Intelligent Agents for Nomadic Users

MSc. Pauli Misikangas

Department of Computer Science, University of Helsinki

P.O. Box 26, FIN-00014 UNIVERSITY OF HELSINKI, Finland

E-mail: Pauli.Misikangas@cs.helsinki.fi

Abstract

Mobile computing equipment such as laptop computers, hand-held devices, and perhaps even wearable computers will soon be in everyday use by everyone, anywhere and anytime. Users are no longer tied to their offices and homes. They will carry a computer with them, and use it while on the move. This introduces a completely new concept to be taken into account by software developers — the location of the user. Supporting nomadic users raises great challenges; communication over a wireless connection must be optimized, and new kinds of applications should be developed. Because of the very dynamic wireless environment, applications must be capable of adapting to the circumstances and making decisions autonomously. Thus, use of intelligent agents to support nomadic users is a natural approach. In this paper, we tackle the problem of predicting the future location of a user, which is often the base for the intelligent behavior of nomadic application agents. We present a method for learning the regular routes of a user, and show how movement predictions can be used to predict the future quality of a wireless connection.

1 Introduction

The age of nomadic computing is coming. Mobile computing equipment such as laptop computers, hand-held devices, and perhaps even wearable computers will dominate computer markets in the future. They will be in everyday use by everyone, providing access to computing services anywhere and anytime. Users are no longer tied to their offices and homes.

They will carry computing and communication equipment with them, and use them while on the move. This introduces a completely new concept to be taken into account by software developers — the location of the user. People behave differently in different locations; their actions, needs, and interests strongly depend on their current or near future location. Users do different things with their computers at home than at work, they need a different kind of information while shopping than on the way to a business meeting, and so on.

When connected to a network, a mobile computer becomes a very powerful tool; it can be used to access all information on the Internet, perform tasks requiring heavy computation as remote processes, and to communicate with other users. When a cable connection is not available, mobile computers are connected to the network using some wireless connection, a mobile phone or wireless LAN, for example. Unfortunately, traditional network applications and protocols designed to be used with a fast and reliable connection do not work well enough with a slow and error-prone wireless link [1]. They cannot properly handle situations in which the quality of the data transmission varies all the time.

Again, the location of the user plays an important role. Every mobile phone user knows that at some locations the connection is always excellent and somewhere else it is always inferior. Even the whole computing environment may vary according to the location; in the office, the user may use a wired LAN network, elsewhere inside the building a wireless LAN, outdoors a GPRS/UMTS network, and at home a modem connection. Whatever the computing environment, whatever the quality of the connection, applications should be able to adapt themselves so that they will optimal for the user.

Solving the problems of wireless communication is not enough — we should also develop new kinds of services and applications in which the location of the user is taken into account. To name some examples, we could develop a group of services targeted to tourists in a strange city: Mobile computers could be used to find information about hotels, restaurants, shops, and exhibitions that are near the user or on his/her way. A hand-held PDA (Personal Digital Assistant) could tell the user which bus to take and when to get off, when given only the destination. A group of tourists could let their PDAs arrange a rendez-vous not too far away from anyone, so that they can meet each other after wondering around the city the whole day — when no one knows the exact location of the others.

As we can see, supporting nomadic users is a great challenge to software developers. Nomadic applications must be more intelligent than most applications of today. Because of the dynamic environment they are used in, applications must be capable of adapting to the circumstances and to the needs of a user. Due to the lack of resources, every operation performed must be considered carefully. For some applications, being passive and waiting for the user to make some requests, is not acceptable. Instead, they should be active, taking the initiative and doing their tasks autonomously. We could, for example, have an intelligent connection management system that makes autonomous decisions on when to open or close a connection, always selecting the best available media according to the user's preferences (e.g. fastest connection, lowest price). Moreover, nomadic applications should work as a team, taking each other into account, negotiating about future actions, and sharing knowledge. Without co-operation and coordination, autonomously working applications could easily end up fighting for available resources so that none of the applications gets its job done well. On the other hand, each application should also be able to operate autonomously, without presence of others. To say it with one word, we need *agents*.

We have already pointed out several application areas for intelligent agents supporting nomadic users. In all of them, intelligent behavior is based on knowledge about the user's location and movement. Thus, it all boils down to the questions: where is the user now, and where is he/she going? The first one is easy to answer if we have some kind of positioning device, for example GPS, attached to the terminal. Future terminals will very likely have a built-in positioning system, so we can safely assume that the geo-location of the user is always known. However, the latter question is a much more difficult one. Several applications need to know the future geo-location of the user, but we cannot assume that this information is always given by the user. Thus, the system must try to predict the user's movement.

In this paper, we will describe a method for learning the regular routes of a user and predicting terminal movement. Movement predictions can then be used by intelligent agents supporting nomadic users. As an example, we present an intelligent agent that predicts future Quality-of-Service of a wireless connection by using the knowledge about future location [2]. But first, we shall give an overview of the Monads project [3] in which this research is carried on.

2 Monads Overview

The research project Monads [3] examines adaptation agents for nomadic users. It is carried out by the Nomadic Computing Group at the Computer Science Department at the University of Helsinki, Nokia Mobile Phones, Sonera, and Nokia Telecommunications; funded also by the National Technology Agency of Finland (TEKES). The project started in February 1998 and is scheduled to run until December 2000.

Figure 1 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 [4] and Wireless LAN [5, 6]. In the future we will also support GPRS [7], perhaps also some WAP protocols [8], 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.

The goal of Monads is to help nomadic users and nomadic application developers in the following ways:

- Improve the efficiency of existing network applications such as 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 maintenance of wireless connections with regard to cost and response time. Often con-

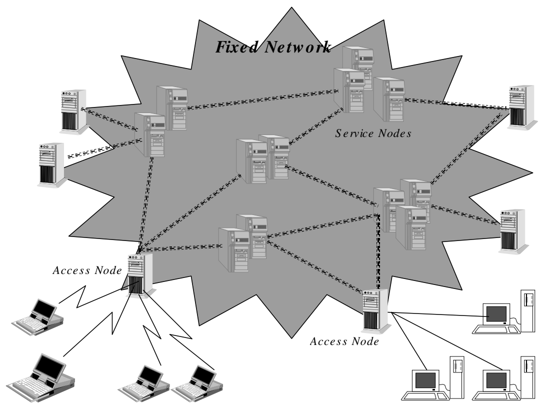

Figure 1: Monads System Reference Configuration

nections are quite expensive, and in those cases the system should carefully consider when the connection is really needed, and work off-line whenever possible.

The Monads system is based on the idea of adaptive, collaborative agents, as depicted in Figure 2. 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.

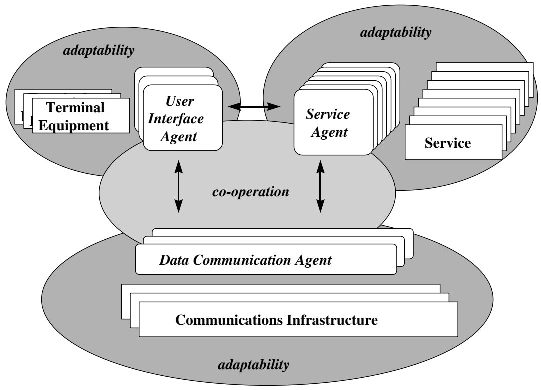

Figure 2: Monads Adaptation Model

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 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.

The Monads architecture, outlined in Figure 3, is based on agents and consists of seven layers. At the bottom we have the underlying network, ranging from slow wireless networks to high-speed fixed networks. On the Transport and Signaling layer, we use the Mowgli communications architecture [9] that takes care of data transmission issues in wireless environments, hiding most of the communication problems from upper layers. However, the message transport layer (Messaging Services) still needs to be mobile-aware and optimized for wireless environments. Therefore, we have made enhancements to existing solutions, such as OMG and FIPA specifications as well as Java RMI, to make them work better with wireless links [10, 11, 12].

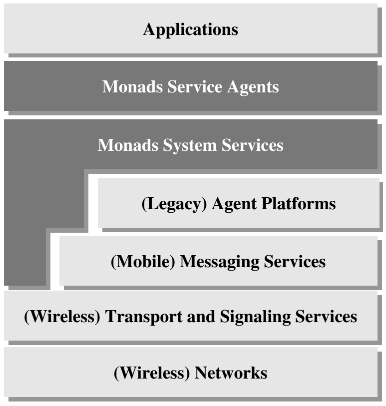

Figure 3: Monads Layered Architecture

On the uppermost layer there are standard, non-Monads applications such as web browsers or email clients. Monads Service Agents implement the adaptation needed by a particular legacy application, using the Monads System Services. In some cases, the whole application may be implemented as 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. All agents run on top of JADE [13] which is a Java-based FIPA-compliant agent platform.

3 Predicting Terminal Movement

In order to predict terminal movement, we must first try to learn the movement patterns of the user who carries that terminal. Luckily, most users do not move randomly with their terminals. Instead, they probably use only a few routes in their everyday life — from home to work, from work to home, and so on. Thus, we believe that the movement patterns of a user are relatively easy to learn. Of course, in order to learn anything about terminal movement, we must obtain the current location of the terminal somehow. Hence, from now on we assume that the terminal has some kind of positioning device (e.g. GPS) attached to it.

The concept we would like to learn — regular routes of the user — is hidden in a continuous stream of information about the location and speed of the terminal, received from the positioning device. Therefore, the sequence of coordinates must first be transformed into a much shorter sequence of important waypoints. Waypoints are added to locations where the direction of movement or speed changes significantly or routes cross. In the example of Figure 4, the original route drawn with a solid line is transformed into a waypoint sequence $\{A, B, C, D, E, F, C, G\}$.

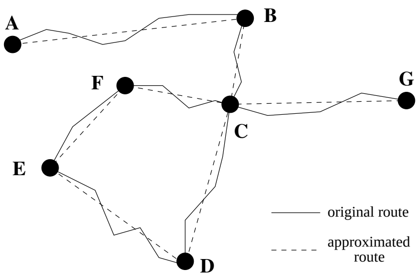

Figure 4: Approximating a route with a few waypoints

We say that a waypoint w_i has a connection to w_j if there has been a sequence of waypoints in which w_j came right after w_i. Thus, waypoints and connections between them form a directed graph that we call a waypoint map. We also define that $S(w_i)$ is the set of all nodes w_j to which w_i has a connection.

Now we can define that a movement pattern is a sequence of waypoints that appears frequently. The original problem of location prediction can also be mapped to a problem: given a list of previous waypoints, what will be the following waypoints? At first sight, this problem looks similar to discovering frequent episodes from a sequence of events, which is currently a hot topic in the knowledge discovery field (see e.g. [14]). Fortunately, the fact that each waypoint has only a few connections makes this problem a much easier one to solve. In addition, we do not need to use all existing waypoints in the movement prediction task. It is enough to take into account only waypoints from which there are at least two different ways to continue, and waypoints where the user has remained for a long time. Thus, the number of necessary waypoints is so low that we can use a simple and efficient data structure called backward tree [15] for learning and predicting the movement of a user.

Each node N of a backward tree consists of a waypoint $N.w$, a set of child nodes $N.c$, and a map $N.f$ that maps waypoints to frequencies defined as follows. Let N_0^k be the root node of a tree T_k (there is a separate tree for each waypoint), and N_i^k be a direct child of N_{i-1}^k. Hence, $N_0^k, N_1^k, \ldots, N_i^k$ is a direct path from the root to the node N_i^k. In addition, for all N_i^k holds that $N_i^k.w \in S(N_{i+1}^k.w)$. Now, $N_i^k.f(w_j)$ is the frequency of the waypoint sequence $\{N_i^k.w, N_{i-1}^k.w, \ldots, N_0^k.w, w_j\}$. Using the frequencies, we can easily calculate probabilities for every possible path following the given sequence of waypoints.

Figure 5 shows an example of a backward tree for waypoint B, when the route traveled so far is $\{A, B, C, G, C, B, A, B, C, D, E, F, C, B, C\}$. The circle on top of the figure is the root node N_0 having $N_0.w = B$. The rectangle beside it is the frequency map $N_0.f$, containing entries $(A, 1)$, and $(C, 3)$. Hence, the frequency of cases where waypoint C has followed waypoint B is $N_0.f(C) = 3$. The root node has two child nodes, so $N_0.c$ contains the nodes labeled with A and C. By looking at the frequency map of the child node on the right, we can see that in cases where the previous waypoints have been C and B (in that order), the next waypoint has once been A and once C.

4 Predicting Quality-of-Service

The characteristics of a wireless environment — long latencies, highly variable delays, and sudden disconnections — raise problems that are not met in wireline networks. Applications designed to be used with a fast and stable network connection may perform poorly when used with a wireless connection. The situation may be improved by using low-level communication methods and protocols optimized for wireless that hide most of the problems from applications, as done in the Mowgli system [9]. However, to really solve the problem, we must redesign the applications, make them more intelligent and capable of adapting to the changes in the

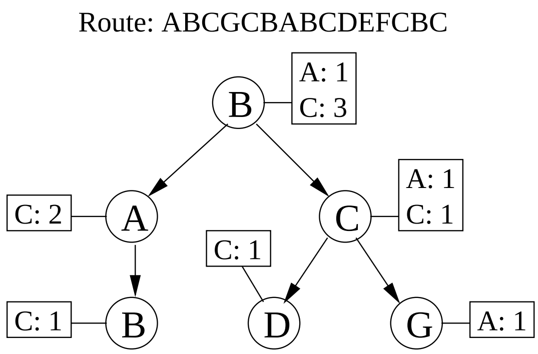

Figure 5: Backward tree for waypoint B.

Quality-of-Service (QoS).

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. A straightforward but often very useful reaction would simply be to pause or to cancel some of the transmissions, or offer the user a change to do this. 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 immediately 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*, as for example, which tasks are allowed to use bandwidth when the connection is about to be lost,

- *data prefetching*, as for example, download something beforehand while the connection is still good,

- *connection management*, as for example, close the connection now 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 such as time of day, day of week, past QoS values, and location of the terminal 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 by computing equipment. Therefore, no system shall ever be able to provide exact QoS predictions. However, we hope that the quantities listed above are enough to make predictions of useful accuracy.

We have implemented an intelligent agent providing QoS predictions into a prototype of the Monads project [3]. So far, our main focus has been on how the location of the terminal affects the QoS. Thus, QoS predictions are based on predictions about terminal movement, achieved using methods described in section 3. The components of the Monads System architecture that involve QoS prediction are shown in Figure 6. The components are:

Perception Service A centralized collection of perceptions. By perception we mean anything that can be observed, such as location and throughput. The Perception Service takes care of collecting values of perceptions within certain time intervals and storing the values.

Location Service Reads location information from a positioning device (e.g. GPS) and builds/updates a waypoint map according to the terminal movement. Information about the terminal location and waypoints reached are stored in the Perception Service.

QoS Management Provides information about the current QoS and stores it in the Perception Service.

Route Modeler Agent An autonomous agent that tries to learn the regular routes of the user, and provides predictions about future locations of the terminal. Collects data (sequences of reached waypoints) with the help of the Perception Service, and builds/updates the movement model as described in section 3.

QoS Modeler Agent An autonomous agent that tries to learn the QoS as a function of location and time, and provides estimates about future QoS at a given location and time. Collects data (time, location, QoS) with the help of the Perception Service, and builds/updates a route-based QoS model.

QoS Prediction Agent An intelligent agent that provides predictions about the future QoS by combining predictions made by the modeler agents described above. If there are alternative ways to predict movement and QoS, this agent is responsible for selecting between them. It may also warn other agents when the QoS is about to change significantly.

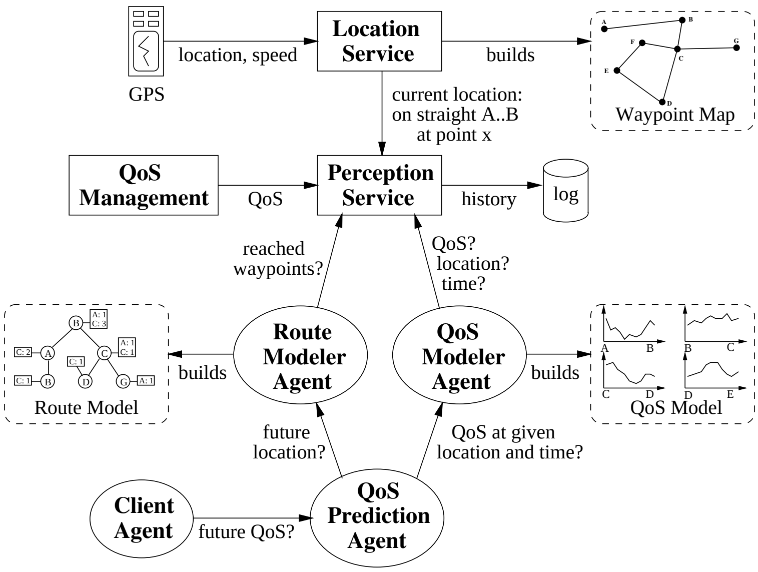

Figure 6: Monads system architecture

5 Conclusions

The age of nomadic computing is coming, which brings great challenges to software developers. People want to use small mobile computers having a wireless network connection to do the same things they are used to doing with desktop computers connected to a network via cable. In addition, a whole group of completely new kinds of services should be developed to support the needs of a nomadic user. We have argued that it is important to take the location of the user into account when developing nomadic applications; both the behavior of the user and the quality of the network connection depend on it. Besides knowing where the user is now, applications may also need to know the future location of the user, in order to prepare for forthcoming situations. With the methods presented in this paper, we can learn the regular routes of a user, and make movement predictions that can be used in various nomadic applications. As an example, we have presented an agent application for predicting the future quality of a wireless connection.

Acknowledgments

This work was carried out as a part of the Monads research project, funded by Sonera Ltd, Nokia Telecommunications, and the National Technology Agency of Finland. I wish to thank the rest of the Monads team, especially my supervisor Kimmo Raatikainen for his helpful comments.

References

[1] IETF, "Performance Implications of Link Characteristics (PILC)." http://www.ietf.org/html.charters/pilc-charter.html.

[2] P. Misikangas, M. Mäkelä, and K. Raatikainen, "Predicting QoS for Nomadic Applications Using Intelligent Agents," in *Proc. of the IMPACT'99 workshop*, 1999.

[3] S. Campadello, H. Helin, M. Misikangas, M. Mäkelä, O.Koskimies, and K. Raatikainen, "Using mobile and intelligent agents to support nomadic users," in *Proc. of the ICIN'2000 Conference*, 2000. Project homepage `http://www.cs.helsinki.fi/research/monads`.

[4] M. Mouly and M.-B. Pautet, *The GSM System for Mobile Communications.* Mouly and Pautet, 1992.

[5] M. Shafi, A. Hashimoto, M. Umehira, S. Ogose, and T. Murase, "Wireless Communications in the Twenty-First Century: A Perspective," *Proc. of the IEEE*, vol. 85, no. 10, pp. 1622–1638, 1997.

[6] K. Pahlavan, A. Zahedi, and P. Krisnamurthy, "Wideband Local Access: Wireless LAN and Wireless ATM," *IEEE Communications Magazine*, vol. 35, no. 11, pp. 34–40, 1997.

[7] G. Brasche and B. Walke, "Concepts, Services, and Protocols of the New GSM Phase 2+ General Packet Radio Service," *IEEE Communications Magazine*, vol. 35, no. 8, pp. 94–104, 1997.

[8] Wireless Application Protocol Forum, "WAP Specifications," 1998–9. http://www.wapforum.org/.

[9] 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, pp. 1337–1348, Sept. 1997.

[10] M. Liljeberg, K. Raatikainen, M. Evans, S. Furnell, N. Maumon, E. Veltkamp, B. Wind, and S. Trigila, "Using CORBA to Support Terminal Mobility," in *Proc. of TINA'97 Conference*, pp. 56–67, IEEE Computer Society Press, 1998.

[11] H. Helin, H. Laamanen, and K. Raatikainen, "Mobile Agent Communication in Wireless Networks," in *Proc. of the European Wireless'99 Conference*, 1999.

[12] S. Campadello, H. Helin, O. Koskimies, and K. Raatikainen, "Performance Enhancing Proxies for Java[2] RMI over Slow Wireless Links." Submitted for publication.

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

[14] H. Mannila and H. Toivonen, "Discovering generalized episodes using minimal occurrences," in *2nd International Conference on Knowledge Discovery and Data Mining (KDD'96)*, pp. 146–151, AAAI Press, 1996.

[15] T. Bell, J. Cleary, and I. Witten, *Text Compression.* Englewood Cliffs, N.J.: Prentice Hall, 1990.