Using Mobile and Intelligent Agents to Support Nomadic Users

Stefano Campadello[1], Heikki Helin[2], Oskari Koskimies[1], Pauli Misikangas[1], Mikko Mäkelä[1], Kimmo Raatikainen[1]

[1]Department of Computer Science, P.O. Box 26 (Teollisuuskatu 23), FIN-00014 UNIVERSITY OF HELSINKI, Finland

[2]Sonera Ltd, P.O. Box 970 (Teollisuuskatu 13), FIN-00051 SONERA, Finland

Email: {stefano.campadello, oskari.koskimies, pauli.misikangas, mikko.makela, kimmo.raatikainen}@cs.helsinki.fi, heikki.j.helin@sonera.fi

Abstract

The environment of nomadic computing is very different from that of traditional distributed systems today. The variety of mobile workstations, handheld devices, and smart phones, which nomadic users use to access data services in the Internet, increases at growing rate. The outcome is new demands for adaptability of data services. Agent technology is one of the software solutions that may be used to fulfill the demand of "anytime-anywhere-anyhow" access to data services. The research project Monads addresses some of the fundamental challenges in supporting nomadic applications: adaptability to available computing and communication resources that vary in tempo-spatial space and short-term (1-30 minutes) predictions of available resources. In addition to adaptability and predictions, efficient agent communication in wireless environments is a mandatory prerequisite.

1. Introduction

Current information services based on the Internet, such as the WWW, are designed for workstations in fixed wireline networks. Wireless data services—current GSM Data Service as well as GPRS, UMTS, IMT-2000, wireless ATM and various satellite systems—enrich the options for communications. Unfortunately, the current Internet solutions are not able to fulfill all the needs of nomadic users. According to Prof. Leonard Kleinrock [1], "Nomadic computing and communications will dramatically change the way people access information—and a paradigm shift in thinking about applications of the technologies involved. It exploits the advanced technologies of wireless, the Internet, global positioning systems, portable and distributed computing to provide anytime, anywhere access. It is beginning to happen, but it makes everything much harder for the vendors." The benefits include increased productivity, "the personal touch", personal environment, interactivity, setting.

The environment of mobile computing is in many respects very different from the environment of the traditional distributed systems of today. Bandwidth, latency, delay, error rate, interference, interoperability, computing power, quality of display, and other non-functional parameters may change dramatically when a nomadic end-user moves from one location to another—from one computing environment to another, for example from a wired LAN via a wireless LAN to a GPRS/UMTS network. The variety of mobile workstations, handheld devices, and smart phones, which nomadic users use to access Internet services, increases at a growing rate. The CPU power, the quality of display, the amount of memory, software (e.g. operating system, applications), hardware configuration (e.g. printers, CDs), among other things ranges from a very low performance equipment (e.g. hand held organizer, PDA) up to very high performance laptop PCs. All these cause new demands for adaptability of data services. For example, palmtop PCs cannot properly display high quality images designed to be looked at on high resolution displays, and as nomadic users will be charged based on the amount of data transmitted over the GPRS network, they will have to pay for bits that are totally useless for them.

Confronted with these circumstances, the nomadic end-user would benefit from having the following functionality provided by the infrastructure: Information about expected performance provided by agents, intelligent agents controlling the transfer operations, a condition-based control policy, capability provided by intelligent agents to work in a disconnected mode, advanced error recovery methods, and adaptability.

The ability to automatically adjust to changes in the environment mentioned above in a transparent and integrated fashion is essential for nomadicity—nomadic end-users are usually professionals in other areas than computing. Furthermore, today's distributed systems are already very complex to use as a productive tool; thus, nomadic end-users need all the support, which an agent based distributed system could deliver. Adaptability to the changes in the environment of nomadic end-users is the key issue. Intelligent agents could play a significant role in implementing adaptability. One agent alone is not always able to make the decision how to adapt, and therefore adaptation is a co-operation effort carried out by several agents. Therefore, there should be at least some level of cooperation between adapting agents.

Software agent technology has gained a lot of interest in the recent years. It is widely regarded as a

promising tool that may solve many current problems met in mobile distributed systems. However, agent technology has not yet been extensively studied in the context of nomadic users, which exhibits a unique problem space.

The research project Monads examines adaptation agents for nomadic users. In the project we have designed a software architecture based on agents and we are currently implementing its prototypes. Our goal is not to develop a new agent system; instead, we are extending existing systems with mobility-oriented features. The Monads architecture is based on the Mowgli communications architecture [2] that takes care of data transmission issues in wireless environments. In addition, we have made use of existing solutions, such as OMG [3] and FIPA specifications [4] as well as Java RMI [5] as far as possible. However, direct use was not sufficient but enhancements for wireless environments were necessary [6, 7, 8]. Currently we are using Jade [9] as the underlying agent system.

The research field of agent technology is huge. Therefore, the Monads project has concentrated on the needs of nomadic users and on adaptability. By adaptability we primarily mean the ways in which services adapt themselves to properties of terminal equipment and to characteristics of communications. This involves both mobile and intelligent agents as well as learning and predicting temporary changes in the available Quality-of-Service along the communications paths. The fundamental challenge in nomadic computing is dynamic adaptation in the triad service–terminal–connectivity (see Figure 1) according to preferences of the end-user.

Dynamic adaptation of a service to the properties of terminal equipment and available communication infrastructure is an attractive feature. To give an example: When the connection is slow, an e-mail agent does not fetch a whole mail folder to a mobile terminal but only the headers and the user can select which mail entries are to be fetched. The faster connection enables mail folders to be fetched automatically to the mobile terminal. The same kind of scenario goes with Web browsing. When the network connection is slow enough, the browser agent may automatically use different kinds of compression methods or even refuse to fetch certain objects.

The rest of the paper is organized as follows. In Sections 2 and 3 we address the key areas of research and development: optimizing communications in wireless environments and predicting Quality-of-Service using intelligent agents. Finally, in Section 4 we state our conclusions.

2. Optimization of Communications

When agents supporting nomadic users are developed, optimization of agent communication for wireless environments is one of the key issues. In this section we discuss agent communication and its optimization. We also give brief descriptions of some communication services available in the Monads agent platform.

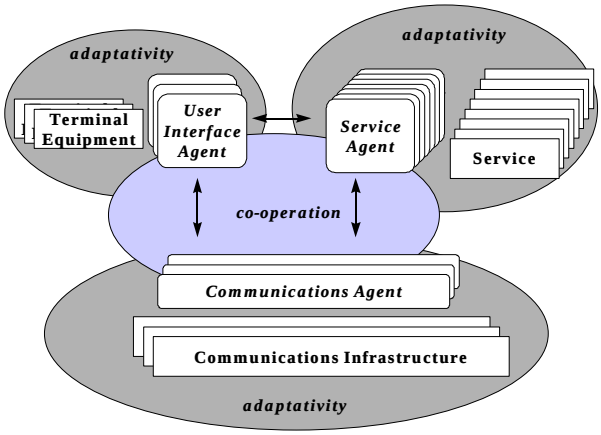

Figure 1: Monads Adaptation Framework

2.1 Agent Communication in Wireless Environments

Communication between agents can be divided into four layers [7]:

1. The *Interaction Protocol Layer* contains high level protocols for an interaction between the communicating parties. Various negotiation protocols and communication patterns, such as FIPA interaction protocols [10], belong to this layer.
2. The *Communication Language Layer* defines the content of messages exchanged between the parties. Examples of agent communication languages belonging to this layer include FIPA ACL [10] and KQML [11].
3. The *Message Transport Layer* deals with messaging protocols such as HTTP [12], Java RMI [5], GIOP [13], and higher level WAP [14] protocols.
4. The *Transport and Signaling Protocol Layer* implements network transport mechanisms, such as TCP/IP, lower level WAP protocols, GSM Short Message Service, and MDCP [2, 15].

When agent communication is implemented for wireless environments, optimizations are needed in each layer. The research project Monads—together with its companion projects (Mowgli, IWTCP, wCORBA [16])—addresses optimization of communication over wireless networks. In FIPA we are actively involved in nomadic application support

and in bit-efficient ACL. In OMG [3] we participate in a proposal for wireless access and terminal mobility in CORBA [17, 18]. In IETF we work in the PILC working group [19] that address the problems of TCP in low-bandwidth/long-latency networks. In addition, we have obtained interesting results in improving the performance of Java 2 RMI in wireless environments [8].

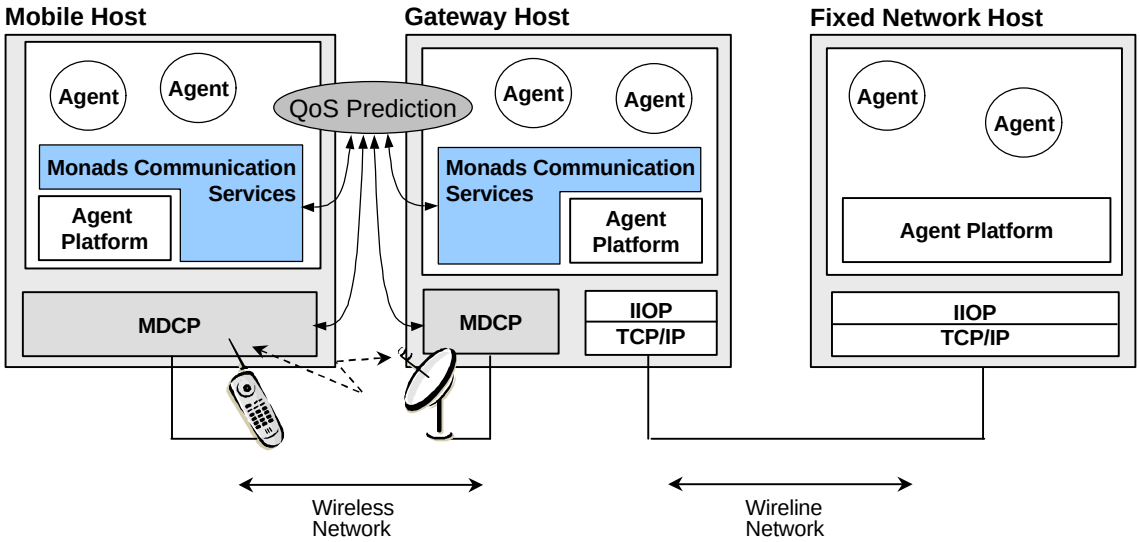

Figure 2: Monads Communication Architecture

On the interaction protocol layer one possibility of optimizing agent communication is to redesign communications patterns. Instead of sending many small messages, coupling several messages together into a single message can improve the efficiency of link usage significantly.

Optimization in the communication language layer includes compression and conversion between media types. For example, bit-efficient representation of FIPA ACL may utilize the link usage more efficiently than string-based representation. In addition to efficient coding of ACL messages, the actual content of the message should also be compressed or converted to a more appropriate form, if possible.

The message transport layer protocols should also be optimized. Protocols like IIOP or Java RMI can be used as message transport protocols but they are quite inefficient in wireless environments. Both of them have a high protocol overhead.

Location transparency in wireless environments needs special support. To name an example, disconnections of a wireless link should be hidden from communicating agents so that they can be found efficiently even if the mobile device is reconfigured after link disconnection. The message transport layer should allow agents to use the same identifiers even if the IP-address of the mobile device changes.

The transport layer should provide an efficient and reliable transport service. It should be transparent to the agent. The transport layer could, for example, automatically select the appropriate protocol to use. When the mobile device is disconnected, the transport layer may use SMS to deliver messages, if such a 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. Although the transport layer should hide the effects of mobility and a wireless environment as far 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 a wireless link.

2.2 Monads Communication Services

The Monads Communication Services are used for communication between agents, agent systems and users, and for transporting agents between Monads agent platforms. Figure 2 depicts the Monads communication architecture, in which MDCP [2, 15] is used as a wireless transport and signaling protocol.

On a higher level, ACL communication between agents is optimized by using bit-efficient, tokenized FIPA ACL [20]. The tokenized encoding reduces the overhead of the rather verbose string-encoded ACL headers to a minimum. Similar syntax-based encoding techniques are used for example to optimize HTTP [21] and XML [22]. Additional improvements in ACL messaging can be obtained by utilizing information about the communication pattern that can be used to further optimize communication.

Java RMI is the standard communication method of distributed computing in Java. It is also utilized by most Java-based agent platforms as the message

transport layer. Its efficiency in wireless environments is therefore crucial. Unfortunately, due to its high protocol overhead, both in data traffic and in round-trips, Java RMI is poorly suited to wireless communication. However, it can be optimized without

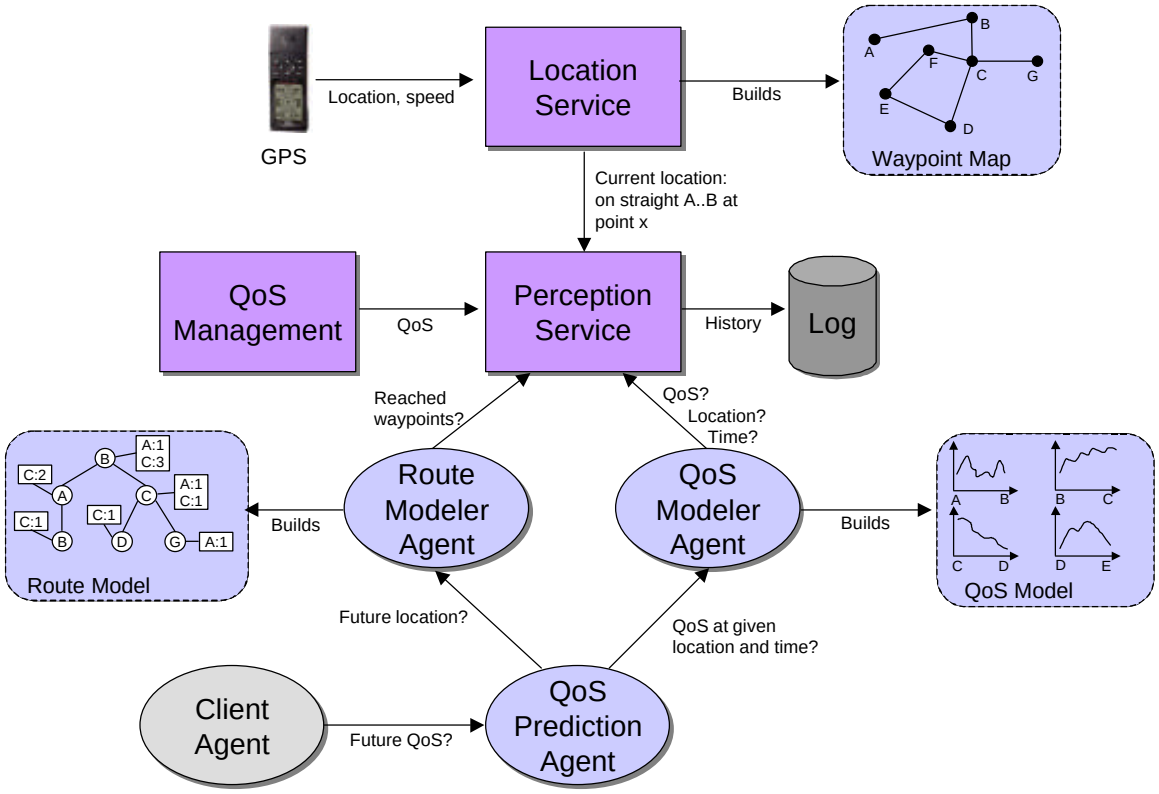

Figure 3: Learning and predicting Quality-of-Service

breaking compatibility with Java RMI specification, and with minimal changes to existing software and network hosts. New software is necessary only at the mobile terminal and at its access point to the fixed network. This is possible by utilizing mediator technology, which is widely exploited in wireless communications. In our prototype implementation, the number of round-trips for a simple invocation was reduced to one third, and the time required to complete the invocation was reduced by 73%. In addition, we can hide temporary disconnection from agents, and mobile-aware agents can attach communication attributes, such as priority, to invocations.

The *Monads Agent Transfer Service* allows agents to move between Monads agent systems. The service is optimized for wireless links by compressing data and minimizing protocol overhead, and is used for transferring agents between a mobile node and the fixed network. For agent transfers within the fixed network, the transfer service offered by the underlying platform is usually used. If transfer to a disconnected host is attempted, the agent is blocked until the host is available, or (if the agent is mobile-aware) until a specified timeout has occurred.

3. Adaptation by Learning

In the Monads adaptation, adaptation is mainly achieved by learning. So far, our main focus has been on learning to predict Quality-of-Service (Section 3.1) [23]. However, various other things about the environment and behavior of the user can also be learned and used for adaptation. For instance, by observing the order in which the user reads his/her e-mails, we could learn which e-mails should be sent first.

It is quite easy to find uses for learning, but putting ideas into practice may be a difficult and time-consuming task. Learning agents are usually *ad hoc* systems, the learning algorithm written directly into the agent program code and parameters tuned by hand for certain learning task. The obvious disadvantage of this kind of programming practice is that the agent cannot be used for any other learning task without modifying the program code. We have used a lot of design efforts to get rid of this problem. As a result, we have a system architecture that supports learning and knowledge sharing. The most important components of the Monads learning services are briefly described in Section 3.2.

3.1 Predicting Quality-of-Service

Software systems that are to be used in wireless environments should be able to adapt to sudden changes in the quality of data transmission over wireless connections. As a minimum, a system should detect when current data transmission tasks may not be completed any longer in a reasonable amount of time due to temporary changes in the QoS. More sophisticated systems could try to adapt to the current QoS by using special data filtering and compression methods, and to refuse to accept requests that cannot be fulfilled within a certain time limit. A good example is a Web browsing agent that automatically shrinks or ignores large images on the requested Web pages when the QoS is not good enough. However, quite often an adaptation that is started right after a change in QoS is detected comes too late. This is especially true when the connectivity was just lost—nothing can be done after detection of a disconnection, but something could have been done beforehand if the system would have been able to predict the disconnection.

Predicting changes in QoS of wireless links will be one of the fundamental requirements for future systems that are supposed to do intelligent adaptation in wireless environments. Estimates of future QoS can be used, for example,

- in scheduling decisions, e.g. which tasks are allowed to use bandwidth when the connection is about to be lost,
- in data prefetching, e.g. download something beforehand while the connection is still good, and
- in connection management, e.g. close the connection immediately to save expenses because the QoS will be inferior during the next 5 minutes.

We believe that predictions of useful accuracy can be made by learning how quantities like time of day, day of week, past QoS values, and location of the terminal affect the QoS.

We have divided the problem of QoS prediction into two sub-problems: predicting terminal movement, and predicting QoS at a given location. The sub-problems are handled by two learning agents, called *Route Modeler Agent* and *QoS Modeler Agent*, that actively collect data from the environment and build/update models that are used for prediction; see Figure 3. Information about the current location and QoS is produced by *Location Service* and *QoS Management Service*, and is delivered to modeler agents via *Perception Service* that is described in the next section. Both models are based on the use of waypoints; instead of expressing a location as coordinates, it is given as a point on a straight line between two waypoints. This reduces necessary calculation and speeds up learning.

An intelligent agent called *QoS Prediction Agent* combines the predictions of future location and QoS. It provides answers to questions like

- what is the expected amount of data we can transfer within the next *t* seconds?
- what is the expected time to transfer *x* kilobytes of data?
- what is the probability that we can successfully transfer *x* kilobytes of data within the next *t* seconds?

Application agents can then adapt their behavior according to the answers. For example, an e-mail agent may decide not to send a large file attached to an e-mail automatically, because of prediction that transferring that file would take at least 3 minutes with 90% certainty. The QoS Prediction Agent is also responsible for choosing between alternative ways to predict movement and QoS. For example, sometimes a calendar agent knows best where the user is going. The QoS Prediction Agent may also inform other agents when the QoS is about to change significantly.

3.2 Monads Learning Services

The *Perception Service* is responsible for collecting values of perceptions within some time intervals and storing the values. By perception we mean anything that can be observed and be of use for learning agents. This may be, for example, information about system events or the status of some continuous or discrete quantity such as the battery level of a laptop computer or communication bandwidth. This centralized handling of perceptions offers many advantages. Decisions on which parts of the collected data should be flushed and when, are much easier to do with centralized data than if all agents had to do this by themselves. Further, previously unknown correlations are more likely to be found with this approach. Also, refining data (e.g. clustering) and avoiding overlapping data is easier.

The *Learning Service* allows the definition of new learning tasks and offers several learning algorithms that can be used to solve those tasks. A learning task consists of one or more input attributes that are used to decide the value of a single output attribute. Learning tasks may be traditional classification problems in which the output value depends only on the given set of input values, or prediction problems in which the history of input and output values must also be taken into account. Actual learning is done by modeler agents that actively try to build models, such as decision trees (see e.g. [24]), for some learning task using data collected with the help of the Perception Service or given by a client agent.

In practice, there may not be enough data available for reasonably fast and accurate learning. The most used solution to this problem is to put all data from several sources together. However, because we cannot transfer much data over the wireless, the sharing of data among modeler agents in different terminals is not possible. Therefore, we share models instead of data. Modeler agents of the same type cooperate via *Knowledge Sharing Service* by exchanging models and testing models made by others. Based on the results of those tests, model combiner agents try to combine the best parts of each model and build new and improved models (see e.g. [25]). The Monads architecture also gives a nice framework for the use of meta-learning agents that build meta-models by learning from the output of other models [26].

4. Conclusions

We have introduced the Monads agent architecture and described the basic set of services available in the Monads system. The architecture and system services were designed to fulfill the adaptation requirements that will be necessary to support nomadic users in the near future.

The fundamental challenge in nomadic computing is dynamic adaptation in the triad service–terminal–connectivity according to preferences of the end-user. The Monads agent architecture and the Monads system services as described in this paper is one possible solution to meet future challenges in nomadic computing. The communication services together with a wireless transport and signaling protocol—MDCP [2, 15] in our case—provide an efficient and reliable transfer infrastructure that can handle sudden drops of wireless links and cope with variable throughput and error rates. Other Monads services are built on that infrastructure in order to provide an adaptive platform for agents supporting applications for nomadic users.

5. Acknowledgements

The authors are grateful to their colleagues in the Monads project, in particular to Markku Tamski from Nokia Mobile Phones, to Heimo Laamanen from Sonera, and to Mustafa Abdulla and Sasu Tarkoma from the Monads group at the University of Helsinki.

6. References

[1] L. Kleinrock, “Nomadicity: Anytime, Anywhere in a Disconnected World,” Mobile Networks and Applications, Vol. 1, No.4, January 1997, pp. 351-375.

[2] M. Kojo, K. Raatikainen, M.Liljeberg, J.Kiiskinen, and T.Alanko, “An Efficient Transport Service for Slow Wireless Telephone Links,” IEEE Journal on Selected Areas in Communications, Vol. 15, no. 7, pp. 1337–1348, Sept. 1997.

[3] Object Management Group, “OMG Home Page,” http://www.omg.org/.

[4] Foundation for Intelligent Physical Agents, “FIPA Home Page,” http://www.fipa.org/.

[5] Sun Microsystems, Java Remote Invocation – Distributed Computing for Java, White Paper, 1998.

[6] M. Liljeberg, K. Raatikainen, M. Evans, S. Furnell, N. Maumon, E. Veltkamp, B. Wind, and S. Trigila, “Using CORBA to Support Terminal Mobility,” Proceeding of TINA’97 Conference, IEEE Computer Society Press, pp. 56–67, 1998.

[7] H.Helin, H. Laamanen, and K. Raatikainen, “Mobile Agent Communication in Wireless Networks,” Proceedings of the European Wireless’99 Conference, pp. 211-216, October 6–8, 1999, Munich, Germany.

[8] S. Campadello, H. Helin, O. Koskimies, and K. Raatikainen, “Optimizing Java 2 RMI for Slow Wireless Links,” Submitted for publication.

[9] F. Bellifemine G. Rimassa, and A. Poggi, “JADE – A FIPA-compliant Agent Framework,” http://www.practical-applications.co.uk/PAAM99/abstracts.html.

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

[11] Specification of the KQML agent communication language, http://www.cs.umbc.edu/kqml/kqmlspec/spec.html.

[12] Internet Engineering Task Force, Hypertext Transfer Protocol – HTTP/1.1, RFC2068, 1997.

[13] Object Management Group, CORBA 2.2/GIOP Specification, OMG Document formal/98-07-01, 1998.

[14] Wireless Application Protocol Forum, WAP Specifications, http://www.wapforum.org/, 1998-9.

[15] J. Kiiskinen, M. Kojo, M. Liljeberg, and K. Raatikainen, “Data Channel Service for Wireless Telephone Links,” IEEE-CS Bulletin of TC on Operating Systems and Applications, Vol. 8, 1, pp. 1-17, 1996.

[16] The Department of Computer Science in the University of Helsinki, ”Home Pages of Research Projects,” http://www.cs.helsinki.fi/research/.

[17] Object Management Group, Telecom DTF White Paper on Wireless Access and Mobility in CORBA, OMG Document telecom/98-11-09, 1998.

[18] Object Management Group, Request for Proposals on Wireless Access and Terminal Mobility in CORBA, OMG Document telecom/99-05-05.

[19] Internet Engineering Task Force, “PILC (Performance Implications of Link Characteristics) Home Page,” http://pilc.grc.nasa.gov/pilc.

[20] Foundation for Intelligent Physical Agents, FIPA 99 Specification – Part 16: Agent Message Transport, 1999, *work in progress*.

[21] M. Liljeberg, H. Helin, M. Kojo, and K. Raatikainen, “MOWGLI WWW Software: Improved Usability of WWW in Mobile WAN Environments,” in Proceedings of IEEE Global Internet 1996 Conference, 1996, pp. 33-37.

[22] WAP Forum, “Binary XML Content Format Specification”, Version 1.1, June 16, 1999.

[23] P. Misikangas, M. Mäkelä, and K. Raatikainen, “Predicting QoS for Nomadic Applications Using Intelligent Agents,” Proceedings of IMPACT’99 Workshop, December 2-3, 1999, Seattle, Wash., USA.

[24] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, 1993.

[25] K.M. Ting and B.T. Low, “Model Combination in the Multiple-Data-Batches Scenario,” in M. van Someren and G. Widmer (eds.), Machine Learning: ECML-97, Lecture Notes in Artificial Intelligence 1224. 1997, pp. 250–265, Springer-Verlag.

[26] A.L. Prodromidis, P.K Chan, and SJ. Stolfo, “Meta-Learning in Distributed Data Mining Systems: Issues and Approaches,” in Kargupta and Chan (eds.), Advances in Distributed Data Mining, AAAI Press, 1999.