Mobile Agent Communication in Wireless Networks

Heikki Helin*, Heimo Laamanen†, Kimmo Raatikainen*

*University of Helsinki, Department of Computer Science
e-mail: {heikki.helin|kimmo.raatikainen}@cs.helsinki.fi
WWW: http://www.cs.helsinki.fi/{heikki.helin|kimmo.raatikainen}

†Sonera Corporation, Mobile Communications
e-mail: heimo.laamanen@sonera.fi
WWW: http://www.cs.helsinki.fi/heimo.laamanen

Abstract — **Software agents are gaining more and more attention in both research and commercial worlds and are sometimes proposed to be used in mobile wireless environments. While agents may solve many problems typical to these environments, agents require special support from underlying architecture. Mobile agents, perhaps the most known class of agents, needs special treatment in these environments. Although moving agent's code from mobile device to stationary host, and running agent there solves the problem of unexpected disconnections, the migration process is sometimes too time consuming compared to traditional message passing. In this paper we discuss about agent communication issues concentrating on the problems the wireless environment causes, and give some guidelines how these problems may be solved.**

I. Introduction

Portable computers and handheld devices with wireless technology offer facilities to nomadic users to employ network services while on the move. Developing applications that make effective use of the Internet resources from mobile wireless platform is challenging for several reasons. First, the characteristics of wireless links are considerably different from those of wireline links.Wireless links typically have low and variable throughput, high latency, highly variable transmission delays, and in some cases long connection establishment time. Furthermore, wireless link may not be available due to deteriorate radio conditions or uncovered area at all. Therefore, a wireless link creates problems for services, which are designed to operate with fast and reliable network connections. Second, the variety of mobile workstations that nomadic users use to access the Internet services increases at growing rate. Equipped with limited processing power, limited amount of memory, and limited display quality these devices are anemic, thus causing severe challenges for application designers — same applications should be able to operate in these devices as well as powerful desktop machines. Third problem is terminal mobility. An essential part of mobile wireless computing is that the user has access to fixed network everywhere and all the time. The mobile device may get a different IP-address every time it connects to fixed network.

Currently these problems are mainly solved using client-mediator-server paradigm [1], [2]. While these solutions are acceptable in many situations, we believe that a new paradigm based on software agent technology will provide us with much better results. However, implementing software agent technology in a wireless environment is also a challenging task. Today's agents and their platforms are designed to operate in fixed network environments, therefore they do not yet address the wireless environment described above. There are several examples. First, running today's agent platforms on an anemic mobile computer is a difficult task. Second, mobile agents migrating in fixed network while user is in disconnected state should have means to detect once user re-connects to fixed network, so that they can found their way back to the home device. Third, minimizing the amount of bits to be transferred over a wireless link in the case when deciding whether a mobile agent should go over the wireless link or data should be transferred over the wireless link is a challenging problem.

In the remainder of this paper we examine agent communication issues in wireless environment, especially a) communication optimization in different communication layers, and b) performance model to be used to calculate an estimate whether an agent should migrate or use message passing from the host it is located.

The rest of this paper is organized as follows. In Chapter II we give an overview of agent communication in mobile wireless environment and in Chapter III we describe the layered model for agent communication, and give possible optimization schemes for these layers. In Chapter IV we provide a performance model for agent migration. Finally, Chapter V concludes the paper and gives directions for future work.

II. Communication in Wireless Environment

Fig. 1 shows different scenarios of agent communication in wireless environment. Case 1 is the most typical one. The mobile device is powerful enough to run full agent system and a number of agents. Agents at the mobile device communicate over the wireless link with other agents at fixed network, and possibly transfer themselves between mobile device and fixed network. In Chapter IV we will give a performance model that can be used as a basis to select appropriate action; whether to migrate or whether to use message passing over the wireless link. In the case 2, the mobile device is sort of a PDA machine that is unable to run full agent system because of hardware and software limitations. However, having a stand-alone agent control tool in the mobile device, a user may, for example, start agents in fixed network and get results back to the mobile device even if the mobile device is itself unable to run agents. In both of these cases, the mobile device communicates with a terminal communication agent (TCA) located at fixed network. TCA acts as a proxy for mobile device while the mobile device is in disconnected state. Furthermore, the TCA takes care of loc-

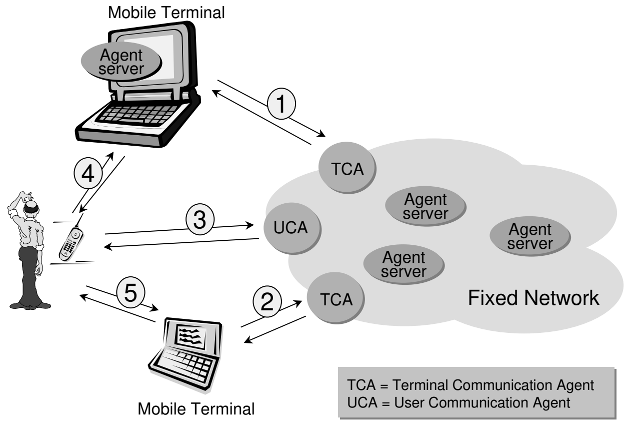


Fig. 1. Agent communication cases in mobile wireless environment

ating mobile devices whose IP-address changes frequently. Once mobile device re-connects to fixed network, possibly with a new IP-address, it informs to its TCA about its new IP-address. TCA takes care of forwarding this information to appropriate agents. In the case 3, the user may have for example a digital phone which is used to control agents at fixed network, and agents can use this phone while reporting results back to user. For example, an agent may send a GSM short message (SMS) to user once it has finish its task, or user may use same technology to cancel agent operation without establishing data connection to fixed network. For this kind of communication, there is a User Communication Agent (UCA) located at fixed network. Cases 4 and 5 are communication between the user and agents in mobile device. This is similar to any communication between the user and the agents — the only difference is in the user interface. In powerful devices an agent may have an advanced graphical user interface, but in some cases the mobile device hardware gives limitations to agents user interface. The communication between the user and the agents is not discussed in this paper.

III. Communication Layers

In order to perform useful tasks, agents should be able to communicate with each other. To achieve interoperability between communicating agents, agents should be able to understand each other — they should use the same transport protocol, they should understand the message transport protocol, they have to use the same agent communication language, they should use same ontologies, and finally, if they are using some interaction protocol, they should use is as it is specified.

Inter agent communication can be divided into four layers (see Fig. 2). Interaction protocol layer contains high level protocols for interaction between communicating parties. Various negotiation protocols and communication patterns belong to this layer. Communication language layer defines the content of messages exchanged between communicating parties. Message transport layer contains various protocols that are used to transfer messages between communicating peers. Furthermore, message transport layer may take care of location transparency, that is, mapping between logical agent name and physical agent location. Transport protocol layer contains actual network transport mechanisms, such as TCP/IP, MDCP [3],WAP [4], and SMS. In order to implement agent communication in a wireless environment as efficiently as possible, optimizations are needed in each of these layers. In this section we show how to optimize and enhance these layers to enable efficient communication in wireless environments.

In interaction protocol layer, an agent may optimize its communication pattern by reducing the number of messages to send by coupling several messages to one. Moreover, an external observer agent may learn about communication patterns used by other agents, and optimize these patterns without interfering the communicating agents. An illustrative example of interaction protocol layer optimization is optimization of HTTP protocol. In normal operation, WWW client first retrieves an HTML page from server. While receiving the page, the client parses the page and explicitly retrieves all embedded objects, such as inline images and applets, from the server. Clearly this is not the optimal strategy. Instead, after receiving page request, the server or a mediator can automatically send the requested page and all embedded objects to client. This technique significantly reduces needed round-trips, and has been successfully employed in various wireless aware Web browsing implementations (see for example [5]).

Optimization in the communication language layer in-

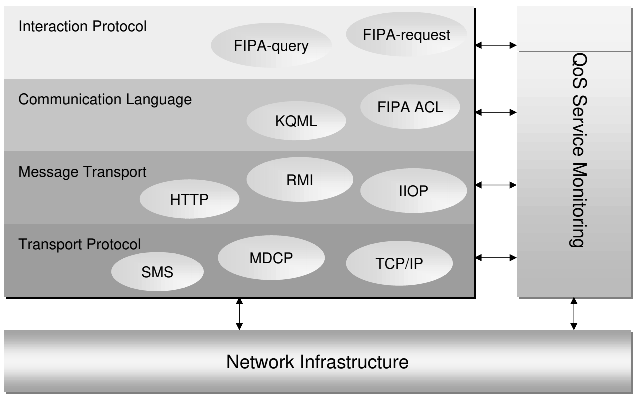


Fig. 2. Agent communication layers

cludes compression and conversion between media types. Agents need a common communication language in order to communicate with each other. Foundation for Intelligent Physical Agents (FIPA) [6] has defined one such language. FIPA ACL [7] defines precisely the syntax, semantics, and pragmatics that is the basis of communication between agents. However, FIPA ACL messages are currently encoded as human readable ASCII strings. Using tokenized binary encoding of FIPA ACL messages instead of ASCII string utilizes wireless link more efficiently. In addition to efficient coding of ACL message, actual content of the message should also be compressed or converted to more appropriate form, if possible. For example, should the content be a GIF-image, it may be converted to JPEG-image with decreased number colors and image quality.

Message transport layer optimization includes optimization of protocols to be used. Protocols such as IIOP [8] or Java RMI [9] can be used as message transport protocols, but these need optimizations in wireless environments. For example, due to its high protocol overhead, both in data traffic and in round-trips, Java RMI is poorly suited for communication in wireless networks.

Location transparency in wireless environments needs special support so that wireless link disconnections can be hidden from communicating agents, and mobile agents can be found efficiently even if the mobile device is reconfigured after link disconnection. Message transport layer should allow agents to use same identifiers even if the mobile device's IP-address changes. This can be achieved by giving to agents global unique identifiers (GUIDs) which are not constructed using IP-addresses, but are somehow mapped to actual transport address only when needed.

The transport layer should provide an efficient and reliable transport service. It should be transparent to agents, and thus agents are unable to optimize anything there by themselves. Transport layer could, for example, automatically select appropriate protocol to use. When the mobile device is disconnected, transport layer may use SMS to deliver messages, if such service is available.

Another important feature of a wireless-aware communication system is failure transparency. The transport layer should hide transient connection failures from agents. For example, if an agent located in a mobile device needs to communicate with an entity located in fixed network, or vice versa, sending a message should not fail if the wireless link is not open, unless defined otherwise. Although the transport layer should hide the effects of mobility and wireless environment as well as possible, it should enable some kind of control to agents. That is, if an agent knows that it is operating in a wireless environment, it should be able to control its usage of wireless link.

IV. Performance Model

In the previous chapter we provided optimizations for agents that are using message passing over the wireless link. Using mobile agent technology, an agent may transfer itself over the wireless link instead of using traditional message passing from a mobile device. In some cases this is more efficient way to communicate, but not always. Furthermore, if an agent migrates to fixed network, it may continue its work there while the mobile device is in a disconnected state. While this is an obvious advantage, there is a number of details that should be taken into account. An agent should carefully analyze whether it should first migrate itself to near to communication peer — for example from the mobile device to the fixed network — or whether it should use message passing from location where it resides. It is evident, that neither always using message passing nor always migrating over the wireless link are optimal strategies [10], [11]. Making wrong decision might have a catastrophic effect on system performance. In an extreme case, an agent might not be able to migrate over the wireless link before disconnection due to its large size, while the same agent using message passing is able to finish its task.

If an agent knows in advance most of the communication attributes and other network characteristics, the agent can estimate whether it should migrate or not. In the general case, however, it is impossible to choose optimal strategy. The agent may not know the network characteristics well enough, or volume of data transfer is unknown. Furthermore, predicting future network conditions in wireless environment is a challenging task.

Next we present one performance model that can be used to estimate whether an agent should migrate or use message passing from the host it is located.

A. Message Passing

In the message passing case, an agent remains stationary and uses message passing. The network load (B_{msg}) of message passing from location L_1 to location L_2 consists of the size of the request B_{req} and the size of the reply B_{rep}:

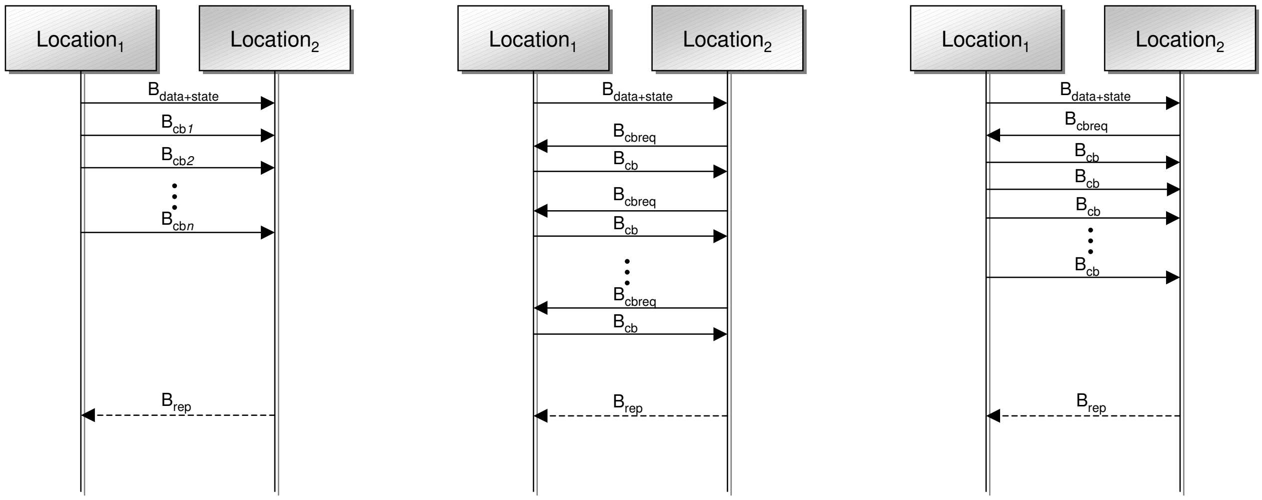
$$B_{msg}(L_1, L_2, B_{req}, B_{rep}) = \begin{cases} 0 & \text{if } L_1 = L_2, \\ B_{req} + B_{rep} & \text{otherwise} \end{cases} \tag{1}$$

Corresponding transfer time (T_{msg}) consist of the time for sending a request and receiving a reply:

$$T_{msg}(L_1, L_2, B_{req}, B_{rep}) = 2\delta(L_1, L_2) + \left(\frac{1}{\tau(L_1, L_2)}\right) B_{msg}(L_1, L_2, B_{req}, B_{rep}), \tag{2}$$

where $\delta(L_1, L_2)$ is network delay and $\tau(L_1, L_2)$ is throughput between locations L_1 and L_2. Because the agent does not migrate to fixed network, it cannot perform any semantic pre-processing on the reply message. The system might provide for example a generic compression service, that can be used to reduce transfer volume. Furthermore, optimization techniques given in Chapter III should be employed. In the performance model given here, it does not matter whether the system compresses the data or not. In case of slow wireless links, the compression phase typically does not affect the transfer time, because even a naive compression algorithm can produce output faster than throughput of wireless link.

Equations given above are not actually enough when estimating transfer time in real environment. For example, it takes time for an agent to locate its communication peer. Typically at least one DNS query is needed, before the agent can even start sending messages. These details are not considered in this performance model — mainly because they are implementation issues. One might think, for example, that there is always one communication channel open to



(a) (b) (c)

Fig. 3. Agent code block transfer scenarios. (a) In case 1 all code blocks are transferred automatically from source system to destination system. (b) In case 2 code blocks are fetched by destination system on demand basis one at the time. (c) In case 4, code blocks are fetched by destination system, but all code blocks are fetched at the same time.

fixed network, where a communication proxy receives requests and forwards them to ultimate destination, and executes necessary DNS queries. Having one communication channel open all the time has additional advantages, supposing that the communication channel is for example TCP socket. This arrangement eliminates the need for three-way handshake which takes 1–1.5 round-trips, and there is no need for slow start in beginning of communication, as an example.

B. *Agent Migration*

In the agent migration case, the agent first moves itself to the vicinity of the communication peer and then uses message passing locally, and thus eliminates the need for message passing over the wireless link. However, the agent should be transferred over the link. The agent code typically consist of several code blocks. Most of the agent systems supporting mobile agents are based on Java. In these systems code blocks are Java classes. In this performance model, however, we do not restrict ourselves to class based systems, but define an agent as a set of arbitrary code blocks. For example, if an agent is coded in Tcl [12], the code blocks can be Tcl source code files.

The agent can be defined as a set of code blocks ($\sum_{i=0}^{n} B_{cbi}$ bytes of code). Furthermore, agent has its own execution state (B_{state}) and arbitrary private data (B_{data}). Furthermore, in some cases, a list of the agent's code blocks is needed (B_{list}). Thus, size of an agent can be described as a four tuple B_{Agent}=($\sum_{i=0}^{n} B_{cbi}$, B_{state}, B_{data}, B_{list}).

There are several possible strategies for transferring an agent from one location to another. The agent transfer schemes we analyze in this paper are based on the ones identified in [13]. In the first case (Fig. 3(a)), the source agent system sends agent's state, data, and all code blocks automatically to destination agent system. This method wastes bandwidth, because destination agent system may have cached some of code blocks, and thus transferring these blocks is unnecessary. After finishing its task, the agent sends a reply message back to original location.

In the second case (Fig. 3(b)), only agent's state and data are transferred automatically. Destination agent system issues requests for missing code blocks on demand basis. This method does not require the source agent system to know all possible code blocks needed before transferring an agent. However, if the source agent system becomes unavailable, and some code blocks are missing, the agent cannot continue its execution at remote site until missing code blocks are transferred. Thus, this strategy is not so suitable to be used in wireless environment because of frequent link disconnections. The third case given in [13] is combination of these first two cases, so that first strategy is used in wireless environment, and second case in fixed network, where it is not so likely that source agent system becomes unavailable.

The last case (Fig. 3(c)) is another combination of the first two. In this strategy, the source agent system sends a list of code blocks necessary to perform the specific agent operation, but it does not send any of the code blocks automatically. Using this list, the destination agent system can request only those code blocks not yet cached. Only one code block request is needed. Even though this method is more efficient, the source agent system has to know the code blocks the agent may need. In the general case this is impossible. Many languages, including Java and Tcl, allow adding new code blocks into agent at runtime. These kind of features make it impossible to create a complete list of needed code blocks. However, it is possible to create an almost complete list and use that. In case of missing code blocks that are not included into the list, second migration scheme can be used.

The network load for migration of an agent using first code block transfer strategy (B_{migI}) can be calculated by

$$(3)\quad B_{migI}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) = \begin{cases} 0 & \text{if } L_1 = L_2, \\ \sum_{i=0}^{n} B_{cbi} + B_{data+state} + (1-\sigma)B_{rep} & \text{otherwise,} \end{cases}$$

where B_{rep} denotes the size of the (optional) reply, and σ denotes the selectivity of the agent, that is, how much the B_{rep} is reduced by remote processing. B_{cr} is the size of code request sent by destination system. If the agent does

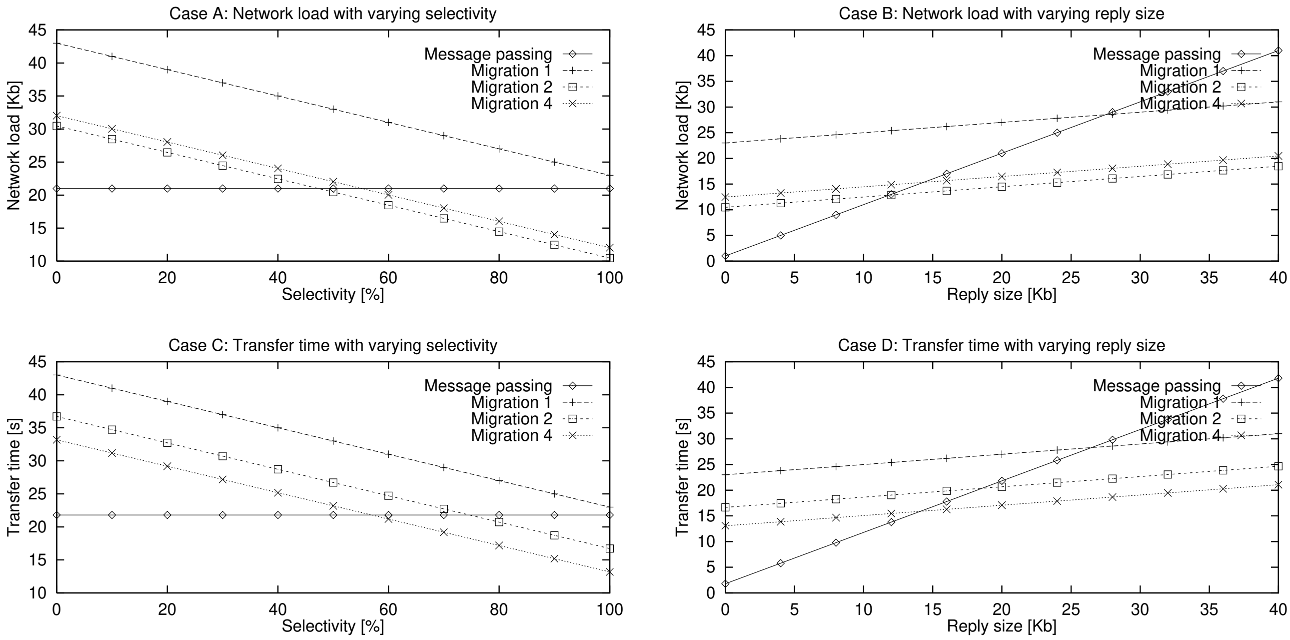


Fig. 4. Evaluation of single interaction. In case A is illustrated network load with fixed reply size and varying selectivity. In case B is shown network load with varying reply size and fixed selectivity. Cases C and D shows corresponding transfer times.

not have to send any reply back to source system, its selectivity can considered to be one.

The network load for migration of an agent using second code block transfer strategy (B_{mig2}) can be calculated by

$$(4)\quad B_{mig2}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) = \begin{cases} 0 & \text{if } L_1 = L_2, \\ \sum_{i=0}^{n} P_i(B_{cr} + B_{cbi}) + B_{data+state} & \\ \quad +(1-\sigma)B_{rep} & \text{otherwise,} \end{cases}$$

where P_i denotes the probability that the code for block B_{cbi} is not yet available at destination location L_2, and B_{cr} is the size of the request to transfer the code.

The fourth case needs a slight modification to equation 4. In this case, a list of code block names (B_{list}) is sent in any case. However, only one request is needed if any of these blocks is missing at remote side. Below is given the equation for calculating network load in the fourth agent transfer case (B_{mig4})

$$(5)\quad B_{mig4}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) = \begin{cases} 0 & \text{if } L_1 = L_2, \\ (1 - \prod_{i=0}^{n} P_i)(B_{cr}) + \sum_{i=0}^{n} P_i(B_{cbi}) & \\ \quad +B_{data+state} + B_{list} + (1-\sigma)B_{rep} & \text{otherwise,} \end{cases}$$

Corresponding transfer times for cases 1,2, and 4 without a reply message are described by

$$(6)\quad T_{mig1}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) = \delta(L_1, L_2) + \frac{B_{mig1}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep})}{\tau(L_1, L_2)}$$

$$(7)\quad T_{mig2}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) = (1 + 2(\sum_{i=0}^{n} P_i)\delta(L_1, L_2) + \frac{B_{mig2}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{req})}{\tau(L_1, L_2)}$$

$$(8)\quad T_{mig4}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) = (3 - 2(\prod_{i=0}^{n}(1 - P_i)))\delta(L_1, L_2) + \frac{B_{mig4}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep})}{\tau(L_1, L_2)}$$

If an agent should send a message back to originator location, one have to add additional delay time to each of these equations.

$$(9)\quad T_{reply\{1,2,4\}}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) = T_{mig\{1,2,4\}}(L_1, L_2, B_{Agent}, \sigma, B_{cr}, B_{rep}) + \begin{cases} 0 & \text{if } L_1 = L_2, \\ \delta(L_1, L_2) & \text{otherwise.} \end{cases}$$

C. Results

Using the performance model given above, an agent can calculate an estimate whether it should migrate itself or whether it should use message passing from the location it resides. Case A in Fig. 4 compares network load volume of message passing to migration while selectivity (σ) is varying between 0% and 100% and reply size is 20kb. The characteristics of the agent and underlying network are given in Table I. The network characteristics are about the same as

TABLE I

AGENT AND NETWORK CHARACTERISTICS

AGENT CHARACTERISTICS:	
Code blocks (B_{cb})	10 * 2kb
Data (B_{data})	2kb
State (B_{state})	1kb
List of block names (B_{list})	1.5kb
Code block request (B_{cr})	0.5kb
Code availability (P)	70%
NETWORK CHARACTERISTICS:	
Throughput (τ)	9600bps
Delay (δ)	400ms

the current GSM network provides with some simplifications. For example, variable delays are not taken into account. Case B in Fig. 4 compares network load volume with fixed selectivity (σ=0.8) while reply size is varying between 0 and 40 kilobytes. Cases C and D in Fig. 4 illustrate corresponding transfer times.

Given the parameters used in this example, migration scheme 1 is the slowest one. In case C it is slower than message passing even if selectivity is 100%. Certainly, if reply message size is bigger than one used in the example, migration scheme 1 will be faster than message passing. Interesting point is that even though the network load in migration scheme 2 is less than in scheme 4, the transfer time for migration scheme 4 less than in scheme 2. This is due to round-trips needed in scheme 2. In worst case, scheme 2 needs $n+1$ round-trips, where n is the number of agent's code blocks, whereas scheme 4 needs only two round-trips. However, in this case, the difference is not significant.

As illustrated in Fig. 4, an agent should use message passing, if the reply size is relatively small or if agent's selectivity is low. In some cases, however, agent should migrate over the wireless link even if message passing seems to be faster. For example, if an agent should have network connection while the mobile device is in a disconnected state, without migrating to fixed network the agent cannot continue its task until the mobile device re-connects. Furthermore, if it is likely that the agent will do the same or similar tasks later, migrating once over the wireless link might be beneficial even if message passing is faster. That is because after one migration process then agent's code can found from fixed network with higher probability, and thus subsequent communication patterns over the link can be accomplished faster. For example, if the selectivity is 30% and other characteristics are the same as in Table I, using message passing takes about 22 seconds, and migration using scheme 4 takes about 27.5 seconds. However, after agent has migrated once over the link, and all its code blocks can be found from fixed network with high probability, migration transfer time is only about 19.5 seconds, while message passing transfer time remains the same. Thus, agents should take into account more than one communication interaction to accomplish their tasks as efficient as possible.

V. CONCLUSION

Implementation of efficient agent communication in mobile wireless environment needs special wireless-aware support from underlying architecture. The architecture should hide the wireless environment from application agents as well as possible, but still allow agents make decisions based on underlying network.

Traditional optimization techniques currently used in wireless environments, such as data compression and caching, are certainly needed, but in addition more intelligent techniques can be used. Especially in a mobile wireless environment, it is important from where an agent should communicate — should the agent first migrate over the wireless link or should it use message passing from the host it currently resides on.

In the near future, we will implement in the Monads [14] agent architecture an efficient agent communication system using optimization techniques given in Chapter III, and an efficient agent migration system, using different migration strategies described in Chapter IV.

ACKNOWLEDGEMENTS

This work was carried out as a part of the Monads research project funded by Sonera ltd, Nokia Telecommunications, and the Finnish Technology Development Centre.

The authors express their thanks to the rest of the Monads team, for the fruitful comments and discussions during the preparation of this paper.

REFERENCES

[1] M. Kojo, K. Raatikainen, and T. Alanko, "Connecting Mobile Workstations to the Internet over a Digital Cellular Telephone Network," Tech. Rep. C-1994-39, University of Helsinki, Department of Computer Science, Helsinki, Finland, Sept. 1994. Revised version published in *Mobile Computing* (T. Imielinski, ed.), pp. 253–270, Kluwer, 1996.

[2] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz, "Improving TCP/IP Performance over Wireless Networks," in *Proceedings of the First ACM International Conference on Mobile Computing and Networking (MobiCom '95)*, (Berkeley, California, USA), pp. 2–11, Nov. 1995.

[3] J. Kiiskinen, M. Kojo, M. Liljeberg, and K. Raatikainen, "Data Channel Service for Wireless Telephone Links," in *Proceedings of the 2nd International Mobile Computing Conference*, (Hsinchu, Taiwan, ROC), pp. 60–69, Mar. 1996.

[4] WAP Forum, "Wireless Application Protocol Forum Home Page." Available at <URL:http://www.wapforum.org>.

[5] M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen, "Enhanced Services for World-Wide Web in Mobile WAN Environment," Tech. Rep. C-1996-28, Department of Computer Science, University of Helsinki, 1996. Revised version published in *Proceedings of the IEEE Global Internet 1996 Conference*, (London, England), pp. 33–37, Nov. 20-21, 1996.

[6] Foundation for Intelligent Physical Agents, "FIPA Home Page." Available at <URL:http://www.fipa.org/>.

[7] Foundation for Intelligent Physical Agents, "FIPA 97 Specification Part 2: Agent Communication Language," Oct. 1998. Version 2.0.

[8] Object Management Group, "CORBA 2.2/IIOP Specification," 1998. Technical Report, formal/98-07-01.

[9] Sun Microsystems, "Java Remote Method Invocation – Distributed Computing for Java," Mar. 1998. White Paper.

[10] T. Chia and S. Kannapan, "Strategically Mobile Agents," in *Proceedings of the First International Conference on Mobile Agents* (K. Rothermel and R. Popescu-Zeletin, eds.), vol. 1219 of *Lecture Notes in Computer Science*, (Berlin), Springer-Verlag, Berlin, Apr. 1997.

[11] M. Straßer and M. Schwehm, "A Performance Model for Mobile Agent Systems," in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'97)* (H. R. Arabnia, ed.), (Las Vegas, USA), pp. 1132–1140, 1997.

[12] J. K. Ousterhout, *Tcl and the Tk Toolkit*. Reading, MA, USA: Addison-Wesley, 1993.

[13] GMD FOKUS and IBM Corporation, "Mobile Agent System Interoperability Facilities Specification," Nov. 1997. Standard proposal.

[14] Monads research group, "Monads Home Page." University of Helsinki, Department of Computer Science. Available at <URL:http://www.cs.helsinki.fi/research/monads/>.