A Context-Aware Hoarding Mechanism for Location-Dependent Information Systems

Abstract

When used in an outdoor environment mobile information systems often suffer from the disadvantages of wireless WANs, especially low bandwidth, high delay, and frequent disconnections. Hoarding is an effective method to overcome these disadvantages by transferring information which is probably needed by the user in advance.

In this paper we propose a generic, context-aware hoarding mechanism. When selecting the information to hoard, it considers the user's future location as well as the expected speed of movement. In contrast to existing hoarding mechanisms it is universally applicable for different types of location-dependent, mobile information systems. Its flexibility allows it to rely on different knowledge sources in order to get information about a user's context.

Keywords: Wireless Communication, Mobile Information Access, Context-Awareness, Hoarding, Info-Station

1 Introduction

With the increasing pervasiveness of mobile computing devices the need for mobile information access grows continuously. A large variety of mobile information systems already exists, e.g. map/navigation systems (Ye et al., 1998) or mobile tourist guides (Davies et al., 1998), (Abowd et al., 1997). Many of these systems are location-aware, i.e. they take into account the user's location when presenting information. The location information, which is gathered by a sensor such as a GPS sensor or an Active Badge (Want et al., 1992) is often used to determine which information is probably of interest for the user. Since the same basic functionalities are useful in almost all location-aware information systems, we currently develop an infrastructure for such systems at our department, which provides these functionalities (Hohl et al., 1999). One of these functionalities is the context-aware hoarding mechanism described in this paper.

Hoarding is an efficient method to overcome the drawbacks of wireless WANs, especially low bandwidth, high delay, and frequent disconnections. The idea is to transfer information, which is probably needed by the user in the near future, in advance, so that it is already stored on the user's mobile device when it is actually accessed. It is even possible to access hoarded information in areas where no network is available at all, e.g. within a building or a tunnel. The problem with hoarding is to predict, which information the user will need. In our mechanism the decision about what information to hoard (hoarding decision) is based on knowledge about a user's future location and speed of movement.

The prediction is necessary since in most cases it is not possible to transfer all the information available in an information system to the user's device, due to storage space limitations or restrictions in the time the user is willing to wait for the hoarding process. Since the prediction can not always be correct, some information items the user requests may not be hoarded on the mobile device. To rate hoarding algorithms usually the hit-ratio is used. It states the part of a user's information requests that can be answered with hoarded information. An efficient hoarding algorithm is valuable in two ways: Firstly, if a given hit-ratio has to be achieved, it minimizes the time, bandwidth and storage space required for the hoarding. Secondly, if there are restrictions in the time, bandwidth or storage space available for the hoarding, it maximizes the achievable hit-ratio.

As communication infrastructure we use the info-station concept proposed in (Badrinath et al., 1996). There, a number of high-bandwidth wireless local LANs with a low range, so called info-stations, are placed in an area otherwise only covered by a wireless WAN (see Figure 1).
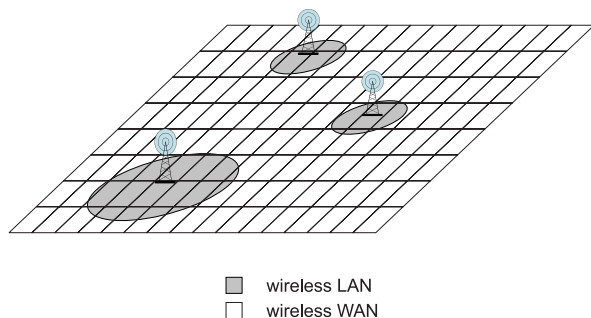

Figure 1: Info-station infrastructure.

When a user arrives at an info-station our mechanism aims to hoard as much as possible of the information he/she will probably need before reaching the subsequent info-station while meeting the restrictions in transfer time and storage space. If the user is not in an area covered by an info-station, no hoarding is performed at all. So the users only suffer from the disadvantages of wireless WAN connections, when they need information that was not hoarded at an info-station. Ideally, they do not need the wireless WAN at all. In systems where there is no crucial information, i.e. information that has always to be available, when a user accesses it, we can even rely exclusively on the info-stations and can completely renounce to wireless WAN technology.

Because LAN technology will always be cheaper and faster than WAN technology, the advantages of our info-station based hoarding mechanism will hold for the future, although new wireless WAN technologies will provide higher bandwidths.

In contrast to existing approaches our mechanism is designed as a platform mechanism, i.e. it has to support all the different types of information systems running on this platform. These systems may differ strongly in the degree of knowledge available about the user's future behavior. For example, a system navigating a user to a known destination can usually provide much more information about the user's future movement than a tourist guide system supporting a roaming tourist can do. Since the hoarding decision is based on the knowledge about a user's future move-

ment, the mechanism has to adapt to different degrees of knowledge in order to exploit the available knowledge always as far as possible.

The remainder of this paper is structured as follows: In Section 2 we make some preliminary definitions and state the design requirements of our hoarding mechanism. Afterwards, in Section 3, the mechanism is explained in detail. Section 4 provides an analysis of the mechanism's efficiency. Before concluding the paper in Section 6, we reflect the related work in Section 5.

2 Preliminaries

In the following, we introduce the system model that we use in the remainder of the paper, and explain which requirements our mechanism has to fulfill.

2.1 System Model

We consider an *information system* as a system which provides access to a set of discrete information items. It consists of a stationary server component on which the information items are stored and one or more client applications which run on the users' mobile devices and through which the users access the information stored on the server component.

Our mechanism is focused on *location-dependent* information systems. This means that it depends on the users' geographic position which information they are interested in. An example for a strongly location-dependent information system is a map application that always presents a map of the area surrounding its user.

The area supported by the hoarding mechanism is logically separated into equally sized *squares*. Although, other shapes would be possible as well, we chose squares as a canonical way of logically separating an area. Every info-station supports a fixed area, for which information can be downloaded at the info-station. This area is called the info-station's *hoarding area*. In reasonable installations the hoarding area should be bigger than the area covered by the info-station's LAN itself and smaller than the whole area supported by the hoarding mechanism.

2.2 Requirements

As we already pointed out the overall objective of our hoarding mechanism is to efficiently support different

types of location-dependent information systems. In the following, we describe the basic requirements the mechanism has to meet in order to achieve this objective.

- *Consider the users' context:* In location-dependent information systems the user's future location is obviously the most important hint on the information items he/she will probably need. Therefore, the hoarding mechanism should use visit probabilities for the decision about what items to hoard. This means, it should consider the probabilities with which a user will visit each square of the hoarding area after leaving the info-station.

  A further important aspect, which has to be taken into account, is the user's expected speed of movement. The reason for this is that a user moving through an area at high speed will probably need less information about this area than a user crossing it at a low speed. In (Ye et al., 1998) the benefits of considering the users' movement speeds for the hoarding decision have been shown.

- *Exploit different knowledge types:* We want our hoarding mechanism to use two different types of knowledge. The first one is knowledge which the mechanism has gained itself by observation of the users. We call this kind of knowledge internal knowledge. Due to privacy reasons the internal knowledge has to be stored anonymously. An example for internal knowledge are rankings of the most requested information items within a certain square.

  The second type, which we call external knowledge, is knowledge that is provided by the user or the client application of the information system. A navigation application, for example, can usually provide exact information about its user's future location.

- *Adaptivity:* Because of the large heterogeneity of information systems, we want to support, the hoarding mechanism has to be highly adaptive. A first aspect of this adaptivity is that it must be able to distinguish the different interests of the users. If an information system is accessed through various applications the information accessed from the users of the different applications will differ. But even the interests of users of the

same application may differ strongly. For example, a business man using a mobile city guide usually has other interests than a tourist using the same system.

The mechanism also has to adapt to different degrees of knowledge about the users' future movement, in order to exploit the available knowledge always as far as possible. This degree of knowledge depends on many factors, e.g. whether the user's destination is known or not, or whether the movement is bound to a given infrastructure, e.g. rails or roads, or not.

3 Hoarding Mechanism

In this section we explain the general idea of our mechanism together with the basic problems we had to solve when realizing this idea. We show how different types of knowledge can be gathered and used for the solution of these problems. We also present the components of a prototype architecture implementing the mechanism and give a detailed description of the overall functionality.

3.1 Basic Idea

Basically, our mechanism works as follows: Every time a user reaches an info-station a hoarding process is initiated. During this process the most popular information items belonging to those squares of the info-station's hoarding area that the user will probably visit before reaching the next info-station are hoarded. In order to determine which the most popular items are, the requirements mentioned above demand to consider different categories of users and applications. To realize this idea three problems have to be solved:

- *Correlation Problem:* The Correlation Problem is to find out which information items actually belong to each square, i.e. to relate information items to squares. The objective is to know for each square which information items the users located at that square need most frequently. Sometimes this is implicitly clear. For example, in a map application displaying a map of the user's environment it is obvious which parts of the map belong to each square. If this is not the case, the mapping of information items to squares has

to be explicitly specified by the information system's administrator through manual configuration or it has to be gained by observation of the users' behavior, e.g. by counting the number of requests of every information item originating from users located at a certain square.

- *Prediction Problem:* As we only want to hoard information items belonging to squares the user will probably visit, we have to make a prediction on what squares this will be. One possible solution to this problem would be to use existing prediction algorithms, such as described in (Chim et al., 1998), (Liu and Maguire Jr., 1995), which often base their prediction on the users' previous movement or movement patterns. However, a prediction based on a user's previous movement fails if the movement changes unexpectedly. Therefore, we do not rely exclusively on this kind of algorithms. We only use them as one possible source of knowledge. Further knowledge sources are the applications themselves, e.g. a navigation application might be able to specify the whole path along which a user will move. Finally, observation of the users can also help to solve the prediction problem. By counting the number of users visiting a certain square, we can determine the probability with which a user visits this square.

  As mentioned in Section 2 the expected speed is an important hint on how much information should be hoarded for a certain square. To predict a user's speed, we consider observation as the most important source of knowledge. From an observation of the users' speed within a certain square, we can derive the average speed at which the users cross this square. As with the prediction of visit probabilities we can imagine the applications as external knowledge sources.

- *Selection Problem:* Based on the predicted user behavior and the correlation knowledge, a decision has to be made on what information items should finally be hoarded. Such a selection is necessary, since in most cases not all information items correlated to probably visited squares can be hoarded, due to the restrictions mentioned above.

In a generalized form these three problems appear in every context-aware hoarding system.

3.2 Internal Knowledge

In our mechanism we use so called hoarding profiles to manage the internal knowledge, which is gained through observation of the users or manual configuration. For every application that accesses a certain information system exists at least one specific hoarding profile, which reflects the behavior of all the application's users. If an application is used by different categories of users a separate hoarding profile for every category can be maintained, which then reflects the specific behavior of the according user category. This is especially useful, if the users differ in their information needs and movement behavior, i.e. the places they visit and the speed at which they cross certain areas. Every specific hoarding profile can recursively be refined to further more specific sub-profiles. Finally, we get a hierarchy of hoarding profiles, where every profile reflects the behavior of all users belonging to one of its sub-profiles or to the category represented by the profile itself. In Figure 2 a part of such a hierarchy of profiles for a mobile tourist guide is depicted. To which profile a user belongs has to be specified by the application or the user himself/herself. For simplicity, we assume that every user is assigned to exactly one profile. Technically, it would be no problem to assign a user to more than one profile and to take into account all these profiles when selecting the information to hoard.
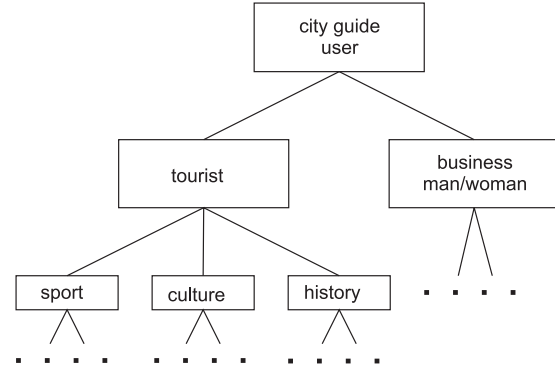

Figure 2: A part of a hoarding profile hierarchy.

Every hoarding profile consists of the following three components:

- *correlation table:* This table stores the internal knowledge about relationships between squares and information items. For every square in the hoarding area it holds a list of information items

4

that users located at the square access most frequently. We get this list by counting the number of requests to every information item. To adapt the list to changes in the users' access behavior the request counters are decreased periodically. We can also use more sophisticated methods to determine the users' access patterns within a certain square, e.g. the methods used in broadcast dissemination systems, such as described in (Hu et al., 1999). Sometimes, e.g. with the mentioned map application, the determination of the access patterns is not necessary, since it is obvious which are the most popular information items for a certain square.

- *visit map:* In the visit map the knowledge about the probabilities with which the squares are visited is kept. Formally, the visit map is a function $v : S \mapsto [0, 1]$, where S is the set of squares located in the hoarding area. v assigns to every square i in the hoarding area the probability with which the square is visited. This time the knowledge is gathered by counting the number of users of the according category visiting a certain square. Again, the counters are decreased periodically to adapt the knowledge to changes in the users' behavior. In this way we get the visit frequency of every square. From these frequencies the probability with which each square is visited can be derived. Figure 3 shows a graphical representation of such a map, which we might get if we consider the visit probabilities of a part of a city's streets. The brighter a square is the higher its visit probability.
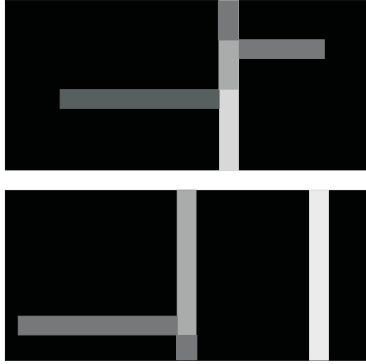

Figure 3: Visit probabilities of different roads.

- *speed map:* Analogous to the visit map the speed

map indicates the average speed of the users in every square. The speed map is a function $sp : S \mapsto \mathbb{R}$ that assigns to every square i in the hoarding area the average speed at which the users cross the square. The information is again gathered by observation of the users. To be aware of changes in the users' behavior only observations within a specific time window are considered.

3.3 External Knowledge

If available external knowledge, i.e. knowledge provided by the users and the applications themselves, is primarily used for the hoarding decision. We prefer external knowledge, since it is specified for one single user instead of a user category or all users of an application as it is the case with the internal knowledge. Therefore, the external knowledge will mostly be more specific and accurate than the internal knowledge. The users are not directly asked for external knowledge, since we consider it more suitable, if the application interacts with them in order to get the information they can offer.

To specify the external knowledge the applications also use visit and speed maps. In contrast to the maps representing the internal knowledge, these maps do not have to specify the visit probability respectively the average speed for every square in the hoarding area. We also allow that an application specifies more than one visit or speed map, since it might have knowledge about separate sub-areas of the hoarding area. Formally, the applications specify their external knowledge in a number of visit maps $v_1, \ldots, v_n$ and a number of speed maps $sp_1, \ldots, sp_m$. A visit map v_i is a function $v_i : S_i \mapsto [0, 1]$, where S_i is a subset of the squares located in the hoarding area. We assume that $S_i \cap S_j = \emptyset$, if $i \neq j$. Analogously a speed map sp_i is a function $sp_i : S_i \mapsto \mathbb{R}$.

To avoid that the users or higher application levels have to specify the visit and speed maps square by square, we propose so called hoarding patterns to ease this specification process. Hoarding patterns are constructs which are transferred into maps by lower application levels or the hoarding mechanism itself. Each hoarding pattern is specified through four components:
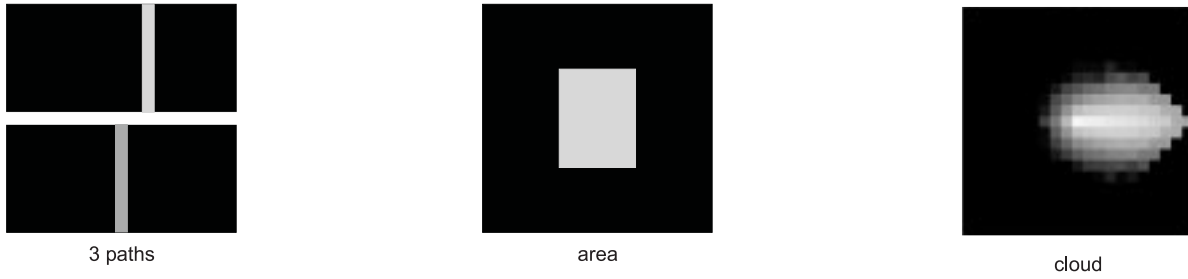
- its name

- a set of parameters

Figure 4: Visit probability maps derived from various hoarding patterns.

- the subset S_i of squares within the hoarding area, for which the resulting map specifies the visit probabilities or average speed.

- the visit probability or expected speed for the area covered by the hoarding pattern

The number of squares specified by the resulting map can be bigger than the number of squares actually covered by the pattern. In this case the visit probability and/or the expected speed of all uncovered squares is set to 0.

Some generic patterns, which are useful in many applications, are directly supported by the hoarding mechanism, i.e. it transfers them itself into visit or speed maps. However, there will also be application-specific patterns which have to be transformed into maps by lower application levels. In some cases it might not be possible to describe the maps with any pattern at all. Then the user or the higher application levels have to specify the maps directly. Examples of generic patterns, which should be supported directly by the mechanism are the following ones:

- *path:* A path is the shortest connection between two squares. The set of parameters belonging to this pattern exists of the start and end point of the path. This pattern is well suited to specify knowledge about a user moving along a road or walkway.

- *area:* We also allow to specify the visit probability or the expected speed for a complete area, e.g. a rectangle or a circle. This hoarding pattern is, for example, especially useful to describe the visit probabilities of buildings. The required parameters have to describe the geometry of the covered area.

- *cloud:* The cloud pattern can only be used to specify visit probabilities. With this pattern the application has to specify the probabilities with which its user moves to the north, east, south and west. Furthermore, the starting point of the movement and the probability with which the user reaches this starting point are required. The application also has to specify the number of squares the user will visit while moving according to this pattern. The motivation for this pattern is that applications can easily determine by observation the probabilities with which a user moves to a certain direction, even if they do not have any other hint on the user's future movement.

Figure 4 illustrates maps resulting from the specification of different hoarding patterns. The width and height of the specified maps is always 20 squares. With the cloud pattern we chose a probability of $\frac{3}{4}$ for movements to the east and $\frac{1}{12}$ for all other directions. The number of visited squares was set to 10.

3.4 Architecture and Algorithm

In the following we introduce a prototype architecture and the algorithm that realizes our mechanism. An overview of the architecture is given in Figure 5. It consists of two components: the mobile component, which runs on every mobile device and the stationary component, which runs on a stationary system, e.g. the information system server.

The *observer* gathers the internal knowledge about the users' movement behavior and preferred information items. It is responsible for maintaining all the hoarding profiles, which are stored in the internal knowledge database. If the hoarding profiles are maintained manually the observer component can be omitted.

If a user with his/her mobile device arrives at an info-station the *coordinator* initiates the hoarding process. Firstly, it requests the mobile component for
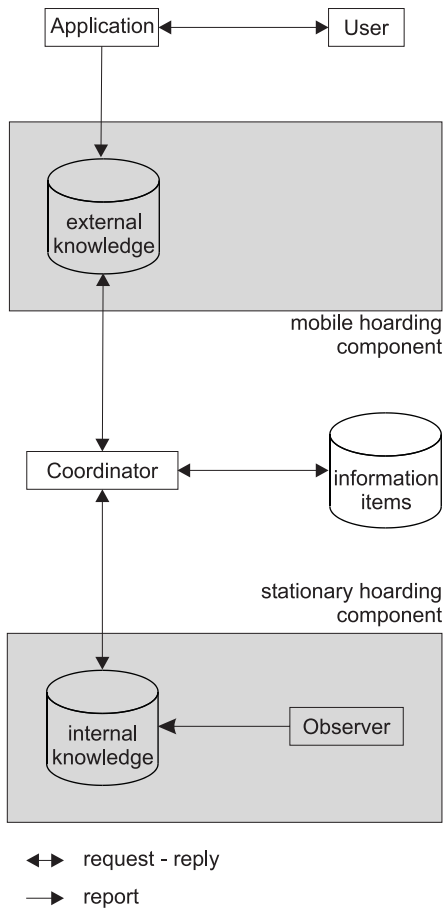
Figure 5: Prototype architecture.

volved in the assembly of the final visit map. In the same way the available speed maps are combined to a final speed map $e : S \mapsto \mathbb{R}$:

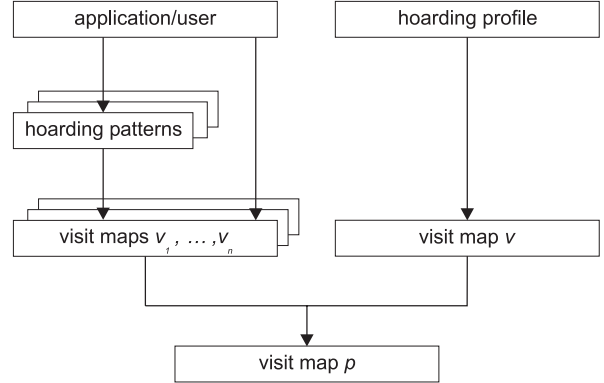$$e(i) = \begin{cases} sp_j(i), & i \in S_j \\ sp(i), & i \notin \bigcup_k S_k \end{cases}$$

Figure 6: Assembly of the visit map p

With the final maps the coordinator calculates a hoarding score $s(i)$ for every square i in the hoarding area. Therefore, it firstly determines for every square i the relative visit probability $\overline{p}(i)$ and the relative expected speed $\overline{e}(i)$:

$$\overline{p}(i) = \frac{p(i)}{p_{max}} \quad \text{and} \quad \overline{e}(i) = \frac{e(i)}{e_{max}},$$

where $p_{max} = \max_{i \in S} p(i)$, and $e_{max} = \max_{i \in S} e(i)$. Now the hoarding score $s(i)$ can be calculated according to the following formula

$$s(i) = \frac{\overline{p}(i) + c \cdot \overline{e}(i)}{c + 1}, \tag{1}$$

where c is a constant, which allows to control the influence the expected speed has on a square's hoarding score. We suppose that in most cases the visit probability is more important than the expected speed. In these cases c has to be smaller than 1. Next the relative hoarding score $\overline{s}(i)$ for every square i is determined:

$$\overline{s}(i) = \frac{s(i)}{\sum_{\forall j} s(j)}, \tag{2}$$

Afterwards, the coordinator calculates the maximum amount of data T that can be transferred without exceeding any of the storage space or transfer time

external knowledge. From this component it receives, if available, various maps and hoarding patterns describing visit probabilities and expected speeds. If some hoarding patterns have been received they have to be transformed into the according maps. Afterwards, the coordinator requests the visit and speed map of the hoarding profile belonging to the user's category from the stationary component. Finally, the coordinator ends up with a number of external visit maps $v_1, \ldots, v_n$, a number of external speed maps $sp_1, \ldots, sp_m$, the visit map v of the hoarding profile, and the speed map sp of the hoarding profile. All visit maps are combined to one final visit map $p : S \mapsto [0, 1]$ according to the following definition:

$$p(i) = \begin{cases} v_j(i), & i \in S_j \\ v(i), & i \notin \bigcup_k S_k \end{cases}$$

Figure 6 gives an overview of the different maps in-

	City Guide	Map Application
primary knowledge source	internal	external (user's destination)
correlation	observation initial: manual configuration	implicit
profiles	> 1	1

Table 1: Characteristics of different instantiations of the hoarding mechanism.

constraints. If there are no such constraints, T is set to a default value. Now the amount of data $t(i)$ to be hoarded for square i can be calculated as follows:

$$t(i) = \overline{s}(i) \cdot T \qquad (3)$$

Finally, the coordinator initiates for every square i the transfer of the top ranked information items related to the square. The information transfer from the information system to the hoard memory for square i is stopped when the amount of data transferred for square i reaches $t(i)$.

We have not assigned the coordinator statically to the mobile or the stationary component, since it depends on the available knowledge which is the better place. If there's more external knowledge than internal it will be better if the coordinator belongs to the mobile component with respect to the communication costs. However, if more internal knowledge is used, the stationary component will be the better place for it.

3.5 Examples

To illustrate the adaptivity of our mechanism we summarized the characteristics of two possible instantiations of it in Table 1. The first instantiation is designed for a mobile city guide and the second one for the map application mentioned in Subsection 3.1.

For the mobile city guide the primary sources of knowledge used to predict the users' future movement are the visit and speed maps of the hoarding profiles. External knowledge will not often be available, since we expect that tourists roaming around in a city can mostly not give any hints on their movement destinations. With the map application things are different. Here, the application is used to navigate a user to a specified destination. So the mechanism can rely on external knowledge in order to predict the future movement.

The two instantiations also differ in their correlation tables. With the tourist guide application the correlation information is gathered by observing the users' access patterns. Initially, a manually configured correlation table has to be used, until enough users have been observed to make reasonable statements on the access patterns. In the map application it is implicitly clear, which information items belong to each square.

Since all users access the same map, we do not have to maintain specific profiles for different user categories of the map application. However, when supporting the mobile tourist guide, different profiles might be useful to represent the interests of different user categories.

4 Analysis

We now analyze the benefit we get from considering visit probabilities and expected speeds. Therefore, we compare our hoarding mechanism to a simple one, which does not use visit or speed profiles. However, this simple algorithm also knows which information items are related to each square.

The simple algorithm works as follows: If there are n squares in the hoarding area and a total of m information items can be hoarded, the simple algorithm transfers $\lfloor \frac{m}{n} \rfloor$ items for each square. Afterwards, it randomly chooses $m - \lfloor \frac{m}{n} \rfloor \cdot n$ squares for which it will transfer one further item. Consequently, with this simple mechanism the number of information items hoarded for every square in the hoarding area is independent of its visit probability or the expected speed in it.

4.1 Metric

The metric we use to compare the hoarding mechanisms is the average hit-ratio a user experiences

when crossing a hoarding area. To determine this hit-ratio we first determine for every square in the hoarding area the average hit-ratio the users achieve while staying in this square. Afterwards, we calculate a weighted average of all these local average hit-ratios, in order to get the overall average hit-ratio for the whole hoarding area. The weights of the local hit-ratios have to be chosen accordingly to the probability with which a user visits the corresponding square and the average number of information items requested by a user in this square.

The basic term of the metric is the average local *hit-ratio* $H(i)$ achieved with the hoarding mechanism in a square i. It is defined as follows:

$$H(i) = \begin{cases} \frac{b(i)}{r(i)}, & r(i) > 0 \\ \infty, & r(i) = 0 \end{cases}, \qquad (4)$$

where $r(i)$ is the average total number of information items requested by the users while staying in square i. $b(i)$ is the average number of items which are both found among the items hoarded for square i and requested from the users in square i.

From the local hit-ratios the average hit-ratio $\overline{H}$ for a hoarding area can be calculated as follows:

$$\overline{H} = \sum_{\forall i : i \in S} w(i) \cdot H(i), \qquad (5)$$

where S is the set of squares located in the hoarding area and $w(i)$ is the weight of square i, which is defined as follows:

$$w(i) = \frac{r(i) \cdot p(i)}{\sum_{\forall j \in S} r(j) \cdot p(j)}, \qquad (6)$$

where $p(i)$ is the probability with which a users visits square i.

Since with every hoarding mechanism high hit-ratios are achievable as long as the number of information items hoarded on the mobile device is big enough, we always have to take into account the number of information items that have been hoarded in order to achieve a certain hit-ratio. So if we want to compare the hit-ratios of two different hoarding mechanisms, we have to assure that the hit-ratios are achieved with the same amount of hoarded data.

4.2   Assumptions

Initially, we assume that the average total number $r(i)$ of information items requested from a user while staying at a certain square i is the same for all squares and denote it with r. A further assumption is that the sets of information items related to different squares do not overlap. We also assume that the users request exactly the items they are expected to request according to the correlation table. If this assumption should not hold true in real scenarios, only the absolute hit-ratios will decrease, but the qualitative results of the comparisons between the simple mechanism and our mechanism will stay the same.

For the average number $b(i)$ of information items requested at square i and found among the items hoarded for square i this means:

$$b(i) = \begin{cases} h(i), & r > h(i) \\ r, & r \leq h(i) \end{cases},$$

where $h(i)$ is the number of items hoarded for square i. Unless otherwise specified we used the following parameter settings for our analyses: The size of the considered hoarding area was set to 100 squares, the maximum amount of hoarded data to $T = 4000kB$, and the size s of an information to $s = 40kB$. For r we chose a default of 5 requests per square.

We also assumed that the expected speed e is the same in all squares and the visit probabilities of the squares are distributed according to a Gaussian distribution. So, we can calculate the visit probability $p(i)$ of square i as:

$$p(i) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(i-\mu)^2}{2\sigma^2}}, 1 \leq i \leq 100$$

Initially, we chose $\sigma = 10$ and $\mu = 50$. The case where both visit probability and expected speed are randomly distributed is considered separately later on in this section. In Subsection 4.4 we also give up the assumption that the number of requested items $r(i)$ is the same for all squares i. In Table 2 we summarized our default parameter settings.

4.3   Constant Speed

The results presented in this section are all based on the assumption that the average speed of the users' movement is the same in all squares. We set the constant c in formula (1) for the determination of the hoarding scores to 0. This means the expected speeds have not been considered, when the hoarding scores were calculated.

According to equations (1) and (2) we can determine the relative hoarding score $\overline{s}(i)$ of square i as

9

Parameter	Value
size of hoarding area	100 squares
amount of hoarded data T	4000 kB
size of information item s	40 kB
# requested items per square r	5
distribution of visit probabilities	Gaussian $\sigma = 10$, $\mu = 50$

Table 2: Parameter settings.

follows:

$$\overline{s}(i) = \frac{\frac{\overline{p}(i)+c \cdot \overline{e}(i)}{c+1}}{\sum\limits_{j=1}^{100} \frac{\overline{p}(j)+c \cdot \overline{e}(j)}{c+1}} = \frac{\frac{p(i)}{p_{max}}}{\sum\limits_{j=1}^{100} \frac{p(j)}{p_{max}}} = p(i)$$

With (3) we can calculate the number $h(i)$ of items our mechanisms hoards for square i:

$$h(i) = round\left(\frac{t(i)}{s}\right) = round\left(\frac{\overline{s}(i) \cdot T}{s}\right)$$

Figure 7 shows the values of $h(i)$ we get for the 100 considered squares. With the simple algorithm one
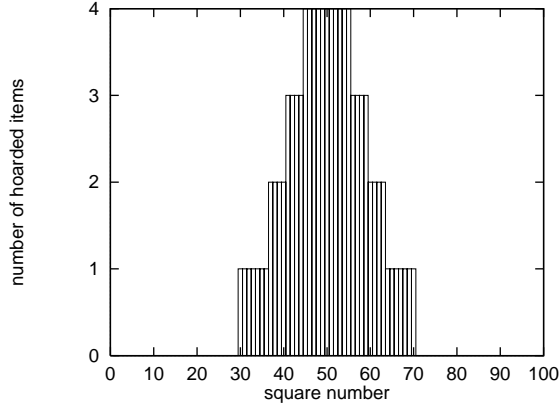

Figure 7: Number of items hoarded for each square.

item is transferred for each square, since the maximum amount of data T allows to transfer a total of $\frac{T}{s} = 100$ items or one item per square.

With (4), (5), (6), and the assumption that $r(i) = r$ for every square i, we get for the average hit-ratio $\overline{H}$ achieved with our mechanism in the considered hoarding area:

$$\overline{H} = \sum_{i=1}^{100} \left(\frac{r \cdot p(i)}{r \cdot \sum\limits_{j=1}^{100} p(j)} \cdot H(i)\right) = \sum_{i=1}^{100} \left(p(i) \cdot \frac{b(i)}{r}\right)$$

In contrast, we get for the simple algorithm:

$$\overline{H} = \sum_{i=1}^{100} \left(p(i) \cdot \frac{1}{r}\right) = \frac{1}{r}$$

Number of requested items: In our first analysis we varied the number r of requested items within each square i. The plot in Figure 8 shows the average hit-ratios $\overline{H}$ we got for both our proposed mechanism and the simple mechanism. The hit-ratios show that it is highly beneficial to consider the user's context, since our proposed algorithm outperforms the simple approach, except for $r = 1$. For this small number of requests the simple algorithm is better, since it transfers one item for every square. Therefore, the requested item can always be found in the hoard. Our mechanism, however, does not transfer any item for some seldom visited squares. If the users visit such a square, they will suffer from a hoard miss. Therefore, we get a slightly smaller hit-ratio with our mechanism in this special case of only one request per square.
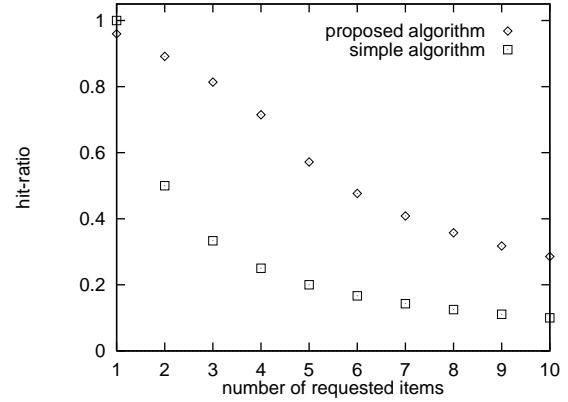

Figure 8: Achieved hit-ratios for different numbers of information items requested in each square.

10

Number of hoarded items: We also examined the effect that the total number of information items transferred between server and mobile device has on the hit-ratios. In other words we varied the value of $\frac{T}{s}$. Thereby, it does not matter, if we vary the size s of an information item or the amount T of data transferred. The results for our proposed mechanism and the simple mechanism are depicted in Figure 9.
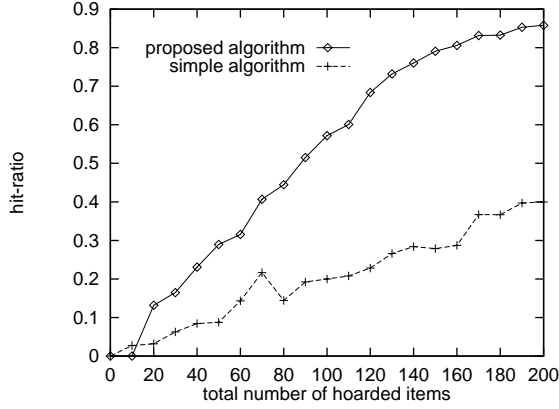

Figure 9: Hit-ratios for different numbers of hoarded information items.

Distribution of visit probabilities: Since we do not know, according to which distribution the visit probabilities in a real environment will be distributed, we were interested in the stability of our results when the type of the distribution of the visit probabilities changes. Therefore, we repeated our first analysis with different distributions. The results in Figure 10 show that the performance of our mechanism is always better than that of the simple algorithm independent of the type of distribution used for the visit probabilities.

Since, our mechanism relies on exploiting differences in the visit probabilities of different squares, the hit-ratio becomes worse when the differences in the visit probabilities decrease. Figure 11 illustrates this effect. It shows the hit-ratios for a Gaussian distribution with different standard deviations. In scenarios where the users have clear preferences in the visited squares, i.e. the standard deviation of the Gaussian distribution is low, our algorithm achieves its best results. But even for high standard deviations, i.e. when the visit probabilities do almost not differ, our algorithm does not perform worse than the simple algorithm.
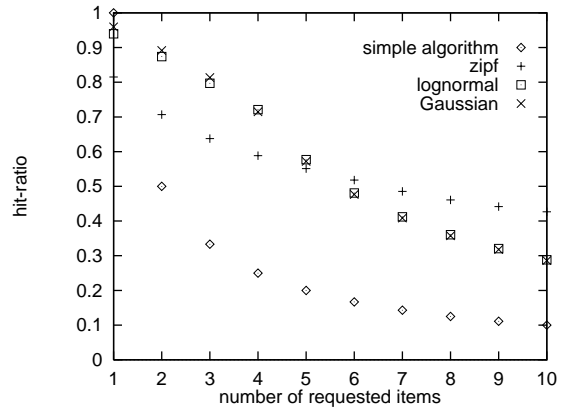

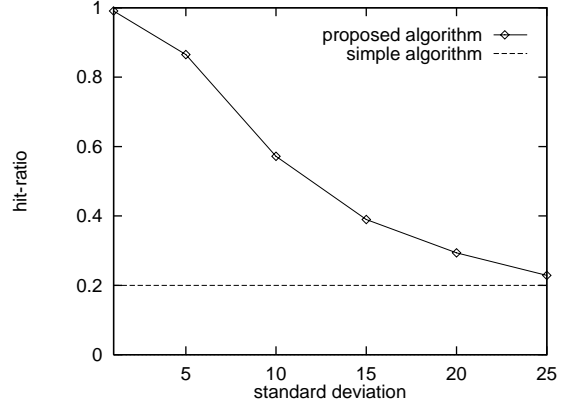Figure 10: Hit-ratios for different distribution types.

Figure 11: Hit-ratios for different standard deviations.

4.4 Variable Speed

We also analyzed the effect a random distribution of a user's expected speeds within the squares has on our mechanism. To assure that the mechanism considers the expected speeds, we set the constant c in formula (1) for the hoarding scores to 1. We assumed that the expected relative speed $e(i)$ of a user in square i can be calculated as follows:

$$e(i) = v_{min} + \left(1 - \frac{p(i)}{p(\mu)}\right) \cdot (v_{max} - v_{min})),$$

where v_{max} is the maximum and v_{min} the minimum speed. The motivation for this assumption is that the users will probably stay longer in the popular, often visited squares than in seldom visited ones. Furthermore, we assume that the number of information items $r(i)$ requested in square i depends on the ex-

11

pected relative speed in this square:

$$r(i) = R - round\left(\left(1 - \frac{p(i)}{p(\mu)}\right) \cdot R\right),$$

where R is the maximum number of items requested in a square. If a user crosses a square fast, only a few items will be requested. If the square is crossed slowly, many items will be requested. Although the assumptions about the distribution of the user's expected speeds and the number of requested items are somehow speculative, we can use them to show the potential benefit we can get from considering a user's future context. Figure 12 shows the hit-ratios we get for different values of R.
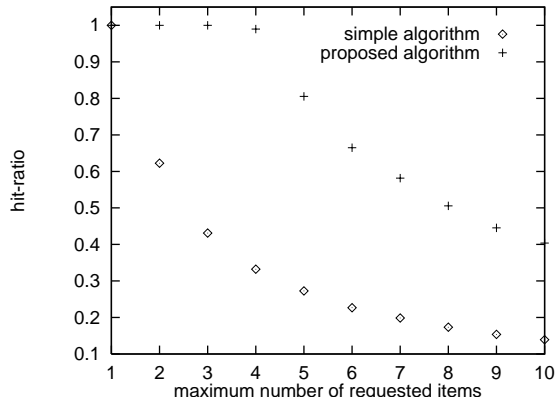

Figure 12: Hit-ratios for different maximum numbers of requested items.

Finally, we determined the number of items that have to be transferred to achieve a given hit-ratio. For this analysis we set the maximum number R of items requested in a single square to 5. The results illustrated in Figure 13 show again the high potential benefit of considering the users' context.

Note, that in this paper we have only analyzed the potential benefits we can get, if the visit probabilities and expected speeds are distributed according to our assumptions. To analyze the efficiency of our mechanism in a real scenario, we would have to verify if these assumptions are correct. But therefore, we need statistics about the behavior of real users, which we do not have so far. However, we showed that even if there are only small preferences in the visited squares, we already get benefits from considering the users' future context.
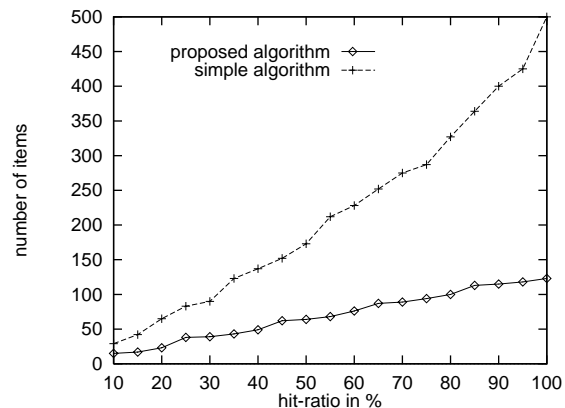

Figure 13: Number of items needed to achieve a given hit-ratio.

5 Related Work

In this section we reflect the work on mobile information access done so far. We evaluate the usability of the existing approaches in the context of location-dependent information systems and compare them to our solution.

In (Chang et al., 1997) an asynchronous information access is proposed, i.e. if an information request occurs while no or only a low bandwidth is available, it is delayed until a high bandwidth network connection is available. The problem with this approach is that the users might then not be interested in the requested information anymore, as in the meantime they moved on to another location.

Other approaches, like (Davies et al., 1999) or (Hu et al., 1999), are based on broadcast dissemination of information. Their primary focus is to reduce the response time and to increase the scalability of the system. If they are location-aware, they are designed to support the users with the information items they need at their current location, e.g. within the coverage area of one cell of the dissemination system. Therefore, broadcast-based dissemination mechanism do not make any predictions on the information items the users will need after leaving for another location. Furthermore, they are usually based only on the users' access patterns and do not use any available external knowledge. If the access patterns of the users differ strongly the efficiency of these approaches decreases. However, a broadcast based information dissemination might be useful in our approach to decrease the bandwidth required for the hoarding processes at the info-stations.

The first hoarding approaches that were especially designed to support users during disconnections, e.g. (Satyanarayanan et al., 1990), relied on user interactions and required a list of the user's preferred information items. This is not applicable in our scenario, because the users do not know in advance which information items they will access. In (Kuenning and Popek, 1997) an automated hoarding mechanism is proposed, which uses semantic distances between files in order to predict which files a user will need. In contrast to our approach the user's location is not considered there. The hoarding tool described in (Tait et al., 1995) also relies only on file access patterns.

In (Ye et al., 1998) the user's position and movement pattern is considered for the determination of the items to be hoarded. This approach is focused on a map application for people driving on roads and can not be used as a generic mechanism for different types of location-dependent information systems. For example, the considered request patterns are restricted to the driving scenario. It is also assumed that the start and end point of the users' trips are known. With our mechanism this is not necessary, since it can rely on the internal knowledge to determine the direction a user will probably follow.

In (de Nitto Personè et al., 1998) it is assumed, like in our approach, that information items can be mapped to certain areas. This work provides a valuable analysis of the effectiveness of location-aware hoarding. However, it is again focused on response time reduction and not on disconnected operations. Knowledge about the users' future movements is only used in the case of a linear movement, e.g. along a road. Then, more information belonging to the area in the users preferred direction is hoarded than for the area in the opposite direction.

6   Conclusion

In this paper we presented a generic, context-aware hoarding mechanism. The main advantage of it is that it can be deployed in any location-dependent information system. It is not restricted to a certain application or user category. Furthermore, it uses both external and internal sources of information about a user's context. In this way it is able to use all available knowledge about a user's future behavior to the maximum extend in order to make precise hoarding decisions.

We explained the three basic problems in develop-

ing our mechanism. Afterwards, we identified the different types of internal knowledge that can be used for the hoarding decision and showed how this knowledge can be gathered and stored in hoarding profiles. We also introduced hoarding patterns as a means for a client application or user to specify external knowledge. We finished the explanation of our mechanism with the illustration of a prototype architecture and the overall functionality of the mechanism.

Finally, we analyzed the potential benefits of using information about a user's future location and/or speed of movement for the hoarding decision. The results showed that even if there are only small preferences in the visited squares, we already get benefits from considering a user's future context.

In the near future we plan to integrate our mechanism in a platform for context-aware applications currently developed at our department (Hohl et al., 1999). We also want to install an info-station infrastructure in order to test our mechanism in a real scenario. We will then be able to fine tune our prototype architecture and to examine the effect that different parameters, e.g. the size of a square, have on the efficiency of our mechanism.

References

Abowd, G., Atkeson, C. G., Hong, J., Long, S., Kooper, R., and Pinkerton, M. (1997). Cyberguide: a mobile context-aware tour guide. *Wireless Networks*, 3(5):421–433.

Badrinath, B. R., Imielinski, T., Frenkiel, R., and Goodman, D. (1996). Nimble: Many-time, many-where communication support for information systems in highly mobile and wireless environments.
http://www.cs.rutgers.edu/~badri/dataman/nimble/ .

Chang, H., Tait, C., Cohen, N., Shapiro, M., Mastrianni, S., Floyd, R., Housel, B., and Lindquist, D. (1997). Web browsing in a wireless environment: Disconnected and asynchronous operation in artour web express. In *Proceedings of the Third Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '97)*, pages 260–269, Budapest, Hungary.

Chim, J., Green, M., Lau, R., Leong, H., and Si, A. (1998). On caching and prefetching of virtual objects in distributed virtual environments. In *Pro-

ceedings of the 6th ACM International Conference on Multimedia, pages 171–180.

Davies, N., Cheverst, K., Mitchell, K., and Friday, A. (1999). Caches in the air: Disseminating information in the guide system. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '99)*, New Orleans, USA.

Davies, N., Mitchell, K., Cheverst, K., and Blair, G. (1998). Developing a context sensitive tourist guide. In *Proceedings of the First Workshop on Human Computer Interaction with Mobile Devices*, pages 64–68, University of Glasgow, Scotland, U.K.

de Nitto Personè, V., Grassi, V., and Morlupi, A. (1998). Modeling and evaluation of prefetching policies for context-aware information services. In *Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking (MobiCom '98)*, pages 55–64, Dallas, Texas, USA.

Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., and Schwehm, M. (1999). Next century challenges: Nexus – an open global infrastructure for spatial-aware applications. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom '99)*, pages 249–255, Seattle, WA, USA.

Hu, Q., Lee, D. L., and Lee, W.-C. (1999). Performance evaluation of a wireless hierarchical data dissemination system. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom '99)*, pages 163–173, Seattle, WA, USA.

Kuenning, G. and Popek, G. (1997). Automated hoarding for mobile computers. In *Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP '97)*, pages 264–275, St. Malo, France.

Liu, G. and Maguire Jr., G. (1995). Efficient mobility management support for wireless data services. In *Proceedings of the 45th Annual IEEE Vehicular Technology Conference (VTC '95)*, pages 902–906, Chicago, IL, USA.

Satyanarayanan, M., Kistler, J., Kumar, P., Okasaki, M., Siegel, E., and Steere, D. (1990). Coda: A highly available file system for a distributed workstation environment. *IEEE Transactions on Computers*, 39(4):447–459.

Tait, C. D., Lei, H., Acharya, S., and Chang, H. (1995). Intelligent file hoarding for mobile computers. In *Proceedings of the First International Conference on Mobile Computing and Networking (MobiCom'95)*, pages 119–125, Berkeley, CA, USA.

Want, R., Hopper, A., Falcao, V., and Gibbons, J. (1992). The active badge location system. *ACM Transactions on Information Systems*, 10(1):91–102.

Ye, T., Jacobsen, H.-A., and Katz, R. (1998). Mobile awareness in a wide area wireless network of info-stations. In *Proceedings of the Fourth International Conference on Mobile Computing and Networking (MobiCom '98)*, pages 109–120, Dallas, TX, USA.