M-commerce services - what do we gain with mobile agents?

Jens Hartmann, Markus Hück
Ericsson Eurolab Deutschland GmbH
Ericsson Allee 1, 52134 Herzogenrath
(Jens.Hartmann, Markus.Hueck)@ericsson.com
Phone: +49.2407.575-121, Fax: -400

8th February 2001

1 Introduction

Agent Technology, in particular Mobile Agent Technology (MAT), is often seen as an enabler for future communication concepts and might pave the ground for flexible environments, where known and trusted agents serving real user demands [HMB98]. A mobile agent is a program roaming the network under its own control on behalf of its owner [CHK95]. It can migrate from host to host and interact with other agents and resources on each host. The agent should be able to execute on any machine within a network, regardless of the processor type or operating system. Thus, an independend agent system has been installed on code system such as a Java Virtual Machine (JVM).

A mobile agent operates independently of the application from which the agent was invoked. The agent operates asynchronously, meaning that the client application does not have to wait for the results. This is valuable for users who could not always be reached by the network, e.g., nomadic users. Remote programming allows a user to delegate a task to an agent. The communication device must be connected to the network only long enough to send the mobile agent on its way and, later, to take it home. It does not need to be connected while the agent carries out its assignment. In this paper an m-commerce agent platform, which helps to carry out search and ordering activities via WAP-enabled mobile phones, is introduced and evaluated. Therefore, three different search methods using stationary as well as mobile agents have been implemented and tested.

2 M-commerce description

Mobile e-commerce (m-commerce) is expected to be the main driver for the upcoming new mobile communication systems. Common e-commerce scenarios assume that the shopper is using a PC and a medium-speed Internet connection (typical modem speed of 50 kbit/s for downlink direction). Using a WWW browser, the shopper can visit multiple electronic merchants, distributed on various hosts. A list of merchants is obtained by a portal site, e.g., yahoo. The shopper compares the prices at different merchants and finally decides to buy the product at the merchant, which offers the best price or makes a trustworthy impression to the shopper. The user provides the merchant with the delivery address, and shipping information. The payment information is sent encrypted to the merchant. The shopping process ends with the merchant's confirmation of the transaction.

This complete shopping scenario takes in average 5 to 15 minutes and a a total of about 150 to 450 kByte of data has to be transfered [Hüc00]. Of course, this is not applicable to 2nd generation mobile phones, due to the limited existing data rates and the comparable high communication costs. Besides, the restricted capabilities of a WAP-enabled mobile terminal regarding display, input, and data transfer rate hinder the browsing of HTML pages. The mobile phone's browser supports in most cases only WML, but not HTML. Even though, today many major WWW sites offer WML content along with normal HTML, WML is not the answer to all restrictions of the terminal.

3 MCAP - An m-commerce agent platform

The Mobile Commerce Agent Platform (MCAP), which is introduced in this paper, provides a convenient way for accessing all kinds of e-commerce services via WAP-enabled mobile communication devices. Many research projects in the mobile commerce area deal only with mobile payment. Other phases of the m-commerce as for example the browsing and ordering raise diverse problems, too, mostly relating to the limited display, input, and transmission speed capabilities of mobile terminals. The MCAP approach shows a solution that handles all phases of the m-commerce, see Fig. 1. The responsibility for the payment transaction is transferred to the MCAP as already proposed in a similar scenario in [HEG+99].

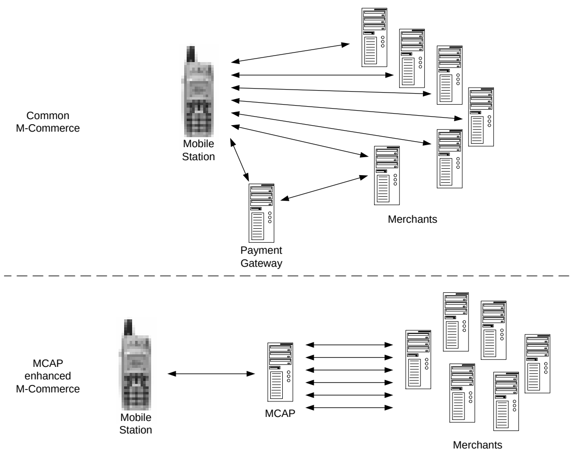

Figure 1: MCAP Placement

The MCAP consists of an agent platform (Voyager), a web server (Apache), and a database (mSQL), and generates a profile for each registered user, containing address and payment information. To fully utilize the services it is necessary for the user to supply the MCAP with a set of valid payment information, e.g., SET and HBCI. Since this is highly sensitive data, a full trusted relationship between user and MCAP is required. A malicious MCAP is capable of charging the user's bank account without any user interference. Merchants have to register at the MCAP. Moreover, they should provide an infrastructure that is compatible to the one used by the MCAP. A network connection should be available at any time, because all queries have to be handled locally at the side of the merchant.

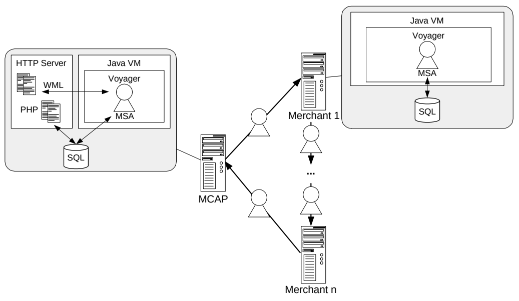

Figure 2: Scenario I: MSA

The agent platform Voyager of ObjectSpace [HGF98] handles the communication of the diverse agents that are responsible for the information retrieval, information storage, user notification, and payment transaction. The agent mobility lies in the scope of functions of Voyager, too. The Apache web server is the front-end for all user communication. It delivers WML content that is generated on demand by PHP scripts.

Services are not limited to the purchase of ordinary e-commerce goods, but can be extended to include hotel rooms, a restaurant guide, or car rentals, for example. Most of these services gain additional advantages, if the user's location is taken into account. The location can be obtained by the mobile network operator or directly via the WTAGSM.Location() function of WMLScript, which returns the Location Area Code (LAC) and the Cell ID [WAP98].

4 Search Agents

This section deals with the information retrieval, which is especially important, because an intelligent search service reduces the number of user interactions to a minimum. Three search scenarios are regarded, two of them are based on mobile agents and the last one on a stationary agent.

4.1 Scenario I: Agent-based search

This implementation demands an agent platform, the Voyager ORB in this case, and an SQL database

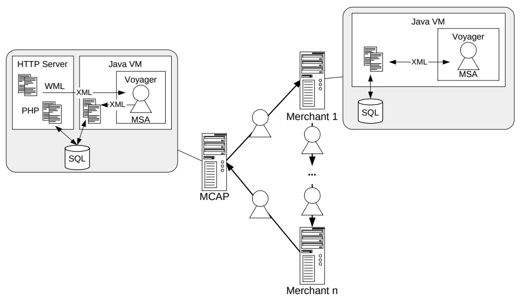

Figure 3: Scenario II: MSA with XML Interface

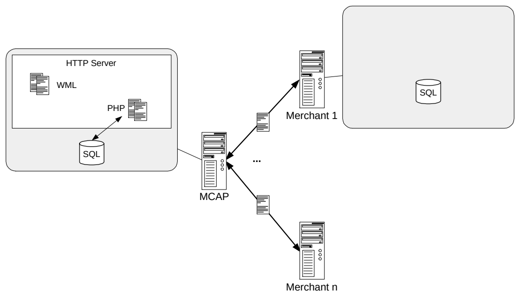

Figure 4: Scenario III: SSA

to be installed on each involved host. A PHP based script within the MCAP receives the HTTP user request. It retrieves the user data and passes the information to the Voyager agent server. An overview of the hosts and installations is given in Fig. 2. The MCAP launches a Mobile Search Agent (MSA) after checking whether the requested product group is in the scope of the available merchants.

The MSA retrieves a list of merchants that come into question for the product. With all information collected, the MSA moves to the first merchant in his list. If the Voyager based merchant has available resources, it accepts and executes the agent after downloading the state and class information.

The agent queries the local database for the product. If the result is qualified, meaning that it matches all necessary conditions in regard to price, availability, and description, the agent keeps the result. Former result sets may have to be overruled to reduce the data size. The search method, the proceeded time, and the timeout selection determine, if the agent moves to the next merchant or returns to his home to deliver the outcomes. Upon arrival at home the agent stores the results in the database. Another agent is triggered, which is responsible for informing the user of the outcome of the product search.

4.2 Scenario II: Agent-based search with XML interface

This scenario is quite similar to the previous one. Again, an agent platform is necessary on all hosts involved. The main difference is that the agents are equipped with an XML interfaces, which enables agent to agent communication. Thus, the MSA does not communicate with databases directly, see Fig. 3. A local agent generates a list of merchants and stores it together with the user request data in an XML document. This document is passed to an MSA. The MSA uses an XML parser to validate the document and to retrieve the information.

The MSA moves to the first merchant in his list. There, it establishs a connection to the Local Merchant Agent (LMA). The MSA looks up the corresponding Java class in Voyager's directory service and generates an XML document, containing the product description and passes it to the LMA. The implementation of the LMA lies in the responsibility of the merchant. The LMA encapsulates the result in an XML document and returns it to the MSA. The MSA decides how to proceed: if its task is fulfilled, it returns to its origin, otherwise the search continues at the next merchant.

At the MCAP the MSA constructs an XML document containing all retrieved results and passes it to the agent XHOME. The XHOME agent is responsible for storing the results in the database and notifying the user.

4.3 Scenario III: SQL-based search

The Stationary Search Agent (SSA) is solely implemented in PHP, because the limited functionality of the stationary agent does not require agent to agent communication capabilities nor agent mobility. The basic setup is shown in Fig. 4. The sequence of tasks is similar to the MSA of scenario I. The major difference is the missing mobility of the SSA. A list of merchants is generated and shuffled randomly to ensure load balancing and equalization of all merchants. The SSA queries the merchants successively for the product: A remote SQL connection is established and an SQL query is sent to the merchant. All results have to be transmitted via the network connection from the merchant to the MCAP, the filtering is performed after reception at the MCAP. When all queries are processed, the filtered results are stored in the database and a user notification is executed.

The merchants have to provide an SQL database with remote accessibility and a predefined set of tables and fields. An agent platform is not required at the merchant.

5 Performance evaluation

The product search time depends on the scenario and the search method: FIRST ends after one result was retrieved, THREE ends after grabbing three results. Search method BEST and SMART always query all available merchants. Best delivers the three best results to the user. SMART differs from the other search methods. It is used to search for products that are not addressed by a concrete name but by a set of characteristics. The agents have to evaluate each data set by comparing a set of parameters, regarding even flexible thresholds. The product search time increases with the number of merchants for all scenarios, but the maximum user request rate the system can cope with is only changed for scenario III, the SSA. For the SSA the maximum user request rate decreases significantly by increasing the number of merchants. The search method SMART sticks out: the maximum user request rate that the SSA can handle is decreased even more, because the SSA has to filter the received result sets at the MCAP, consuming more processing power, whereas the MSAs filter the result sets right at the merchants. Figure 5 shows this trend in detail. It shows the maximum user requests for a fixed MCAP workload of 80% dependent on the number of requested merchants. The values for the MSAs are constant at 3.2 and 4.8 user requests per second, whereas the SSA shows its dependence to the number of requested merchants. The break-even is reached at 18 and 27 merchants for the MSA and the MSA with XML, respectively. From the MCAP's point of view, the MSA is very attractive, if a high number of merchants has to be requested or the information retrieval demands a more complex task than a simple query can accomplish. Since the number of requested merchants is not necessarily known at the beginning of a product search, the usage of MSAs helps to estimate the workload of the MCAP.

Figure 6 represents the product search time for a well-assumed mixture of all four search methods. The SSA's maximum user request rate is located in between the values of the MSA with and without XML interface.

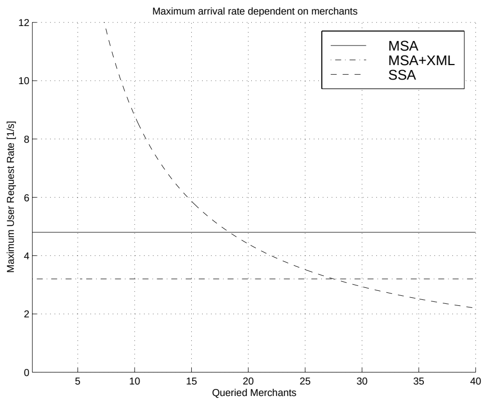

Figure 5: Ratio of merchants to user request rate

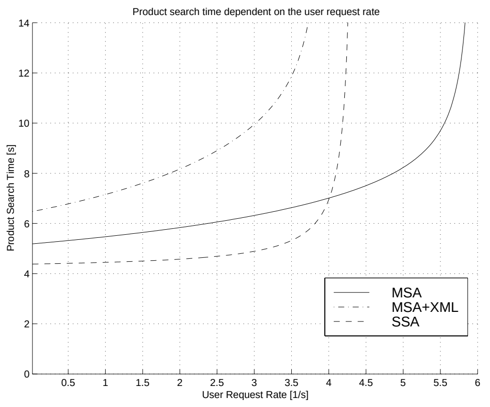

Figure 6: Mixture of all four search methods

More details of the performance study will be included in final version of this research paper.

6 Results and Conclusion

The center of this implemented m-commerce scenario is the MCAP. It is responsible for receiving the user requests, handling the user profiles, launching the search agents, and stores all information in a database. The implementation regards stationary (SSA) and mobile agents (MSA) for performing the product search. MSAs migrate from host to host to query local resources. Two kind of MSAs have been implemented: one uses SQL to query the local database, the other one uses an XML interface to communicate to local agents. The XML generation and parsing requires more performance overhead, but enables a versatile and expandable communication interface towards MCAP and merchants. The alternative to MSAs is the Stationary Search Agent (SSA). The SSA executes all queries from the MCAP by using SQL. The SSA is in most cases faster than the MSAs: a search over 40 merchants takes about 2 seconds for the SSA, the MSA needs five times longer, and the MSA with XML interface seven times longer. This is caused by the higher processing time of mobile agents in comparison with their stationary counterparts. On the other hand MSAs can move to the place of the data source to filter the results locally, the SSA has to transfer all data sets that come into question from the merchant to the MCAP and filter them locally. This is exhibited in the behaviour of the search times for the SMART search. The processing time difference of MSA and SSA falls behind the difference in transfer times. A SMART search performed by an SSA takes about 40 seconds, whereas the MSAs just need about 30 seconds, transferring only a fraction of the data volume.

Due to the fact that mobile agents use distributed processing capacities, the scaling behaviour of the MCAP differs for MSA and SSA. While MSAs generate a fixed load on the MCAP, due to the fact that the MCAP is not involved in the actual product search, SSAs start all actions at the MCAP, resulting in a load that increases with the number of merchants and the complexity of the filter process. Hence, depending on the used hardware at a certain number of merchants a break-even between SSA and MSAs will be reached.

Mobile agents are flexible, if equipped with the capability to react to changes to the normal transaction strategy. Static agents lack the ability for direct communication with a counterpart. Each message is transferred over a network link. This feature is especially critical, if the network link is not steady as for a radio link. Mobile agents can operate without the interference of the home server. A dynamic communication interface such as XML can define new transactions as they become appropriate, e.g., enabling the agent for a direct purchase.

All presented search scenarios are capable candidates for an m-commerce service. The application of the scenario has to be decided from case to case. Each implementation has its advantages and disadvantages, in regards to duration, quality, and costs. The SSA is faster as the MSAs for the ordinary search, but falls behind the MSAs, if a more complex search or task is requested.

References

[CHK95] David Chess, Colin Harrison, and Aaron Kershenbaum. Mobile agents: Are they a good idea? IBM Research Report, 1995.

[HEG+99] Jens Hartmann, Rune Evensen, Carmelita Görg, Peyman Farjami, and Hai Long. Agent-based banking transaction and information retrieval - what about performance issues? In *European Wireless'99*, Munich, Deutschland, Oktober 1999.

[HGF98] Jens Hartmann, Carmelita Görg, and Peyman Farjami. Agent technology for the umts vhe concept. In *ACM International Workshop on Wireless Mobile Multimedia*. ACM SIGMOBILE, Dallas, USA, October 1998.

[HMB98] Lars Hagen, Thomas Magedanz, and Markus Breugst. Impacts of mobile agent technology on mobile communications system evolution. *IEEE Personal Communication Magazin*, August 1998.

[Hüc00] Markus Hück. Performance evaluation of an agent-based e-commerce application for wap-enabled mobile communication devices. Master's thesis, RWTH Aachen, Dep. of Communication Networks, August 2000.

[WAP98] Wireless application protocol architecture specification 1.1. WAP Forum, 1998.