Predicting QoS for Nomadic Applications Using Intelligent Agents

Pauli Misikangas, Mikko Mäkelä, and Kimmo Raatikainen

Department of Computer Science, P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 UNIVERSITY OF HELSINKI, Finland
E-mail: {Pauli.Misikangas, Mikko.Makela,Kimmo.Raatikainen}@cs.helsinki.fi
Fax: +358-9-7084 4441

Abstract

Communication environments involving wireless networks challenge the system software supporting applications that end-users are used to, but which are primarily developed for wireline networks. The characteristics of wireless networks are very different from those of wireline networks; long latencies, highly variable delays and sudden disconnections (or extremely long latencies) create problems that are not met in the wireline networks. It is widely recognized that these problems must be tackled on all levels of communications. Moreover, the characteristics of available Quality-of-Service for nomadic users vary both in time and in location. Therefore, it is insufficient to react to the current situation. In the research project Monads we have addressed short term predictability of available QoS so that applications can adjust their behavior to the forthcoming QoS. The results presented in this paper clearly indicate that intelligent agents can learn the key characteristics and their temp-spatial variation of available QoS quite quickly, so that reasonable predictions of available QoS in the near feature are possible.

1 Introduction

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 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 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 dropout, but something could have been done beforehand if the system would have been able to predict the dropout.

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), *data prefetching* (e.g. download something beforehand while the connection is still good), and *connection management* (e.g. 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 like *time of day*, *day of week*, *past QoS values*, and *location of the terminal* affect the QoS.

This paper is structured as follows. In section 2 we will describe the methods we have examined for predicting QoS in a prototype system of the Monads project [5]. Since the Monads project is targeted to nomadic users, our main focus will be on how the location of the terminal affects the QoS. The methods have been implemented into *intelligent* and *learning agents* that are described in section 3. In section 4, we will present some preliminary test results obtained from a simulation. Finally, we will make conclusions about the usefulness of our approach and outline future plans for our research.

2 Methods for QoS Prediction

We have divided the original problem of QoS prediction into two subproblems: *predicting terminal movement* (section 2.1) and *predicting QoS at given location* (section 2.2). In section 2.3 we will describe how these predictions can be combined to answer 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?', and 'what is the probability that we can successfully transfer x kilobytes of data within the next t seconds?'. The methods described in the following subsections assume that the underlying probability distribution remains static. In practice, distributions may vary according to time. We handle this by using separate models for different times.

2.1 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 1, the original route drawn with solid line is transformed into a waypoint sequence $\{A, B, C, D, E, F, C, G\}$. 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*.

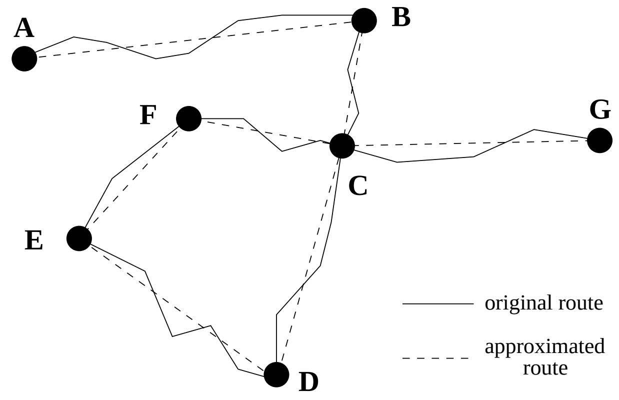

Figure 1: Approximating a route with a few waypoints

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. [3]). 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 is 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* [1] for learning and predicting the movement of a user. With a backward tree we can easily calculate probabilities for every possible path following the given sequence of waypoints.

2.2 Prediction of QoS at Given Location

In this section we concentrate on how to predict QoS in case the location is known. We will focus on prediction of *throughput*, but the same methods can also be used to predict other characteristics of QoS like latency and dropout probability. Our approach is based on routes which gives us a reasonable balance between efficiency and robust predictions.

If we are traveling along a route from where we have collected information about QoS sometime before, a large part of the prediction problem can be described as estimating how confident we are that the values we have observed are a representative sample of the underlying probability distribution. In case there are many attributes (e.g. time and location), we must also estimate, how much each attribute contributes to the observed values.

We have several options of how to save information about our observations on the route and how this information is used for predictions. For practical reasons we divide the route between two waypoints into parts where, in each part, the QoS is imagined to be the same from start to end. This is not a very restrictive assumption given that the parts are short enough. Perhaps the simplest way of information saving would be to calculate the mean for all QoS values over the time for each route-part. The weaknesses of this approach include its static nature (even very old observations are as important as the new ones) and that we need to somehow smoothen the predictions before our sample size is so large that the calculated mean approaches 'the true mean' (i.e. the value given by the underlying probability distribution). The first problem can be solved e.g. by using a cumulative weighted average instead of an average, the second by initializing route-parts with prior values, for example the recent global QoS values. A step forward from using averages, involving a little more calculations, is to use simple probability distributions. Utilizing Bayesian methods (see e.g. [2]) we can then use prior distributions for smoothing and make direct probability statements of QoS within the route-part in question.

So far, we have concentrated entirely on predicting QoS on routes the user has traveled before. To improve initial prediction accuracy, we can use some kind of *QoS-map* as a prior knowledge of the area's QoS properties. It may be obtained from a service provider or as a result of information sharing. There are many reasonable options to implement a QoS-map. In the case of areas with relatively static QoS, a point based map formed with *vector quantization* (see e.g. [4]) is a good candidate. That kind of map is efficient to transfer, calculations based on it do not require much processor-time, and further, it is possible to obtain an estimate of the prediction accuracy. However, for efficiency reasons it is wise to do the calculations for each route-part from the QoS-map only once, so that the map affects the prior distribution of the part.

2.3 Combining the Predictions

In previous subsections we have looked at ways to calculate probabilities for the user's future routes and QoS on them. Now we must combine the two prediction-sets in order to answer questions mentioned earlier.

The calculations are question dependent. However, for most questions concerning probability we must calculate the sum distribution of the QoS aspect in question. For each possible route, we multiply the sum distribution of the involved route-parts with the probability of the route and add to the sum over all possible routes. Now, to answer for example the question 'what is the probability that we can successfully transfer X kilobytes of data in Y minutes?', we calculate the portion of the (throughput) sum distribution greater than X. If we used averages for route-parts, we get only the mean of expected QoS and must use a separate model to estimate the reliability of the prediction.

3 System Architecture

We have implemented the methods described in the previous section into a prototype of the Monads project. The goal of Monads is to advance working in a wireless environment (e.g. a laptop computer connected to a fixed network via GPRS) by using intelligent and adaptive agents. The components of the Monads System architecture that involve QoS prediction are shown in Figure 2. The components are:

Perception Service A centralized collection of *perceptions*. By perception we mean anything that can be observed, for example 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 2.1.

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 the QoS model as described in section 2.2.

QoS Prediction Agent An intelligent agent that provides predictions about the future QoS by combining predictions made by the modeler agents described above (see section 2.3). 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.

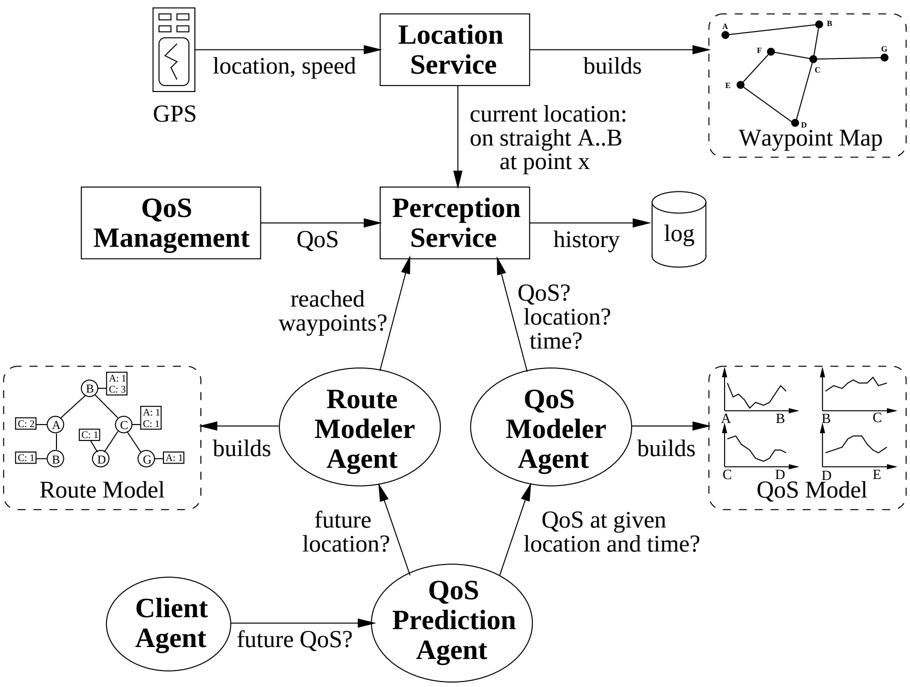

Figure 2: Monads system architecture

4 Test Results

Our QoS prediction approach is based on an assumption that the expected throughput varies according to location and time. This is not quite true for a GSM data connection,[1] but the forthcoming GPRS technology should have this property. In order to test the usefulness of our methods before GPRS is available, we have built some simulation capabilities into the Monads system. Because of the modular design of our system, it was enough to replace the real *Location Service* and *QoS Management Service* with simulated ones called *Route Simulator* and *Throughput Simulator*. The Route Simulator simulates the movement of a user by using a set of pre-defined waypoints and routes read from a file. The Throughput Simulator produces simulated throughput values for each location visited by using a 'QoS-map' read from a file. Both simulators include randomness, so that each simulation run is different.

'What is the expected amount of data that we can transfer within the next t units of simulation time?' This was one of the questions we asked our QoS Prediction Service. For each prediction, we also checked what would have been the correct answer by collecting throughput values for the period in question. We compared two implementations of this service. The first version, called *Nomadic QoS Prediction*, uses the methods we have described in this paper, i.e. its output is based on predictions of the user movement. The second version bases its predictions only on the average of recent throughput values. We call this latter version *Static QoS Prediction* because it assumes that the throughput remains at the same level as before.

Figure 3 shows a typical example of results from a simulation run. The solid line is the correct answer, i.e. the amount of data that actually could have been transferred within the given time limit, and the two other lines are the predictions made by Nomadic/Static QoS Prediction. As expected, the static version performs quite poorly, because it cannot foresee changes in QoS. So, if the throughput level changes significantly immediately after a prediction, the prediction may be totally wrong. In the beginning, the Nomadic QoS Prediction behaves just as badly as the static one because the 'user' is moving via unseen routes.[2] However, when on a route traveled before, the predictions are very close to the correct answer. Figure 4 shows the average error made by the methods over several simulation runs with different configurations.

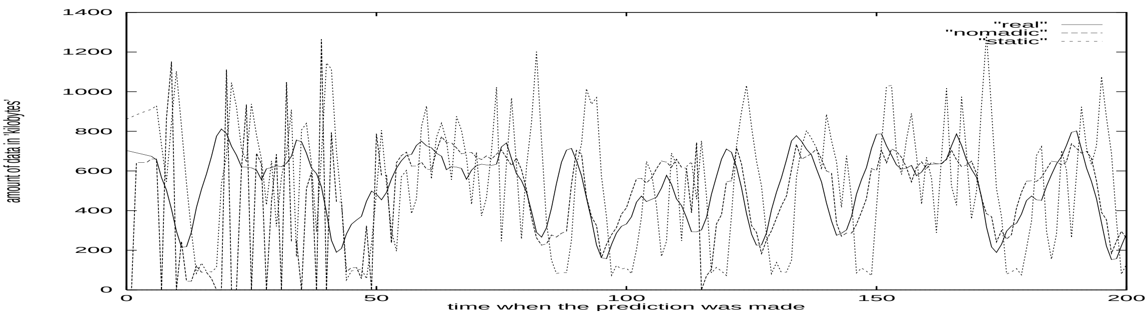

Figure 3: Nomadic vs. Static QoS Prediction

[1] In our field tests, most of the time throughput stayed at a very constant level and dropped only for short moments while changing the cell. The only useful thing that we could learn about GSM data connections is the probability of dropouts in different areas, but we have not yet paid much attention to this issue.

[2] No prior knowledge about the QoS-map or user routes was used in the simulation.

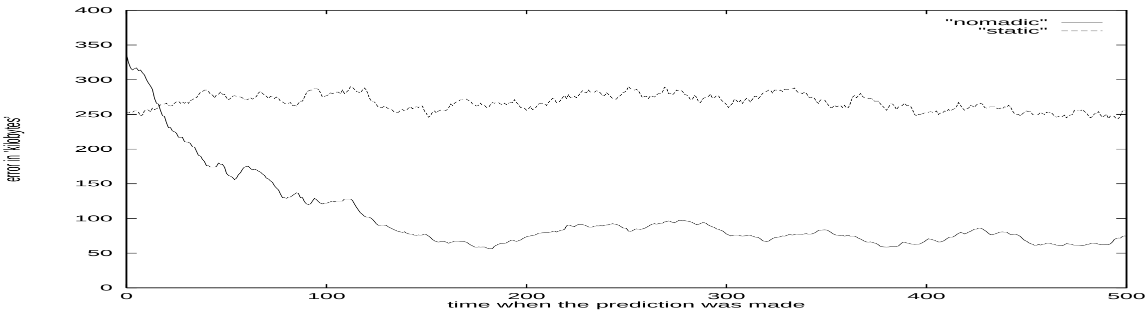

Figure 4: Average error

5 Conclusions

We have introduced a way of predicting available Quality-of-Service in wireless networks using intelligent agents that learn movement patterns of a user and the characteristics of available throughput in space. In the near future we will also consider other QoS aspects than throughput. The reported results clearly indicate that agents can quite quickly learn characteristics that enable reasonable short-term predictions.

The flexibility of the Monads agent system allows introduction of various kinds of modeler agents, each of which concentrates on learning a specific aspect of user or network behavior. These agents utilize the Monads system services and provide predictions to agents that combine those predictions in order to do predictions of higher levels of abstractions. In the future we will develop additional modeler agents. We will also examine alternative learning algorithms and pay special attention to combining predictions.

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. The authors express their thanks to the rest of the Monads team.

References

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

[2] A. Gelman, J. Carlin, H. Stern, and D. Rubin. *Bayesian Data Analysis.* Chapman & Hall, 1995.

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

[4] N. Nasrabadi and R. King. Image coding using vector quantization: A review. *IEEE Transactions on Communications*, 36(8):957–971, 1988.

[5] K. Raatikainen, L. Hippeläinen, H. Laamanen, and M. Turunen. Monads — Adaptation agents for nomadic users. In *Proc. of the Infrastructure Forum in Telecom 99*, Geneva, Switzerland, October 1999. ITU.