system (Phase 2+); Call Waiting (CW) and Call Hold (HOLD) supplementary services; Stage 2, ETSI, 1999.

[12] GSM 03.91 version 5.0.2, Digital cellular telecommunications system (Phase 2+); Explicit Call Transfer (ECT) supplementary service – Stage 2, ETSI, 1996.

[13] GSM 02.84 version 6.0.0 Release 1997, Digital cellular telecommunications system (Phase 2+); MultiParty Supplementary Services – Stage 1, ETSI, 1999.

[14] GSM 03.72 version 7.0.1 Release 1998, Digital cellular telecommunications system (Phase 2+); Call Deflection (CD) Supplementary Service; Stage 2, ETSI, 1999.

[15] WAP Forum, Service Loading Specification, November 1999.

[16] WAP Forum, Wireless Telephony Application Interface Specification – GSM Specific Addendum, November 1999.

This page intentionally left blank

Chapter 6

Service distribution and security

Juhana Räsänen
First Hop Ltd, Finland

1. Introduction

Agents and agent-oriented computing are concepts with interesting characteristics that have opened up new possibilities for distributed computing and software distribution over networks. However, before those possibilities are considered, it may be beneficial to discuss briefly what is actually meant by agents in this context. A weak notion of agency given in the literature [WOO 95] is that agents are defined to be entities having four properties: i) autonomous (operate without human intervention), ii) social (communicate with other agents), iii) reactive (respond to environmental changes) and iv) proactive (act to achieve a set of goals).

However, in this paper these properties are not emphasised, because from the security point of view the most interesting issue is agent mobility [SAH 98] and especially how the mobile agent code is secured.

Therefore a more general definition of distributed agent systems [NIK 99] (p. 3) is adopted:

"An agent is a piece of program code and data, organised as a unit that may be loaded to a node and run."

This definition, whilst keeping the weak notion of agency in mind serves us in this paper, which is organised as follows: some possible scenarios for agent-oriented service distribution are given in Section 2 and the security and especially *trust models* for such scenarios are discussed in Section 3. Implementation of the security architecture outlined using Java and an XML-based variation of IETF SPKI (Simple Public Key Infrastructure) certificate standard is described in Section 4, with concluding remarks in Section 5.

2. Service distribution using agents

Service distribution in this context refers to the physical transfer and deployment of the program code implementing the service rather than the distributed programming techniques (which of course may be used in actual service implementation). Traditionally,the software is distributed on a physical media such as a CD-ROM, but the Internet and new packet-based mobile communication technologies have created a new, more dynamic way of distributing software and software-based services. Java applets embedded to web pages can be seen as a (very) simple form of agents fitting into the definition given in Section 1. Automatic or semiautomatic software upgrades over the network also fit into this category and interestingly can be seen to fulfil even some of the traditional agent characteristics, such as the lack of human intervention. These examples can be understood as simple agent-based service distribution techniques, but as we shall see later in this section, more complex applications can be expected in the future.

Security-wise there is little carried out in these examples: in the case of applets the applet is simply denied access to any critical resources in the target host (by executing the applets in a so-called sandbox environment) and in the case of automatic software downloads the source of software is fully trusted. The security issues are discussed in more detail in Section 3, but before that two examples of the more complicated agent-based service distribution systems are described to illustrate the security problems that arise.

2.1. *Fixed broadband networks*

A Java-based software architecture to distribute services in fixed residential broadband networks has been discussed in [RÄS 99]. In the architecture described the central concept is that the network nodes, especially the access nodes closest to the end-users, are programmable using an agent-like programming model. The network provides a direct API to its resources for the service providers to use, and the service implementations can be deployed into the network nodes for execution. In this model the network nodes are general-purpose computing platforms controlling the network devices (such as routers, firewalls or switches), capable of receiving and hosting specific-purpose *service agents* that the end users access and use. This enables, for example, the service agents to control the individual connections, their routing or QoS parameters on the network side, without having to implement that functionality in the end-user client software. The de-coupling of the user interface and the network control part of the service makes it easier to provide the same service in different kinds of broadband access networks, such as ATM, xDSL or some WLL (Wireless Local Loop) network. Also adaptation to different kinds of terminal devices, such as set-top-boxes, PCs or mobile devices, is easier.

Figure 1 shows a fixed broadband access network node capable of hosting service agents. The service provider has installed a service agent (SA) onto the service agent platform (SAP). The end-user accesses the service via a service client (SC) that communicates with the SA. The SA can use the API provided by the SAP to control the network node and open a connection for the service, such as a constant bit rate virtual circuit for a video stream.

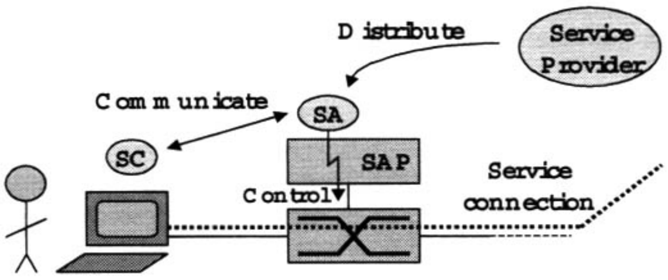


Figure 1. *Service distribution in fixed access network*

The security aspects of this kind of architecture are interesting and more complex than those of the examples given in the previous section. In this system it is clear that the service agents cannot be simply put into a sandbox, because they need to access the network resources of the network node in which they are being executed. On the other hand, the network operator is unlikely to fully trust the software to do only what it is expected to do and not exceed its permissions. It is also unfeasible that the operator would investigate all software prior to installation to the network. Furthermore, different service providers may have made different kind of agreements with the operator about using the network resources, such as bandwidth, QoS or temporary disk storage. As a result, the operator must be able to control on a per-service basis what resources are granted to the service agents and that the agents are not able to interfere with the node or other agents.

2.2. *Mobile computing*

Mobile computing is a new and interesting application domain for agent-based service distribution. The development of packet-based wireless networks, such as Wireless LAN and GPRS, enable the introduction of IP-based services into the mobile world. At the same time the integration of the general-purpose PDA devices and mobile phones forms a completely new computing and service execution environment. Together these two developments create a basis for a number of new applications, usage scenarios and requirements.

Mobile environment has some special characteristics, which emphasise the importance of a sound security model. It can be assumed that downloading and dynamic installation of executable code to mobile devices is going to be even more common than in the fixed networks and PC environment. There are three reasons for this: first of all, the memory capacity of the devices is probably always going to be lower than that of the fixed network terminals, which means that not all applications may be stored in the device at the same time. Second, the devices may not even have a convenient physical medium (such as CD-ROM) for installing new applications.

Third, the usage situations are much more dynamic and context-dependent, which means that the software simply cannot be pre-installed for all possible usage situations.

Examples of such situations are the typical ad hoc networking scenarios: location dependent information services, wireless conference and meeting networking services, multi-player gaming, etc. The location dependent information services could be implemented, for example, using short-range broadcast transmitters, so-called *beacons*. These devices would broadcast over Bluetooth or IrDA to indicate to nearby mobile devices that a local information source is available. The mobile devices would then connect to the information source and download, for example, a guiding application that would provide information for a user in a town which h/she is not familiar with. The application would have to be downloaded semi-permanently to the device, because the connection to the local beacon will be lost when the user moves out of its range. In conference or meeting situations the venue of the meeting could provide some infrastructure services for the participants, such as printing, projectors, bulletin boards or Internet connectivity. It cannot be assumed that all such services could be described with some universal interfaces so that the mobile devices could simply use them by calling remote APIs. Instead, the device must download code, which for example is a driver for a printer available in the conference room or implements some special service just for that conference. The Jini technology [EDW 99] by Sun Microsystems has many similar ideas and usage scenarios. The multi-player gaming applications could be imagined to be used, for example, in airport waiting lounges as an entertainment during the waiting for the plane. Again, if the game application is provided by the airport or some passenger wishing to challenge nearby people into a game, the mobile devices participating in the game must download the application.

In mobile case the situation can also be reversed: the mobile device might want to have some code of its own to be executed on the network side as a mobile agent. This usage might be related to, for example, covering for temporary lack of connectivity or to save the limited bandwidth resource on the air interface by having some computations be performed on the network side instead of the terminal. In this case the network must download code from the mobile device and execute it on behalf of the user, but otherwise the situation is the same: all code that the mobile devices might want to execute on the network side cannot be pre-installed into the nodes.

3. Trust in agent-based service distribution

All of the examples described in Section 2 have one thing in common: executable code (which is called below an "agent" for convenience) is downloaded from a not fully trusted source and executed on a platform that has critical resources. The critical resources must be protected from misuse, but at the same time must be allowed to the agents that have a permission to use them. The permission can be based on the organisation or an individual considered as the owner of the agent having either paid for the usage of the resource or the origin of the agent being trusted to some degree. The problem to be solved is how the amount and nature of trust (or the amount of resources paid for) can be expressed in a practical and secure way and how the run-time permissions granted for the agent can be resolved. (Methods for enforcing permissions in the agent execution environment is outside the scope of this discussion, but it is to be noted that the standard Java 2 security architecture could be used in a Java-based environment.)

3.1. *Identity-based trust management*

There are two basic approaches for representing the trust relationships. The traditional way is based on *identity*: if the identity of the origin of the agent can be verified, the local security policy configuration can be consulted to resolve what rights the agent can be granted. This is the case in signed Java applets, for example: the applet is cryptographically signed using a private key. The applet package contains a *certificate* written for the public key that corresponds to the private key used to sign the applet. The certificate is signed as well and states that the key contained in it belongs to the organisation, whose name is usually given as a domain name. The applet receiver must first verify that the applet signature is valid, which guarantees that the applet code has not been modified after the signature was created. Then the certificate is verified as belonging to the organisation where the applet originated from. The last step is to check in the local security policy - what rights belong to the organisation.

This approach has two fundamental problems: first, maintaining the local security policy is often unfeasible. For example, in the case of mobile users sending agents to the network for execution the operator would have to maintain a policy entry for all users. The second drawback is the trust relationship: How exactly does the verifier of the certificate actually know that the certificate is really trustworthy and not faked by someone who attempts to distribute a malicious applet? In the X.509 public key infrastructure (PKI) [HOU 99] the certificate is signed by a trusted third party (TTP), for example a public certificate authority (CA) such as Verisign or Canada Post Corporation. If the verifier trusts that the CA signs certificates only for real organisations who really are who they claim to be, then certificates signed by that CA can accepted. Optionally, the certification can be multi-level: a CA may sign a certificate for organisation A, which in turn uses the certified key to sign a certificate for organisation B, etc. The certificates form a *chain*, which can be verified step-by-step to trace the chain back to the trusted CA.

However, the problem is that the certificate or certificate chain says *nothing* about the actual trustworthiness of the certificate owner. The meaning of an X.509 certificate is merely that public key X belongs to an organisation whose name is believed to be Y by the CA. It can be said that the first drawback mentioned above is actually a result of the second one: a local security policy must be maintained to represent trust for the identities provided by the certificates. This is a fundamental property of the X.509-like PKIs. Because the root of the trust is the CA that has no relationship whatsoever to the certificate verifier, it cannot make any assumptions about the trustworthiness of the certificate owner to the verifier.

3.2. *Authorisation-based trust management*

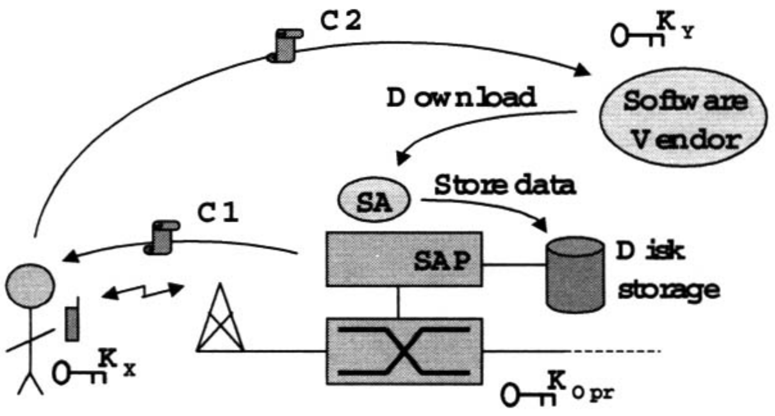
The authorisation-based approach for trust management is described in detail by Nikander [NIK 99]. The approach has two key characteristics: first, the trust root is the actual verifier of the certificates (e.g. the mobile user) and second, the certificates themselves carry the granted access rights in the form of authorisation information. These kinds of certificates are called authorisation certificates to distinguish them from the identity (or name) certificates such as X.509. An experimental RFC concerning authorisation certificates has been published by the IETF SPKI working group [ELL 99].

In practice the usage of authorisation certificates means that the verifier explicitly expresses trust by writing certificates for others. The certificates qualify the trust in a format the verifier is able to interpret later, when the certificates are presented and verified. Certificate chains can be formed in the same way as with X.509, but in addition to binding the public keys together, the authorisation information is also passed in the chain by *delegation*. Delegation means that a certificate holder is able to pass the rights in the certificate further by writing another certificate that contains the same rights (or a subset of them). When the certificate chain is verified, the rights passed in the chain are given as an intersection of the rights in all certificates. Delegation itself can be controlled with a so-called delegation bit, which 'speaks' as a binary truth value, whether the rights can be delegated further or not; in this sense delegation is a special kind of right itself.

If we take mobile agent execution on the network side as a concrete example to illustrate the usage of authorisation certificates (see Figure 2), the situation is quite different from what it would be with the identity certificates. First of all, the trust root is now the operator who owns and manages the network nodes. The protected resources in the nodes, such as connectivity and disk space, are sold for the operator's customers to be used by the agents. The permission to use such resources is represented by writing and signing a certificate C1 which says, for example, that a user having public key X is allowed a 10 MB quota of disk space in the node and permission to open outgoing connections. The user, on the other hand, has acquired software from vendor with public key Y. The software uses network side agents to work on the behalf of the user when h/she is temporarily disconnected from the network; let us say that the software uses an agent that acts as a voice-mail box for the user and temporarily stores the messages on the network side.

The code for the agent has been signed with private key Y to ensure the integrity of the code after the vendor has released it. However, the network operator cannot simply trust all agents signed by the vendor, because the network resources are sold to the users, not the software vendors. The missing link in the chain is the delegation of the rights from the user to the agent: The user must write and sign certificate C2 which says that software signed with Y can use 5 MB of disk quota. Now the agent and its code can be sent to the network node together with the certificates (or the certificates can be stored in some directory service, such as DNS [NIK 98]). The node is able to verify that the agent is signed by key Y, which has been granted 5 MB of disk quota by key X, which is turn has been granted 10 MB of quota by the operator's key. Thus it can be concluded that the agent can be allowed 5 MB of free disk space (but not, for example, open outgoing connections, because that permission was not granted in the certificate chain).

Some details to be noted about the example: Why is the agent code not directly signed by the user? The reason is that if the vendor's key Y is given the rights rather than the agent directly, the network is able to download the agent code directly from the vendor site and the mobile user never has to store the code into the mobile device. This may be of importance if the agent is relatively big and transferring it on the air interface is expensive. Instead, only instructions for loading and starting the agent can be sent by the mobile device to the network node. On the other hand, in this kind of arrangement the user effectively lets any software of the vendor use the permissions in the certificate. This may sound somewhat risky, but if the certificate is further limited to be valid only for a short period of time and only for one active agent at one time (which are restrictions that can be enforced by the network node), the user may prevent misuse by the vendor.



C1: X may use 10 MB and open connections, signed: Opr
C2: Y may use 5 MB, signed: X
SA signed by Y

Figure 2. *Mobile agent execution on the network side*

4. Implementation considerations

It can be argued that Java has taken the role of a de facto implementation language for all kinds of applications involving code and object mobility. This is also the case in the mobile agent environments such as MAGENTA [SAH 98]. From the security point of view Java is also a preferable environment for systems in which run-time access control is performed on code. The Java 2 security model [GON 99] is especially very suitable for this kind of access control because of its fine-grained permission model and other security features built into the language and execution environment. In Java it is possible to express permissions like those given in the examples of the previous section.

However, the trust model default Java 2 security architecture is heavily based on the X.509 identity certificates along the lines of Section 3.1 above. Work has been done to extend the architecture with a authorisation-based trust model and even prototypes capable of carrying Java 2 permissions in SPKI certificates for granting Java code its run-time permissions have been made [PAR 98].

In spite of being an IETF RFC, the SPKI certificates have one downside from the practical point of view: they have not gained wide acceptance and the working group responsible for developing the RFC seems to have quietened down considerably after the RFC had been accepted. The reasons for this can only be guessed at, but at least one factor might be the non-standard and somewhat uncommon S-expression format for the certificate encoding. Although the importance of a widely used encoding format should not be overstated, it does have an effect on the individual developers who have to constantly learn and adopt new technologies and data formats to do their daily jobs.

For this reason, and in an attempt to revive the SPKI certificates as a standard authorisation mechanism, a proposal to use XML as the encoding format for authorisation certificates has been made [PÄÄ 00]. The proposal defines a DTD (Document Type Definition) for authorisation and name certificates that obey the SPKI theory and semantics. We see this as a promising alternative to be taken into use in the Java-based agent systems for several reasons. First of all, an implementation of the XML encoded certificates for Java would be relatively easy to make just by combining existing XML parser packages, Java 2 security architecture interfaces and cryptography packages. Secondly, if the implementation were tightly integrated into the Java 2 security architecture in the same way as in the earlier prototype [PAR 98], the existing agent environments could be easily made to utilise the authorisation-based trust model. This is because at the application level there is not much visible for the programmers: if the environment security model is already based on the Java 2 primitives, the authorisation can be seen only as a new way to assign the permissions to the agents, but the actual run-time access control is performed by the Java 2 platform as before. The third reason we see for the XML-based certificates becoming more popular than the original SPKI is the wide interest there is towards XML at the moment. XML has a good chance of becoming the standard way to represent structural information, and certificates are certainly a good case of structured documents. The benefits of authorisation in general in agent-based software and service distribution is so evident that we expect to see many more applications coming up in the future.

5. Conclusions and further work

We have shown that agent-based software distribution is a technology that has uses in service delivery in both fixed broadband and packet-based mobile networks. The trust model used to assign permissions on agents and their code for performing run-time access control was seen to benefit from the authorisation-based model compared to the simple identity-based model. In practice authorisation-based security architecture would be relatively easy to implement in a Java- and XML-based agent environment, even with minimal modifications to the existing agent applications or platforms. Thus we conclude that all building blocks for creating a secure, authorisation-based agent environment using standard and familiar technologies exist.

As for further work we see the need for more application case studies and working prototypes utilising authorisation techniques to gain more understanding of the application requirements. The prototypes also reveal the usability requirements which are of great importance in environments like mobile networks, where most end-users consist of ordinary citizens rather than security experts, to whom the concepts of permissions, access control and authorisation are more than familiar. On the XML certificate encoding front we plan to continue pushing forward the draft in the IETF and a release of a reference implementation is also planned later this year.

Acknowledgements

There are many people who have participated in the effort of researching and developing an authorisation-based security model for Java platform, but the *primus motor* of that work is Pekka Nikander, PhD and acting professor at the Telecommunications Software and Multimedia Laboratory of Helsinki University of Technology. His research has been the inspiration for this paper among many others. The efforts and co-operation of my colleges Juha Koponen and Juha Pääjärvi at First Hop is highly appreciated, as well as Jonna Särs (former Partanen) from Nixu Ltd. Finally I would like to thank my professor Olli Martikainen for his support for writing this paper and guidance in my MSc work about service distribution in fixed networks, which inspired me to study the problems of service security in the first place.

REFERENCES

[EDW 99] EDWARDS W.K., *Core Jini*, Prentice Hall, New Jersey, USA, 1999.

[ELL 99] ELLISON C., FRANTZ B., LAMPSON B., RIVEST R., THOMAS B., YLÖNEN T., "SPKI Certificate Theory", *RFC 2693*, Internet Engineering Task Force, September 1999.

[GON 99] GONG L., *Inside Java™ 2 Platform Security*, Addison-Wesley, Reading, Massachusetts, USA, June 1999.

[HOU 99] HOUSLEY R., FORD W., POLK W., SOLO D., "Internet X.509 Public Key Infrastructure Certificate and CRL Profile", *RFC 2459*, Internet Engineering Task Force, January 1999.

[NIK 98] NIKANDER P., VILJANEN L., "Storing and Retrieving Internet Certificates", in *Proceedings of the 3rd Nordic Workshop on Secure Computer Systems*, Trondheim, Norway, November 1998.

[NIK 99] NIKANDER P., An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems, Doctoral Dissertation, Helsinki University of Technology, Espoo, Finland, March 1999.

[PAR 98] PARTANEN J., Using SPKI Certificates for Access Control in Java 1.2, Master's Thesis, Helsinki University of Technology, August 1998.

[PÄÄ 00] PÄÄJÄRVI J., *XML Encoding of SPKI Certificates*, work in progress, Internet Draft draft-paajarvi-xml-spki-cert-00.txt, March 2000.

[RÄS 99] RÄSÄNEN J., Broadband Service Architecture, Master's Thesis, Helsinki University of Technology, Espoo, Finland, June 1999.

[SAH 98] SAHAI A., MORIN C., "Mobile Agents for Enabling Mobile User Aware Applications", in *Proceedings of the ACM Conference on Autonomous Agents 98*, Minneapolis, USA, 1998.

[WOO 95] WOOLDRIGE M., JENNINGS N.R., "Intelligent Agents: Theory and Practice", *The Knowledge Engineering Review*, Vol. 10 (2), 1995, p. 115–152.

Chapter 7

Merkato™: a platform for market-based resource allocation

Giovanna Giammarino, Jean-François Huard and Nemo Semret
Invisible Hand Networks, Inc, New York, USA

1. Introduction

Over the past few years, the Internet has become an important mechanism for conducting business. It has helped reduce business costs and enabled the customized delivery of goods and services. To facilitate exchange of goods and services, electronic commerce technologies have been developed, ranging from simple Internet shopping sites to auction sites such as eBay and Yahoo.

At the same time, a small but growing market for telecommunication network bandwidth has developed.

However, bandwidth is unlike traditional commodities, goods and services, in that:

1. It is a shared resource that cannot be stored; as such, any unutilised capacity by one party can be made available to all other parties having access to it or it is lost;

2. Consumption of the resource is real-time, that is, it can occur simultaneously with the buying;

3. Demand fluctuates very rapidly and tends to be elastic.

Recently, the need for a dynamic bandwidth commodity market has been recognized. Bandwidth exchanges have developed between new network companies and Internet service providers. The old system of signing lengthy contracts after weeks or months of negotiation does not move fast enough anymore.

The allocation of information service resources can be viewed as an exchange of commodities. In particular, in the case of bandwidth, Dow Jones is set to launch a bandwidth index that will provide prices for long-haul data routes and will enable companies to use these figures to peg the changing process of contracts when they are buying access to networks.

The emerging telecommunications market calls for new mechanisms for real-time trading of network resources such as bandwidth. The goal is to replace the existing human-based resource trading system, as illustrated in Figure 1, by an agent-based resource allocation system such as depicted in Figure 2.

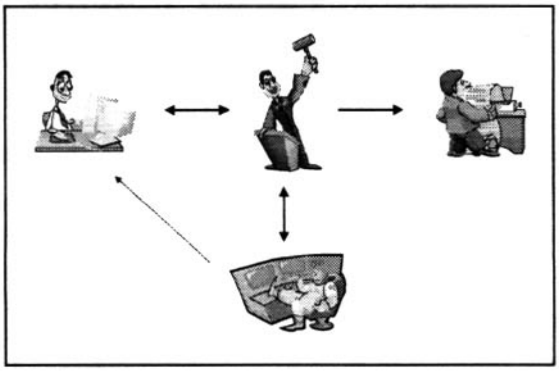


Figure 1. *Current human-based service provisioning: a buyer (left), the sales process (middle), billing (right), and operations (bottom)*

In this paper, we propose a platform for resource allocation built upon fundamental research carried out in the pricing community [NETGAMES]. More generally, the platform addresses the problem of market-based resource allocation with the following characteristics:

1. The sharing of the limited resource is done in a competitive environment;

2. Whoever needs the most resource and has the ability to pay should get the resource;

3. The resource needs of the participants change in time. As such, any static solution would be inefficient;

4. The entities making decisions are independent and dispersed. This calls for a decentralized architecture that works well in large-scale networks.

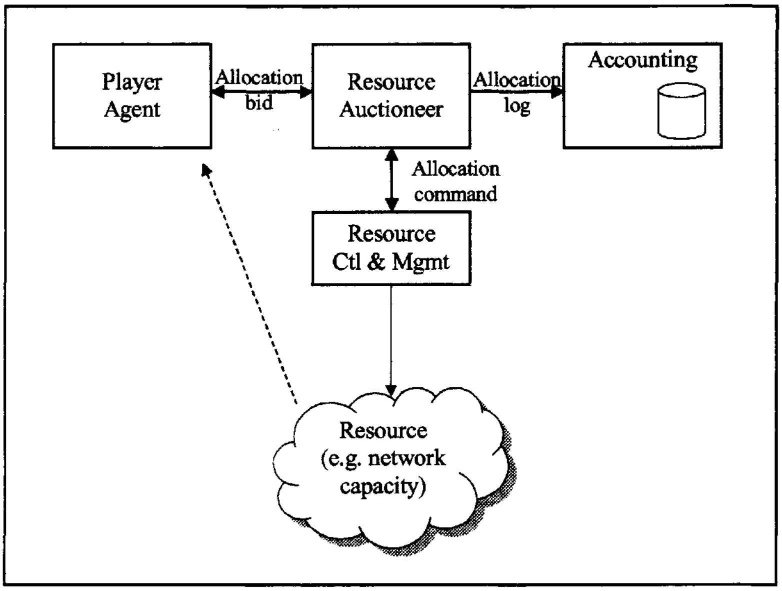


Figure 2. *Agent-based resource allocation system*

1.1. *Market-based resource allocation using agents*

It has long been recognised (and documented since Adam Smith's analysis of markets in the late 18th century [SMI76]), that distributed self-optimising agents often perform resource allocation more efficiently than centralised systems. This is the case not only in production of industrial goods by people, but also in provisioning bandwidth in a communication network. In the latter case, and more generally with "information goods", there is in an additional need to automate the process, with software rather than human agents conducting the small-scale activity of the marketplace.

To see why this is so, imagine that the oil market was such that when a car pulls into a gas station, oil has to be pump from a well, refined into gasoline, transported into pipelines and delivered to the car within seconds. Even if the consumer sees quasi-fixed prices, the negotiations in the wholesale and distribution chain would have to be automated.

Today's telecommunication resource market can be characterized as follows:

1. Everything from the raw material (e.g. bandwidth), to the goods (digital content and communication services), the production, delivery of the goods, and the payment, can or must be handled electronically;

2. Resources are distributed geographically, as well as owned by different entities with different interests;

3. Users are many, very distributed and, even if they wanted to, could not centrally plan their use of resources.

With these characteristics in mind, it becomes imperative to incorporate market mechanisms a priori into the design of management and control systems for information resources, rather than imposing them a posteriori in the operation of commercial services.

Using autonomous software agents in these marketplaces also has a purely economic benefit: that the markets can be designed, and agents behave truly rationally. Market design, which in a human agent world is based on an imperfect analytical model, becomes exact in a world of autonomous agents, and theoretical economic efficiencies can thus be achieved.

There is no denying that autonomous agents, if not carefully designed, can result in excessive complexity and overhead on the system to be managed. Indeed, fundamental trade-offs must be made between computational complexity and signalling load, scalability and resource allocation efficiency. For example, on the Internet, a design that had an economic agent action for every packet transmitted would obviously not be sufficiently scalable – and in some cases not even if it was one action for every TCP connection. On the other hand, large aggregate traffic flows must clearly be "negotiated" in some sense if any quality is to be maintained at reasonable cost across multiple hops in a network.

1.2. *Agents in resource allocation trading system*

The use of agents in resource allocation trading systems has advantages and disadvantages as discussed in the previous section. In this section we present a list of requirements that general agent-based resource allocation architectures must provide:

1. Flexible resource abstraction so that various kinds of resources can be modelled and traded;

2. Support for a wide range of market mechanisms, designed and validated for the particular resource, to be respected;

3. Extensibility of strategic behaviour of the agents to satisfy sophisticated resource needs[1];

4. Inter-operation with accounting, billing, and other supporting services;

5. Real-time control of a variety of underlying resources;

6. Flexible time scales of resource allocation, from milliseconds to months;

7. Minimal obtrusiveness in the underlying use of the resources to be allocated;

8. Flexibility in utilisation of computing resources, through agent mobility.

In the next section, we present the Merkato platform, which was architected to satisfy the above requirements.

2. Merkato platform architecture

The Merkato architecture is based on the concepts of autonomous agents and market-based resource allocation (see Figure 3). Agents are the main components of the architecture and correspond to different entities in the market (sellers, buyers and auctioneers) that wish to trade resources.

1. The designer of a particular system may wish to restrict the range of behaviour of the agents, but should take into consideration that if there is real value in the resources, large numbers of users will naturally try to bypass any restrictions if that will increase their utility. Thus, to the extent possible, custom strategies should be supported and taken into account in the design of the market rules.

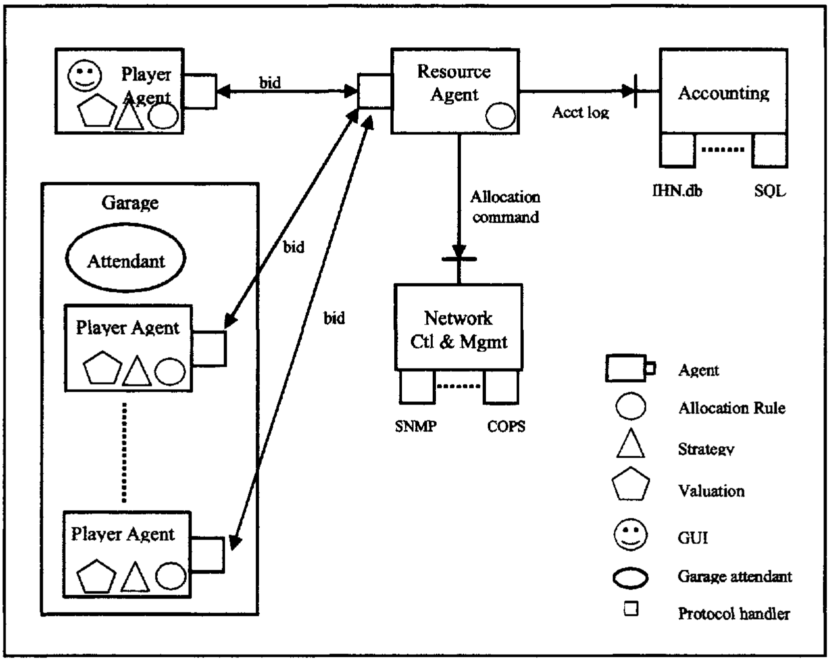


Figure 3. *Market-based resource allocation using the components of the Merkato architecture*

The market-based resource allocation approach introduces the concepts of allocation rule, valuation and strategy. The 'allocation rule' corresponds to the market mechanism. It consists of information that is shared by all participating agents and defines the rules of the market. Examples of market rules are English auctions and continuous bid-ask trading. Several allocation rules have been described in the literature; the interested reader is referred to [RZ94, VAR95, SEM99]. A 'valuation' function corresponds to the value associated to a unit of resource at a given time. The 'strategy' corresponds to the biding mechanism for resources. Strategies and valuation can depend on any external information such as accounting and network congestion. Strategies and valuations must follow the 'allocation rule'; otherwise the player agent will be penalized. In the following sections, each components of the Merkato architecture is briefly described.

2.1. *Resource agent*

The Merkato resource agent runs the marketplace. Receiving bids from player agents it computes how much resource should be allocated to each player based on the market allocation rules. After the resource allocation has been computed, it controls the resource allocation through a network control and management interface and stores accounting information for future usage such as billing.

In a realization of the architecture, there are several resource agents; one for each resource. Examples of resource are link bandwidth, buffer space, and server processing power. Resource agents run within servers that can be distributed all over the network.

2.2. *Player agent*

The Merkato player agent acts on behalf of buyers and sellers of resources. It sends bids to a resource agent in order to trade (sell or acquire) a resource. Following the market allocation rule, the player agent uses its 'valuation' function (that corresponds to how much it is willing to pay for resources at a given time) and its 'strategy' (algorithm) to bid on available resources.

Player agents have a graphical user interface associated with them. The user uses the interface to visualise the market and either bids manually or configures the valuation and strategy that its agent must use on its behalf.

Player agents are component of mobile software that can run on the user's computer or on a server when stored in a garage. The graphical user interface is available only when the agent software is running on the user computer.

2.3. *Garage and attendant*

The garage is a component that serves the purpose of storing agents and enabling their execution remotely from the user's computer. The garage contains an attendant whose tasks consist of helping mobile agents find a parking place in the garage and ensuring proper agent execution when they want to run autonomously. The garage performs the function of a distributed database to store the agents and of a distributed processor to run them. The garage can be located close to the resource agents, but does not have to be.

Garages can also be used to form coalition of agents, using the attendant to bid for aggregated resources on behalf of the coalition.

2.4. *Network control and management*

The network control and management component is used to commit the resource allocated to a player after the resource agent has closed a trade. The component abstracts a network control and management system by providing a common interface for all systems supported. The component loads multiple protocol handlers in order to allocate resource on each of the network elements it can control.

2.5. *Accounting*

The accounting component is used to store information for billing purpose. Like the network control and management, it abstracts the accounting system by providing a common interface for all accounting systems it supports. The component loads a multiple accounting system and database handler for each of the systems it supports.

2.6. *Directory service*

The directory service is not shown in Figure 3. It allows player agents to find the location (in the form of a URL) of resource agents, garages and player agents based on resource requirements, description and availability of resources.

3. Realizing the architecture

Several design requirements have shaped the way the Merkato platform architecture was realized. Among others was the need for platform independence that suggested an implementation in Java. As shown in Figure 4, the system design is composed of four layers. The OS layer (shown as the top layer) corresponds to the platform operating system which is machine dependent and on which the Java platform (shown as the second layer) runs. Our design uses standard Java concepts that allow the Merkato platform to run on any Web browsers and servers, and, as Java application.

The Merkato architecture (shown as the third layer) runs on the Java platform. As illustrated in the figure, the player agent can be executed on the client side (user computer) from a Java application or an applet running in a Web browser. On the server side, the player agent is executed in a servlet that corresponds to the garage component in the architecture.

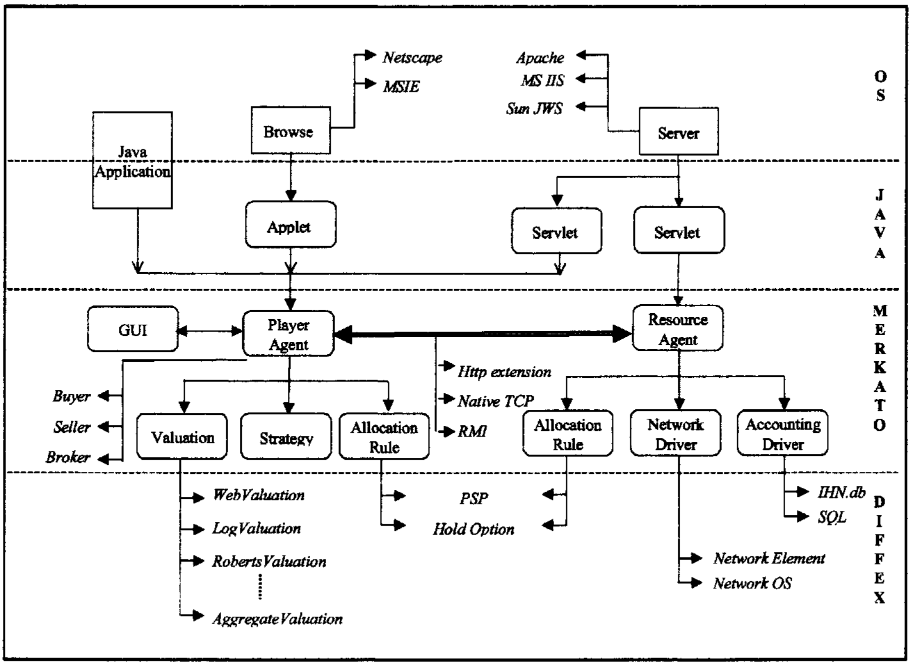


Figure 4. *Merkato system design*

The resource agent is always executed on a Web server. Each resource agent runs in its servlet. From that point of view, one can see that the architecture is scalable since resource agents can be distributed to run on any Web servers supporting the concept of servlet.

Communications between the player and resource agents are done via a resource agent proxy using any of http extensions, native TCP or Java RMI. All communications are secured using secured socket layer (SSL).

Finally, Figure 4 illustrates the fact that each agent (player and resource) has an allocation rule object. The player agents also have valuation, strategy and GUI objects, and the resource agents have networking and accounting drivers for interfacing with external supporting systems.

3.1. *Scalability*

The Merkato architecture was realized as a scalable and lightweight system. The resource allocation computations are distributed over multiple servers using modular servlets. The mobile agent software can run on the users' machine as well as on remote servers. As such, the computational load can be distributed over the network and this is easily realized due to the Merkato's Java platform independence.

3.2. *Modularity and extensibility*

The Merkato platform was realized as an extensible system using plug and play modular components and open APIs. The modularity of the platform allows new components to be easily added to the architecture. Furthermore, the use of modular components and open APIs results in an extensible system where third parties are allowed to customize as well as extend the system functionalities. For instance, as shown in the bottom layer of Figure 4, extending the allocation rule component customizes the market mechanism. Similarly, the player agent can customize its valuation and strategy.

Valuations can be represented in many ways: from low-level description that describes the valuation function (e.g., from pairs of quantity and price and a curve interpolating them) to high-level descriptions such as the number of hits on a web site, the average file size downloaded, the expected delay and the value of each hit. From a high-level description, an appropriate valuation object is constructed. Valuation can be further extended to be time-dependent and dependent on the network state.

There exist engineering tradeoffs for the representation of the valuation. A low-level description requires possibly a large amount of data to be transferred for each bid, resulting in an inefficient use of the communication resources and delaying the resource allocation. On the other hand, a high-level description reduces the optimality of the resource allocation algorithm.

Finally, the resource agent proxy, network driver and accounting drivers can also be extended to work with different types of protocols, tracking and billing systems.

3.3. *Agent mobility*

Mobility of the agents is provided through XML. Using XML, Merkato defines a syntax and semantics for agent description. Each agent can be transferred from one location to another by generating its XML description. Templates are used to generate agents, but once it is parameterised, the agent can be re-instantiated either in a garage or on the users computer in an applet or Java application.

3.4. *Market mechanism*

This section discusses two of the market mechanisms implemented in Merkato. These mechanisms are based on fundamental research in pricing.

3.4.1. *Progressive second price auction*

The Progressive Second Price auction (PSP) [LS00] is a new decentralized mechanism for allocation of variable-size shares of a resource among multiple users. Under elastic demand, the PSP auction is incentive compatible and stable. PSP is efficient in that the equilibrium allocation maximizes total user value. In a dynamic real-time setting, PSP is guaranteed to converge rapidly. PSP applies to independent resource sellers on each element of a network with arbitrary topology, with players having arbitrary routing and provisioning constraints. This networked PSP auction model can apply to virtual networks, virtual paths, and edge capacity allocation networks.

3.4.2. *Hold option*

The Hold Option (HO) [SL99] is a new concept for advance price and quantity guaranteed reservations of network resources in a real-time market environment. Periodic auctions (progressive second price or other) among arrivals grouped in batches gives rise to the spot market of capacity charges. A reservation guaranteeing access for an arbitrary duration with a capacity price below the bid can be made at any time before or during service, thus eliminating the risk-inherent to the spot market-of losing resources to higher bidders before service completion. The reservation is defined as a hold option, which is analogous to derivative financial instruments (e.g. options, futures) integrated over time. Based on a heavy-traffic diffusion model for the corresponding two-stage queuing system, reservation fees can be computed as the fair market price of a hold option.

3.5. *Security and authentication*

Security issues are addressed on two levels: the traditional communications level and from the perspective of the market stability.

Traditional security mechanism for communications are assured through the use of e-commerce standardized secure socket layer communications and encrypted password protection.

The market stability is provided through the implementation of specific allocation rules that assures the stability of the system. To protect the market from exploitation, each user is required to pay a bid fee for every bid sent. The implementation of a bid fee prevents users from trying to artificially destabilise the resource price and preventing a resource allocation to converge. To prevent the "man in the middle attack" (a third party intercepting a bid from another agent and keeping sending the same bid over and over,) timestamps are used to discriminate every bid. Also the scalability of the architecture protects the system from failing in case of a "denial of service attack".

3.6. *Use of standards*

Merkato is designed to make use of industry standards wherever possible to increase its robustness and compatibility with other platforms:

1. Merkato is written entirely in Java, allowing both the client and server-side to run on a wide range of operating systems including Linux, Solaris and most other flavours of Unix, and Microsoft Windows;
2. Database connectivity is via the JDBC standard, allowing Merkato to run alongside most existing database systems for accounting, billing and settlements;
3. Inter-agent communications can use purely extensions to HTTP, thus making it possible to run without interfering with firewalls, proxies and other middleware;
4. Secure communications and secure transactions are assured through the use of e-commerce standardized SSL communications and encrypted password protection. Users can securely participate in Merkato using ordinary browsers from any host on the Internet;
5. To control and manage network elements, the network drivers use standards such as SNMP and COPS;
6. The description of agent is specified in XML.

4. Illustration

4.1. *Inter-ISP bandwidth trading*

The global Internet is a collection of thousands of inter-connected networks. Each Internet Service Provider (ISP) must connect its network to other ISPs to ensure that the traffic from its customers can reach its destination in any other network. Traditionally, inter-connections were according to a free peering model, on the assumption that exchanged traffic was more or less equal in both directions between any two ISPs. In today's commercial Internet, the reality is that a small ISP needs a backbone ISP much more than vice-versa. For the backbone ISP, the added value of being connected to one smaller network is marginal, but for the latter, being connected to the backbone is essential. Thus, in recent years, large ISPs have been shying away from free peering with smaller providers.

Instead, a crude and static dichotomy has emerged, wherein the large ISPs charge the small ones for "transit service", in effect treating them as any corporate customer, while continuing to peer with each other at no charge. This is economically inefficient, as it fails to cover the range of the dynamic relationships between networks as traffic patterns evolve.

Furthermore, it has encouraged the development of private (bilateral) peering. This increases the overall instability and cost of the infrastructure. Ideally, all ISPs would be able to exchange traffic at peering points on a multilateral basis. This allows data to travel to the destination using the best possible path - often directly to the destination ISP, whereas bilateral connections imply that traffic may have to travel through several more networks to reach the final destination. To get the same degree of interconnectivity, it is more expensive and time-consuming for ISPs to connect one-to-one which requires O(N2) contracts and connections for N ISPs, than to all connect to a peering point, which only requires O(N) contracts.

The fundamental issue is thus the lack of a sustainable economic model to go along with the architectural advantages of peering.

"The Internet interconnection environment remains one where there are no soundly based models of financial settlement in widespread use today. [...] These issues [...] will shape the future of the entire global ISP industry."[2]

The supply of bandwidth has been growing very rapidly, by many measures, doubling every year. However the demand is also growing - traffic volume on Internet backbones doubles every three to four months. As capacity is being added in a race to keep up with demand, and the demand patterns shift rapidly, congestion occurs in different parts of the Internet. Thus, high-quality service offerings require more or less explicit allocation of resources, which in turn requires differentiated pricing.

Bandwidth is an essential commodity, and a dynamic commodity market for bandwidth would provide multiple benefits, among them:

1. On a short time-scale, price incentives for efficient allocation of resources, allowing more valuable traffic (e.g. multimedia streams and e-commerce transactions) to not be slowed down by less quality-sensitive traffic (e-mail, bulk content replication for caching);

2. On a medium time-scale, information for better provisioning by network managers, based on demand for different flavours of quality of service;

3. On a long time-scale, better capacity planning and investment decisions, guided by derivative (e.g. futures) markets.

4.2. *Diffex*

Diffex is a market-based bandwidth allocation system for bandwidth exchange between peering ISP. It is built upon the Merkato platform and enables real-time bandwidth markets between peering ISP.

2. "Interconnection, Peering and Settlements", http://www.telstra.net/gih/peerdocs/peer.html. Also in "ISP survival guide", G. Huston, John Wiley & Sons, 1999.

Diffex allows ISPs to buy and sell bandwidth from each other in real-time. Diffex auctions, in real-time, the outgoing bandwidth on each ISP's line out of the exchange, ensuring that, at all times, the bandwidth goes to the buyer with the highest value for it. Diffex also allows for buying and selling in advance (i.e. making reservations with guaranteed capacity and capped prices), through a derivative market of options, in effect enabling the trading of risk and hedging, for original buyers and sellers as well as purely "financial" players.

4.2.1. *QOS support*

Diffex interfaces with routers to control the allocation of bandwidth in real-time. It supports different network equipment vendors' quality of service capabilities and related technologies, both proprietary and standards-based. For example, a Diffex bandwidth allocation may be mapped to:

1. A set of class-based weighted-fair queueing parameters;
2. A service level agreement based on Diff-Serv standard traffic classes; or
3. Capacity within MPLS tunnels.

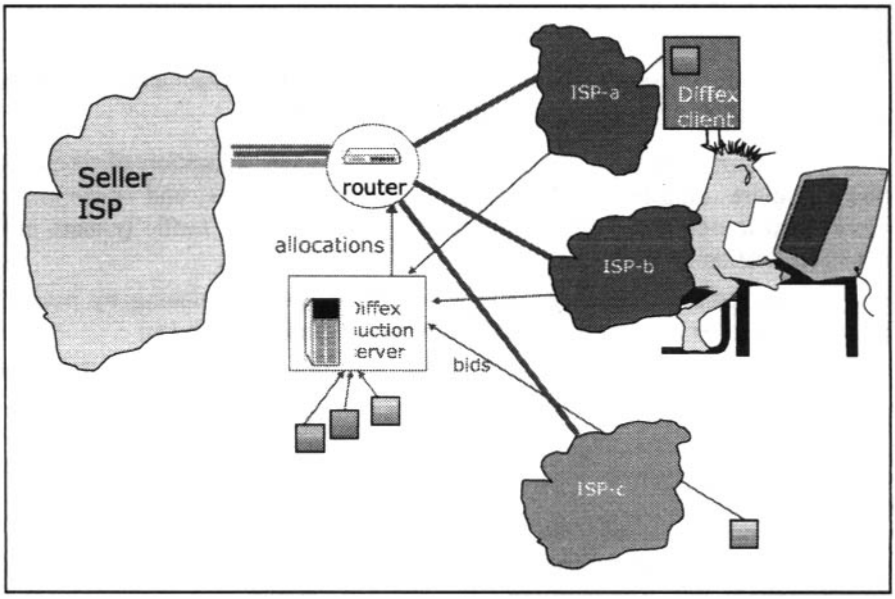


Figure 5. *Diffex software: bandwidth exchange between peering ISP*

4.2.2. *Market mechanisms*

A dynamic bandwidth market cannot be realized by simply applying traditional mechanisms from other commodity markets, without risking instability and inefficient results. Diffex markets are run with the Progressive Second Price auction as the spot market, and the Hold Option as the derivative market mechanism. These mechanisms are proven to be optimal (in terms of speed, efficiency and stability) for the real-time buying and selling of internet bandwidth.

5. Related work

The need for appropriate design of economic agents and mechanisms can be found in [RZ94, VAR95]. Early work on computational agents can be found in [AXE84, MD88, WHGKS92, SEM95]. Finally, [SH00] presents some recent work on auction house based on combinatorial allocation.

[RZ94] applies the general approach and the mathematical tools of game theory in a formal analysis of rules governing the high-level behaviour of interacting computer systems. It describes a theory of high-level protocol design that can be used to constrain manipulation and harness the potential of automated negotiation and coordination strategies to attain more effective interaction among machines, even those that have been programmed to pursue different goals. Merkato design philosophy, both in the agents and the mechanisms, was influenced at the early stages by [RZ94], where the following are identified as the basic principles of automated agent mechanism design: stability, simplicity, efficiency, and fairness.

The importance of incentives and the methodology of mechanism design were crystallized for information resources in part by [VAR95].

In [AXE84], computer tournaments and mathematical analysis demonstrate that cooperation based upon reciprocity can emerge and prove stable provided the shadow of the future is long enough.

[MD88] examines markets as a model for computation and proposes a framework-agoric systems-for applying the power of market mechanisms to the software domain. Markets are considered as ecosystems and evolving software spread across a distributed computer system serving different owners pursuing different goals. [MD88] explores the consequences of this model and outlines initial market strategies.

In [WHGKS92], Swarm, a multi-agent software platform for the simulation of complex adaptive systems is presented. In this system the basic unit of simulation is the swarm, a collection of agents executing a schedule of actions. Swarm provides reusable components for building models and analysing, displaying, and controlling experiments on those models.

In [SEM95], TREX, a resource exchange system is presented. TREX is a client-server game, in Java, implementing auctions for network resource sharing. TREX is the precursor to the Merkato platform.

In [SH00], NOMAD, a mobile agent system for an Internet-based auction house is presented. Agents travel to auction houses and participate into auctions on behalf of users. The NOMAD architecture contains an agent dock, which is similar to the Merkato garage. As opposed to Merkato, in which all player agents can be programmed to react to external events, have their own valuation, strategy and bidding rule objects, in NOMAD, several types of mobile agents are provided.

6. Conclusion

We have presented the architecture and design of Merkato. In briefly describing the Diffex system, which is based on the Merkato platform, we illustrated the extensibility of Merkato. By designing market rules and interfaces for different resources, one can extend such a platform to a wide range of applications in real-time markets for information resources, with unprecedented efficiency.

REFERENCES

[AXE84] AXELROD R., *The Evolution of Cooperation*, Basic Books, 1984.

[LS00] LAZAR A.A., SEMRET N., "Design and Analysis of the Progressive Second Price Auction for Network Bandwidth Sharing", *Telecommunications. Systems, Special Issue on Network Economics,* to appear.

[MD88] MILLER M. S., DREXLER K. E., "Markets and Computation: Agoric Open Systems," *The Ecology of Computation*, Elsevier Science Publishers/North-Holland, 1988.

[NETGAMES] resource available at http://comet.ctr.columbia.edu/networking_games/

[RZ94] ROSENSCHEIN J. F., ZLOTKIN G., *Rules of Encounter*, MIT Press, 1994.

[SEM95] SEMRET N., "TREX – The Resource Exchange", 1995, available at http://comet.columbia.edu/~nemo/Trex

[SEM99] SEMRET N., Market Mechanisms for Network Resource Sharing, Ph.D. Thesis, Dept. of Electrical Engineering, Columbia University, May 1999.

[SH00] SANDHOLM T., HUAI Q., "Nomad: Mobile Agent System for an Internet-Based Auction House", *IEEE Internet Computing*, Mar/Apr. 2000, pp. 80–86.

[SL99] SEMRET N., LAZAR A.A., "Spot and Derivative Mechanisms in Admission Control", *16th Intl. Teletraffic Congress*, Edinburgh, June 1999.

[SLCL00] SEMRET N., LIAO R.F., CAMPBELL A.T., LAZAR A.A., "Peering and Provisioning of Differentiated Internet Services", *IEEE INFOCOM 2000*, Tel Aviv, April 2000.

[SMI76] SMITH A., *The Wealth of Nations*, W. Strahan & T. Cadell, 1776.

[VAR95] VARIAN H. R., "Economic Mechanism Design for Computerized Agents", *USENIX Workshop on Electronic Commerce*, July 1995.

[WHGKS92] WALDSPURGER C. A., HOGG T., HUBERMAN B. A., KEPHART J. O., STORNETTA W. S., "Spawn: A Distributed Computational Economy", *IEEE Trans. on Soft. Eng.*, vol. 18, no. 2, Feb. 1992, pp. 102–117.

This page intentionally left blank

Index

This page intentionally left blank