Monads – Adaptation Agents for Nomadic Users

Kimmo Raatikainen
Department of Computer Science, University of Helsinki, Finland
E-mail: Kimmo.Raatikainen@cs.Helsinki.FI

Lassi Hippeläinen
Nokia Telecommunications
E-mail: Lassi.Hippelainen@nokia.com

Heimo Laamanen
Sonera Ltd
E-mail: Heimo.Laamanen@sonera.fi

Matti Turunen
Nokia Mobile Phones
E-mail: Matti.Turunen@nmp.nokia.com

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 very different compared to today's environment of traditional distributed systems in many respects. Bandwidth, latency, delay, error rate, interference, interoperability, computing power, quality of display, among other things may change dramatically as a nomadic end-user moves from one location to another—from a computing environment to another, for example from a wired LAN via a wireless LAN (from an office) to a GPRS/UMTS network (to the field). 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 display properly 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.

A nomadic end-user confronted with these circumstances 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 computers, and today's distributed systems are already very complex to use as a productive tool; thus, nomadic end-users need all the possible 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 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.

Agent technology based on software 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 that exhibits a unique problem space.

The rest of the paper is organized as follows. In Section 2 we give an overview of the research project Monads and its results so far. We discuss requirements and goals, present our reference configuration, and introduce the Monads Agent Architecture and System Services. In Sections 3 and 4 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 5 we state our conclusions.

2. MONADS OVERVIEW

The research project Monads—carried out by the Nomadic Computing Group in the Department of Computer Science at the University of Helsinki, Nokia Mobile Phones, Sonera, and Nokia Telecommunications; funded also by the National Technology Agency

of Finland (TEKES)—examines adaptation agents for nomadic users. The project started in February 1998 and is scheduled to run until December 2000.

In the project we have designed a software architecture based on agents and we are currently implementing prototypes based on the Monads architecture. The software architecture of Monads is based on the Mowgli communications architecture [2] that takes care of data transmission issues in wireless environments. In addition, we have made us of existing solutions, such as OMG and FIPA specifications as well as Java RMI as far as possible. However, direct use was not sufficient but enhancements for wireless environments was necessary [3, 4, 5].

The research field of agent technology is huge. Therefore, in the Monads project we have 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 goal of Monads is to design an efficient and reliable software architecture based on adaptive software agents and to develop prototypes based on that architecture. The objective of the prototype implementation is to evaluate the functionality of the essential features of the designed architecture.

Dynamic adaptation of a service to the properties of terminal equipment and available communication infrastructure is an attractive feature. To name an example, having a slow connection, the e-mail agent does not fetch whole mails to a mobile terminal but shows only mail headers and the user can select which ones to fetch. The faster connection enables mails to be fetched automatically to the mobile terminal. The same kind of scenario goes with WWW 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.

2.1 REQUIREMENTS AND GOALS

The Monads system is targeted for nomadic users with mobile computers (laptops or future advanced PDAs) that need to be connected to a fixed network. The connectivity varies according to time and location: no connection at all, a narrowband wireless link (like GSM Data or GPRS), wireless LAN, narrowband fixed modem link, or high-speed LAN.

Besides dynamic adaptation to various kinds of terminal equipment and to changing network conditions, the Monads architecture should meet the following requirements. Firstly, the system must operate even if the terminal device is disconnected. The length of disconnection may vary from temporary failures in the underlying wireless network infrastructure to much longer user initiated disconnections. Secondly, the system must optimize the bandwidth it uses over the wireless link. This can be done by running optimized versions of protocols, compressing data, and reducing interactions between communicating peers, for example. Thirdly, the system must provide an easy-to-use application programmer's interface for developing mobile-aware software agents. However, it is inadmissible to design an easy to use architecture that suffers performance problems, and therefore is useless for developing applications to mobile wireless environments. Finally, the system must support communications with legacy software.

This means that the system must support at least some standard protocols such as IIOP, SNMP, HTTP and SMTP.

The system should help nomadic users in the following ways:

– Improve the efficiency of existing network applications like Web browsers. This should be possible with none or minor modifications to these applications.

– Provide a base for building Monads-specific applications that take full advantage of the Monads system, and therefore are able to use wireless links more sensibly than regular applications.

– Optimize the establishment, configuration and maintaining of wireless connections with regard to cost and response time. The cost of a connection is considered to be based on:

  1. Connection duration
  2. Bandwidth of the wireless link (might vary during the lifetime of the connection)
  3. Amount of data transferred
  4. Time (e.g. cheap rates at night)

The way these factors affect the total charge varies for different connection types. Often connections are quite expensive, and in those cases the system should carefully consider when the connection is really needed, and work off-line whenever possible.

2.2 MONADS AGENT ARCHITECTURE

It is well-known that fixed network services often work poorly in a wireless environment [6, 7], and have to be adapted to the wireless environment if they are to work efficiently. An example of this is the Mowgli WWWsoftware [8], which adapts Web browsing for low bandwidth wireless links.

2.2.1 Adaptation Model

The Monads system is based on the idea of adaptive, collaborative agents, as depicted in Figure 1. Each type of agent encapsulates knowledge about its particular domain: *Data Communication Agents* understand the properties of different communication infrastructures, such as GSM Data, WLAN or GPRS. *User Interface Agents* know the capabilities of the terminal, such as its display type. And finally, *Service Agents* are aware of the constraints that apply to their service, such as the minimum bandwidth it needs, and know if and how it can be scaled down for low-bandwidth connections.

Each of these agent types uses their knowledge for adaptation, both internally and collaboratively. Data Communication Agents adapt to the communication infrastructure, so that service agents do not have to concern themselves with how Quality of Service constraints are mapped to the parameters understood by the communication infrastructure. User Interface Agents adapt to the capabilities of the terminal device, so that services do not have to concern themselves with the capabilities of the terminal.

However, if the service is to operate efficiently, service agents themselves must also adapt. There is no sense in trying to send high-resolution real-time video over a link that does not

have enough bandwidth, for example. Instead, the video should be scaled down, and the frame rate lowered, so that transfer is possible. A service agent can also improve performance if it understands about terminal capabilities. For instance, if a terminal cannot show color images, transferring them in color to the terminal is inefficient, even if the user interface agent is able to adapt by converting the images to monochrome. Instead, the service should convert the images prior to transfer, so that bandwidth is not wasted.

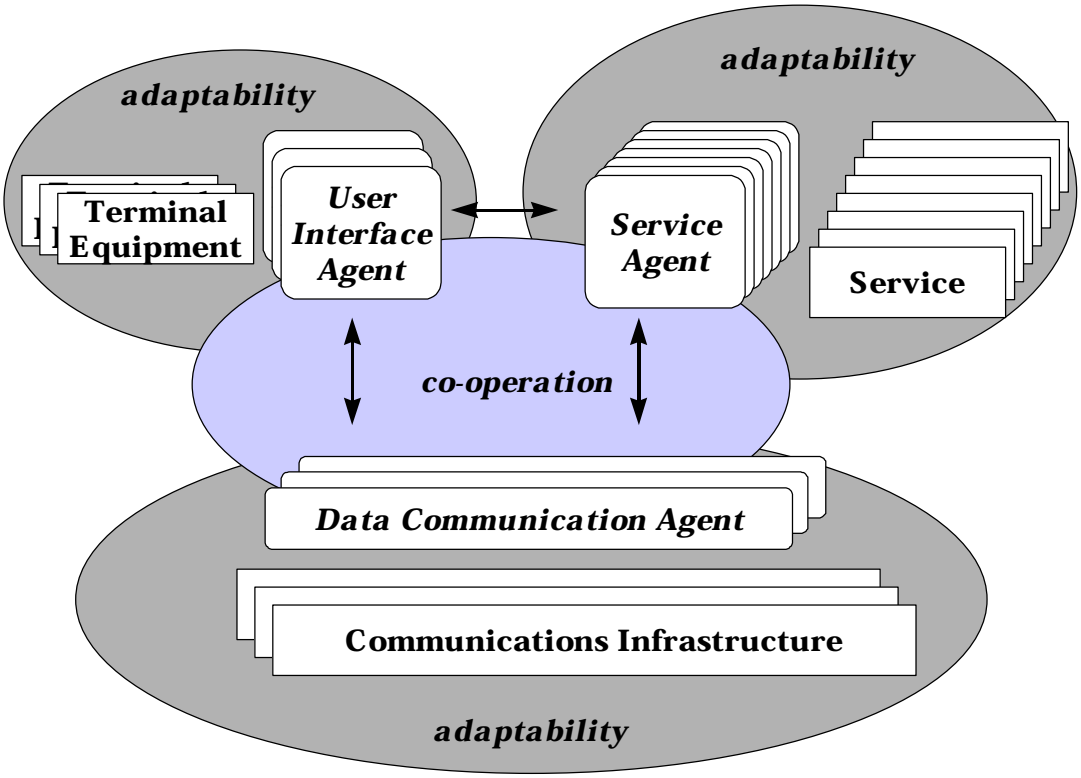

Figure 1: Monads Adaptation Model

2.2.2 Reference Configuration

Figure 2 outlines the native computing and communication environments in Monads. The key elements are terminals, access nodes, and service nodes. In the Monads architecture a mobile terminal device is connected to the fixed network through a weak connection, ranging from slow wireless networks to high-speed fixed networks. Supported wireless data network technologies will include existing technologies, such as GSM Data and Wireless LAN. In the future we will also support GPRS, perhaps also some WAP protocols, and other next generation wireless technologies.

The access node is a fixed host in the fixed network, which provides connectivity for mobile terminals to the fixed network. An access node can be hosted by a public service provider, or it can be located in the private network of an organization. The service nodes are hosts in fixed networks providing different kinds of services to both nomadic users and users using wired connections.

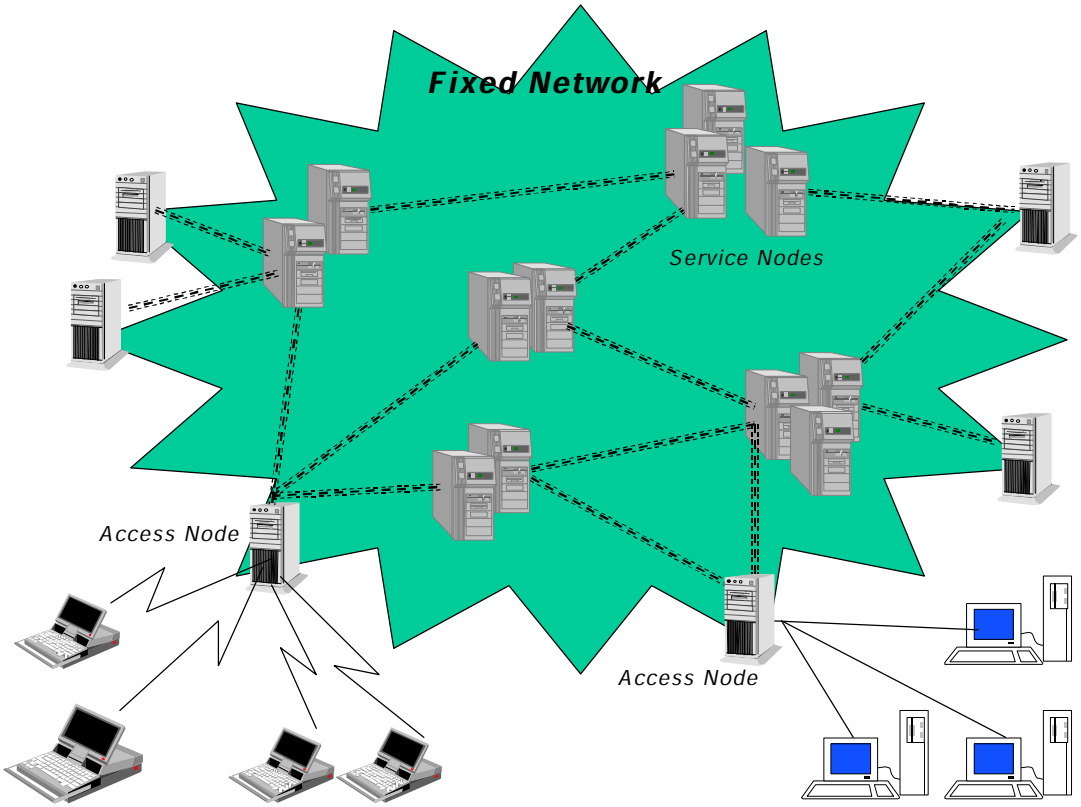

Figure 2: Monads System Reference Configuration

2.2.3 Layered Architecture

The Monads architecture, outlined in Figure 3, is a layered architecture that tries to utilize existing solutions, which will be extended with Monads System Services. Monads Service Agents implement the adaptation needed by a particular application, by Web browsing or by e-mail, for example. On the uppermost level there are standard, non-Monads applications such as web browsers or email clients. However, some applications may be implemented entirely by Monads Service Agents. It should be noted that agent-based Monads applications may entirely be executed on the mobile terminal, may be partitioned between the mobile terminal and fixed network, or may entirely be executed in the fixed network. The latter is especially important when using terminal equipment without significant computing capabilities.

In Monads we take transport and signaling protocols as they are. We only assume that they are optimized for wireless environments. In our prototype implementation we use the MDCP protocol from the Mowgli project [2], which is an experimental transport protocol for low-bandwidth and error-prone wireless links. The MDCP protocol hides most of the problems a wireless communication link creates. The message transport layer (Messaging Services) needs to be mobile-aware and optimized for wireless environments. In Monads we are working on this issue. In our prototype the Messaging Services will be Monads System Services but we hope that in the future there will be international standards. Monads System Services also utilize existing Java based agent platforms like Voyager [9] or JADE [10].

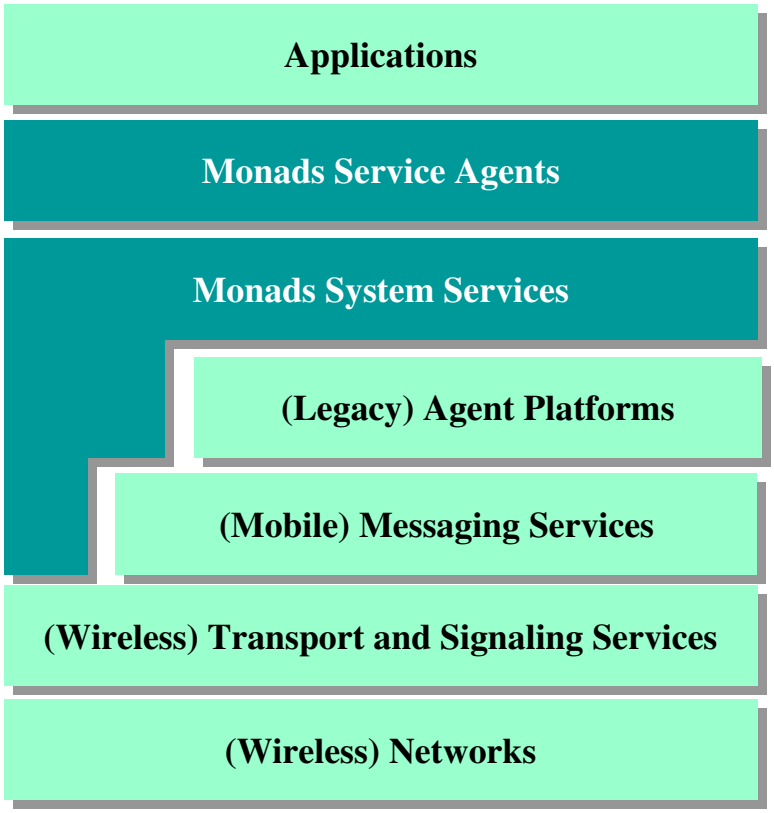

Figure 3: Monads Layered Architecture

2.3 MONADS SYSTEM SERVICES

The services offered by the Monads System are grouped in eight service groups. The system as a whole is presented in Figure 4. The groups of Monads System Services are:

Management Services: Managing agents, agent servers and system services and resources. The services include Agent Server Management, User Agent Management, (dynamic) Resource Management, Persistence and Activation Service.

Communication Services: Communication between agents, agent systems and users, and transporting agents between agent servers. The services include Agent Transfer Service, Data Transfer Service, Stream Service, QoS Management, Event Service, RMI Service, User Interface Service. These services are described in Section 3.2.

Tracing Services: Storing information about the status of the system. The services include Perception History, Log Service, Account Management, Performance Tracing, Debug service. Those services that are used in QoS prediction are discussed in Section 4.

Storage Services: The services include Caching Service and Database Service.

Brokering Services: White and Yellow Pages implemented in Naming Service and Brokering Service.

Knowledge Services: Creating predictive models and sharing information. The services include Learning Service and Knowledge Sharing Service. These services are described in Section 4.

Profile Management Services: Managing user and terminal profiles through User Profile Management and Terminal Profile Management. ***Security Services***: Methods for protecting, authenticating and monitoring agents, agent systems and users. Security is not explicitly addressed in Monads. Instead we assume that CORBA Security [11], forthcoming specifications by Foundation for Intelligent Physical Agents (FIPA) [12] and results from some EC/ACTS Climate projects [13], such as Scarab, can be exploited in the Monads agent architecture.

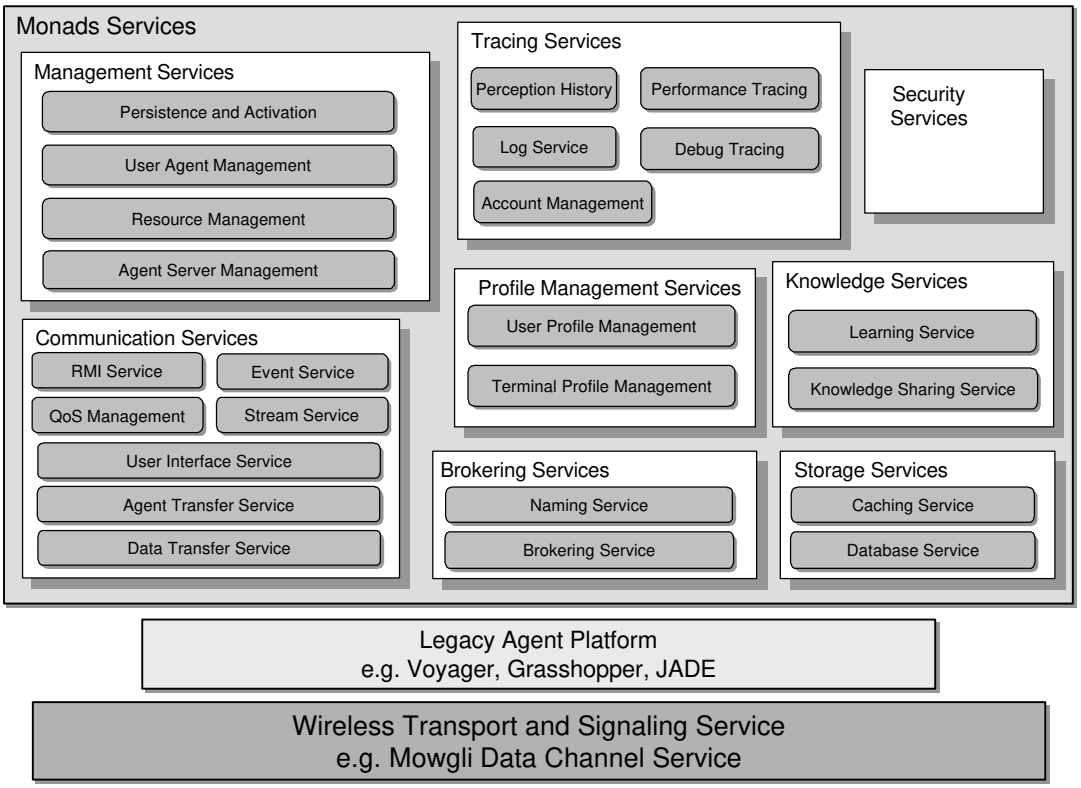

Figure 4: Monads System Services

3. OPTIMIZATION OF COMMUNICATIONS

3.1 Agent Communication in Wireless Environments

Five different scenarios of agent communication in a wireless environment are outlined in Figure 5. In Case 1 the mobile device is powerful enough to run a full agent system, maybe a lightweight one, and a number of agents. In Case 2 the mobile device is a PDA that is too "small" to run a full agent system; not even a lightweight one. The device, however, has a stand-alone agent control tool so that the user can start agents in the fixed network and get results back, for example. In both cases the mobile device communicates with a Terminal Communication Agent (TCA) in the fixed network. The TCA acts as a proxy for the mobile device. It implements a set of Monads communication services, mobile messaging

services, and wireless transport and signaling services. The TCA also takes care of disconnections and handoffs.

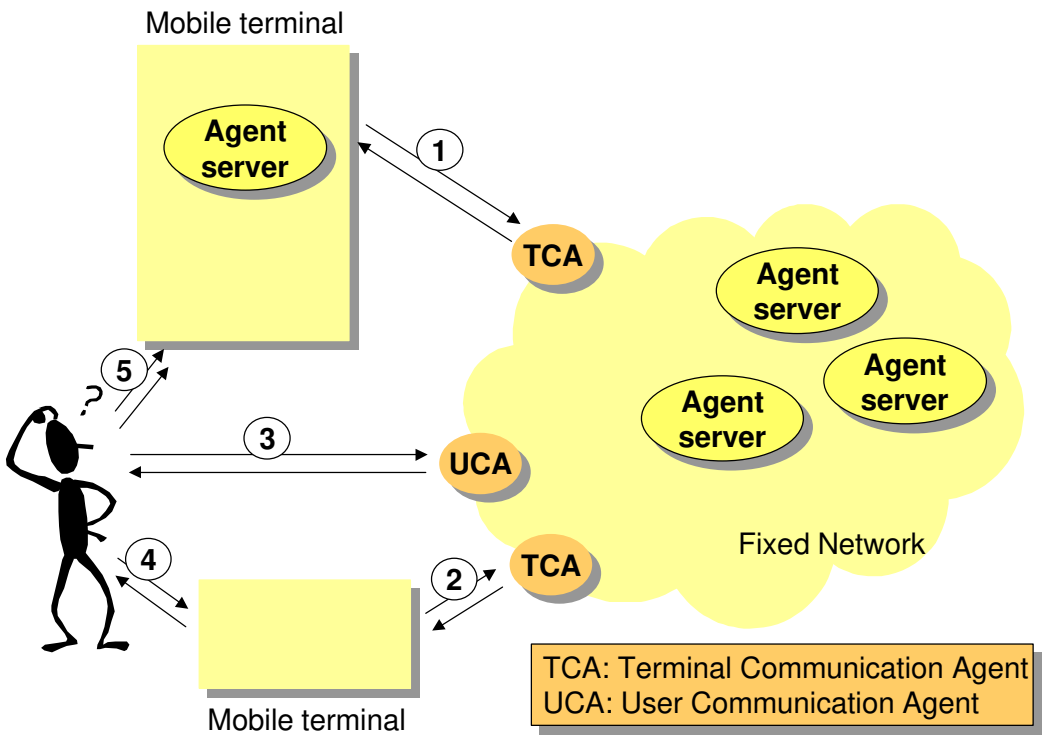

Figure 5: Agent Communication Scenarios in Wireless Environments

In Case 3 the mobile device of the user is a plain digital phone or a low-end PDA that only enables simple messaging. The user can control the agents in the fixed network by using GSM short messages (SMSs), for example. An agent can inform the user when it has completed a given task, has received interesting results, and so on. In this case the user communicates with a User Communication Agent (UCA) in the fixed network.

Case 4 and 5 involves human-agent interactions. These interactions are similar to any human-agent interaction. They are independent from the wireless environment. Instead, device characteristics—display quality, screen size, audio and video capabilities—heavily affects appropriate ways of carrying out the interactions.

Communication between agents can be divided into four layers:

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 interactions [14], belong to this layer.
2. ***The Communication Language Layer*** defines the content of messages exchanged between the parties. Examples of communication languages belonging to this layer include FIPA ACL [14] and KQML [15].
3. ***The Message Transport Layer*** deals with messaging protocols such as HTTP [16], Java RMI [17], GIOP [18], and higher level WAP [19] protocols.

4. ***Transport and Signaling Protocol Layer*** implements network transport mechanisms, such as TCP/IP, lower level WAP protocols, SMS, and MDCP [20].

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 [21])—addresses optimization of communication over wireless networks. In FIPA we are actively involved in TC-E that deals with nomadic application support and with bit-efficient ACL. In OMG [22] we participate in a proposal for wireless access and terminal mobility in CORBA [23]. In IETF we work in the PILC working group [24] that addresses 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 [5].

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 significantly improve the efficiency of link usage.

Optimization in the communication language layer includes compression and conversion between media types. For example, FIPA ACL messages are currently encoded as human readable ASCII strings. If tokenized binary encoding would be used, the wireless link could be much more efficiently used. 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 wireless link should be hidden from communicating agents so that mobile agents can be efficiently found 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 wireless link.

3.2 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.

The ***Agent Transfer Service*** allows agents to move between agent platforms. All agent platforms that carry the Monads services are able to transfer agents by using either their internal transfer service (through the underlying Agent platform) or the Monads transfer service. The Monads transfer service is optimized for wireless links, 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.

The ***User Interface Service*** takes care of all the communication between the Monads System objects and the user. When an agent wants to signal a message to the user or ask him for information, an instance of the service, if not existing, is started. Depending on the user preferences described in the User Profile and the capabilities of the terminal described in the Terminal Profile, a (possibly graphical) user interface is built.

The ***RMI Service*** provides for RMI communication between agents. Basically, an agent can get a normal Java object reference by providing the Global Unique Identifier (GUID) of the peer agent as a parameter. The object reference is to a proxy object, similar to CORBA client stubs. The agent can set attributes (such as priority) for communication to the obtained reference. Moreover, different attributes can be specified for request and reply (e.g. reply has lower priority than request). Our prototype implementation reduced the execution time of the "hello world"-program from 8.4. to 2.3 seconds over GSM Data Service [5].

The ***Data Transfer Service*** is designed to transfer larger chunks of data. Each separate data item forms a Data Transfer Object (DTO). A DTO can, for example, be a mail message, a web page or a file. A DTO can be created directly from a file, or it can be created empty and filled with data later. Once the DTO has arrived to its destination (even though all its data may not have), the "arrivedDTO" event is sent to the receiver, with the DTO reference as a parameter, allowing the receiver to process the DTO. DTOs are subject to a scheduling algorithm, which selects the next DTO to be transferred based on DTO priorities and whether the DTO is ready for transfer. A DTO can be parameterized to affect the way its transfer is scheduled, usually by setting its priority.

The ***Stream Service*** provides for byte streams with communication attributes such as priority. The byte streams are bi-directional, and in wireless environments they accept Quality of Service (QoS) requirements. Since they will be ordinary TCP connections in the fixed network, no QoS will be taken into account there. The Stream Service can be used to access Mowgli data channels, and all other communication services are built on top of it.

The ***Event Service*** allows an agent to register itself as a consumer or a supplier of an event. New event types can be added. Events sent to the Event Service by suppliers are forwarded to all registered consumers. The supplier may assign attributes to an event, similar to those used with RMI. The event supplier can limit the distribution of an event to certain agent groups, or exclude certain groups from the distribution. The limitations can also be attached to the event type instead of a single event. Similarly, the event consumer can refuse to receive events from certain groups, or only receive events from certain groups.

The purpose of the ***Quality of Service Management*** is to provide an interface to agents for retrieving network performance data, such as transmission rate, throughput, and round-trip time. In addition, it is used to inform agents about the status and changes of the network services, that is an agent may subscribe network-related events using Event Service. Event

Service employs QoS management to actually get these events, and delivers events to agents. Furthermore, agents may use QoS management to explicitly control network services. To name an example, an agent may establish or tear-down the network connection.

4. 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. The minimum requirement should be that a system should detect when current data transmission requests tasks may not be completed any longer in a reasonable amount of time due to temporary changes in the QoS.

A straightforward but often very useful reaction would simply be to pause or to cancel some of the transmissions. A reasonable alternative might be to inform the user about the new situation and to offer him or her a change to pause or to cancel some of the requests. 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 can not be fulfilled within a certain timelimit. A good example is a Web browser 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 immediately after a change in the QoS is detected comes too late. This is especially true when the connectivity was just lost—nothing can be done after detection of a dropout but something could have been done beforehand if the system would have been able to predict the dropout.

Predicting changes in the 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***: which tasks will be allowed to use bandwidth when the connection is about to be lost soon
- in ***data prefetching***: to download something beforehand while the connection is still good,
- in ***connection management***: to close the connection now in order to save expenses because the QoS will be execrable during the next 5 minutes.

Useful predictions can be achieved only by ***learning*** how some variables like time and location affect the QoS. Unfortunately, the QoS is also affected by several factors like weather that are very hard or impossible to observe and/or to predict through computing equipment. Therefore, no system shall ever be able to provide exact QoS predictions. However, we believe that predictions of useful accuracy can be made using quantities like *time of day*, *day of week*, *recent QoS values*, and *location of the terminal*.

In practice, the user does not have a connection open all the time, which divides the prediction problem into two cases: prediction without and with the information about recent QoS trends. Intuitively, the latter problem is easier to cope with and several approaches, proposed for other contexts, are available. One of them is *Example-Based Reasoning* (EBR). In its most simple form, the closest matching trend with the current trend (according to some distance metric) would be chosen to predict the future of the current one. This approach, however, can not use all additional information available from other trends. A slightly different approach is to model the phenomenon that creates the

observed behavior. Methods like *Hidden Markov Models* have been applied to situations like this when the time-series has many unknown factors.

In the Monads project we have divided the original problem into two subproblems: ***predicting terminal movement*** and ***predicting QoS at a given location and time***. By useing the Log Service, Perception History and Learning Service available in the Monads system, our Modeler Agents build models of terminal movements in time and space and of QoS-maps. Below we will briefly summarize these services.

The ***Log Service*** depends on other services and agents that provide measurements to be traced when the Log Service registers itself to obtain them. When a measurement is registered, its name and the manner of obtaining the measurement, either polled by Log Service or pushed by a service or agent, are stored. Measurements are time-stamped and can have some optional attributes, like type for categorizing them or an estimated cost for obtaining the measurement.

The ***Perception History*** is responsible for collecting and storing information that can be used for learning. For example, "perceptions" may be information about system events or the status of some continuous or discrete quantity like the battery level of a laptop computer or communication speed.

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. Usually learning tasks are traditional classification problems in which the output value depends only on the given set of input values. These problems can be handled with some well-known learning algorithms like C4.5 [25] or CN2 [26]. However, in many cases the output value also depends on previous values of input and output attributes. This kind of prediction tasks must be handled with some special algorithms which will be decided later.

Actual learning is done by ***Modeler Agents*** that actively try to build models, such as decision trees, for some learning task using data collected with the help of the Perception History or given by the client. 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. This approach is motivated by the good test results achieved in combining models built from multiple batches of data [27].

5. 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* (see Figure 1) according to the 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 the Mobile Messaging Services and Wireless Transport and Signalling Services 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.

In the Monads project we have already designed an agent architecture and specified a basic set of system services. Currently we are implementing a prototype in order to verify functionality and implementability of our ideas. Preliminary results are promising; using our RMI Service prototype we were able to reduce the execution time (elapsed time) of the "hello world"–program from 8.4 to 2.3 seconds over GSM Data Service [5], for example. We anticipate similar improvements also in the other communication services. In the next phase of the Monads project the focus will be on learning and predicting available Quality-of-Service during the next 1–30 minutes when both end-users and terminals are on the move.

6. ACKNOWLEDGEMENTS

The authors are grateful to their colleagues in the Monads project, in particular to Markku Tamski from Nokia Mobile Phones and to the Monads group at the University of Helsinki: Stefano Campadello, Heikki Helin, Oskari Koskimies, Pauli Misikangas, Mustafa Abdulla, Mikko Mäkelä, Sasu Tarkoma.

7. 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] 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.

[4] H. Helin, H. Laamanen, and K. Raatikainen, "Mobile Agent Communication in Wireless Networks," To appear in the *Proceedings of the European Wireless'99 Conference*, October 6–8, 1999, Munich, Germany.

[5] S. Campadello, H. Helin, O. Koskimies, and K. Raatikainen, "Optimizing Java 2 RMI for Slow Wireless Links," *Submitted for publication*.

[6] T. Alanko, M. Kojo, H. Laamanen, K. Raatikainen, and M. Tienari, "Mobile Computing Based on GSM: The Mowgli Approach," in *IFIP'96 World Conference - Mobile Communications*, Sep. 1996, pp. 151–158.

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

[8] 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.

[9] ObjectSpace, "Voyager Home Page," http://www.objectspace.com/voyager/.

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

[11] Object Management Group, *CORBA Security 1.5*, OMG Document ptc/99-02-01, 1999.

[12] Foundation for Intelligent Physical Agents, "FIPA Home Page," http://www.fipa.org/.

[13] EC/ACTS CLIMATE Cluster, "CLIMATE Home Page," http://fokus.gmd.de/ima/climate/.

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

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

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

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

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

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

[20] 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.

[21] The Department of Computer Science at the University of Helsinki, "Home Pages of Research Projects," http://www.cs.Helsinki.FI/research/.

[22] Object Management Group, "OMG Home Page," http://www.omg.org/.

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

[24] Internet Engineering Task Force, "PILC (Performance Implications of Link http://pilc.grc.nasa.gov/pilc.

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

[26] P. Clark and T. Niblett, "The CN2 induction algorithm," *Machine Learning*, Vol. 3, pp. 261–283, 1989.

[27] 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.