In Proc. of IEEE Pervasive Computing 2003 (PERCOM'03). IEEE Press, March 2003.

Location-Detection Strategies in Pervasive Computing Environments

Sebastian Fischmeister, Guido Menkhaus, Alexander Stumpfl
University of Salzburg
Software Research Lab
{*lastname*}@SoftwareResearch.net

Abstract

Pervasive computing environments accommodate interconnected and communicating mobile devices. Mobility is a vital aspect of everyday life and technology must offer support for moving users, objects, and devices. Their growing number has strong implications on the bandwidth of wireless and wired networks. Network bandwidth becomes a scare resource and its efficient use is crucial for the quality of service in pervasive computing. In this article we study process models for detecting location changes of moving objects and their effect on the network bandwidth. We simulate a scenario of 10^4 *moving objects for a period of* 10^7 *time cycles while monitoring the quality of service with respect to network bandwidth for different location detection strategies. The simulation shows that the class of strategies implementing a synchronous model offers better quality of service than the timed model. We conclude the article with a set of guidelines for the application of the strategies we have investigated.*

1. Introduction

Technological advances allow manufacturing smaller and more portable but increasingly powerful computing devices, which are equipped with wireless communicating capabilities. The integration of information processing into wirelessly interconnected everyday objects is the primary goal of pervasive computing [12, 9]. A pragmatic variant of the pervasive computing paradigm has been described by IBM chairman Lou Gerstner: *a billion people interacting with a million e-businesses through a trillion interconnected intelligent devices.* People move around and therefore it is important for such a scenario to operate that e-businesses and the interconnected devices know where they are and what other objects are in their vicinity. Location information seems crucial information for e-businesses and services. The pervasive environment is strongly interconnected and the participants profit from this fact by detecting their surrounding and offering proactively their service to objects in their vicinity.

Considerable advances have also been made in the area of wireless communication [10, 1, 11]. Methods to detect the location of mobile objects have been developed [6, 5]. These systems (wireless communication techniques and methods to detect the location of objects) need to be designed to cope with the fast growing number of interconnected users, objects, and services. The systems are designed according to the following two models: synchronous model and timed model [7]. In the synchronous model, a service is executed in the context of some process that generates events as stimulus for the service. An event is generated when a user or an object changes its location. In the timed model, the service checks at fixed intervals each object if it has generated an event. Locating systems implement the timed model, whereas location systems are designed around the synchronous model (c.f. Section 2).

In this article we consider the following scenario: A device might offer or perform a service cooperatively with other objects or devices in its vicinity. Therefore, the device must continuously track the users and devices entering and leaving its location. It can eventually query newly arrived objects to cooperate. We study process models for detecting location changes of objects. In this process models we use symbolic positions as locations [8]. Symbolic positions describe abstract areas in which objects are located and in which several objects can reside; in contrast to physical positions that are unique for each object. The timed model and two processing strategies for the synchronous model are simulated and evaluated within the proposed process model. We show that the synchronous model provides higher quality of service (QoS) than the timed model.

The remainder of the article is structured as follows: The motivation for this work is presented in Sec-

tion 2. Section 3 describes the simulation model and the processing strategies. The results are discussed in Section 4. The article closes with a conclusion in Section 5.

2. Motivation

Pervasive computing environments often include wireless communication techniques and methods to detect the location of objects. This can be accomplishes either with locating systems or with location systems. The key difference between locating and location systems is that locating systems focus on the identification of the location of objects, whereas location systems identify objects at specific locations. Locating systems are used in GSM and other wireless communication systems. Location systems can be implemented, e.g., with WLAN [2, 3].

Location-aware services derive their profit from merging two pieces of information: The identification of an object $o \in O$ and its location $l \in L$, with O being the set all objects participating in the system, and L being the discrete set of locations of a system. For example, in the context of mobile telephony, an object may represent a cellular telephone and thus its user. The location is equivalent to the area covered by the access point of the user. The sets O and L form a new set: the object – location space S, $S = O \times L$. An element of the object – location space, $s = (o, l)$ is used to parameterize a service. On the one hand, s can be interpreted as an object o being at a location l. On the other hand, s can be seen as a location l where an object o currently resides.

Generally, the information required by a location-aware service is only partially known at service request-time. To successfully process a service request, both pieces of information, the identification of an object as well as the location of this object, need to be known. That means that the service derives one piece of information from the other.

2.1. Locating systems

Services using a locating system know the identity of their clients (e.g, through authentication), but are ignorant about their location. Locating systems serve to derive the location of an object on the basis of its identification. These systems give answer to the question (Figure 1): At which location l is the object o. Locating systems implement the mapping $\lambda : O \rightarrow L$, where every object has a location and at most one location.

Locating systems retrieve the information where a specific object resides. Reactive services use it to determine the location of the client. Reactive services are used by clients that actively "pull" information from the service by issuing an explicit request. Proactive systems use the system to permanently track the position of a specific object. Proactive services deliver or "push" information to a client without explicit request [4].

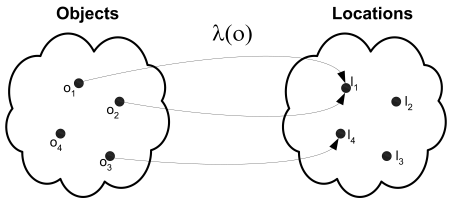

Figure 1. Mapping objects onto locations. Object o_1 and o_2 are at location l_1 and object o_3 is at location l_2.

2.2. Location systems

Locating systems derive the identification of a subset of objects on the basis of a location. They give answer to the question (Figure 2): Who or what are at location l. Location systems map the location l to a set of objects: $\omega : L \rightarrow \mathcal{P}(O)$. $\mathcal{P}(O)$ is the power-set of O. Every location contains zero to several objects.

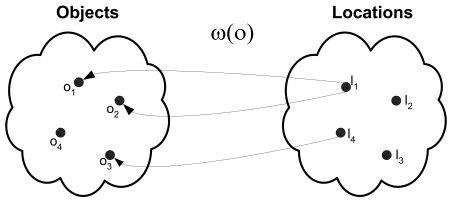

Figure 2. Mapping locations onto objects. Location l_1 hosts the objects o_1, o_2, and o_3. Location l_2 hosts the object o_4.

2.3. Discussion of Locating and Location systems

The popularity of wireless communication and the penetration of more and more portable communication devices made it necessary to efficiently locate mobile users. Locating systems are very good in locating a user. However, new types of services, which operate on group of objects having a specific feature, enjoy more and more popularity (e.g., location-based instant messaging). These services require location systems, which effectively calculate the set of objects residing at the

same location. The next paragraph shows that the effort of simulating one type of system by the other is considerable:

Locating system simulating a location system. In order to derive the location for each object in the system, the location for each object needs to be compared to a given location m: The set of objects P at a given location m is $P = \{o \in O \,|\, \lambda(o) = m\}$. In this case ω would be defined as $\omega(l) = P, P = \{o \in O | \lambda(o) = l\} \subseteq O$. Locating systems employ the timed model. They check periodically at fixed intervals which objects reside at the location m by trying to scan every object in its domain.

Location system simulating a locating system. In order to locate an object o, location systems need to verify for each location l whether the object is present at that specific location, until the object is found: that means checking if $o \in \omega(l)$ for all $l \in L$. Thus, λ would be defined as $\lambda(o) = \bigcup\{l \in L \,|\, \omega(l) = o\}$. Location systems apply the synchronous model, where the environment generates events for objects changing their location. The amount of events generated is determined by the behavior of the context of the objects.

3. Simulating processing strategies for user migration

Having acknowledged the importance of location information in a pervasive computing environment and their efficient processing, we simulated three processing strategies for detecting location changes of objects. In this section we describe the simulation, our model, and the assumptions for testing strategies that process user migration information for proactive services.

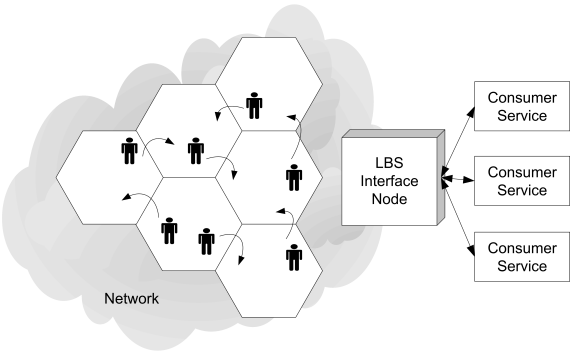

Figure 3. Overview of a traditional cell-based locating system.

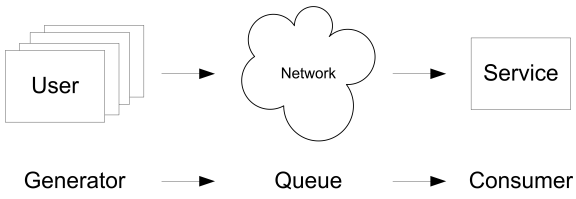

Figure 4. Simulation model.

Figure 3 shows a traditional locating system that uses symbolic locations. Objects move and change their symbolic location. An interface server provides access to the location information of the users for the consumer services. Various consumer services use this interface server to provide their service to the users. We simplified this model and created our simulation model (see Figure 4). Users generate location-change events when migrating from one location to another. Location-change events are dispatched into the network and delivered to the service (we put together the interface server and the network to create a unified model for all three strategies). The service consumes the events. The model consists of the following elements:

- **Generator.** We model the user migration behavior through cell residence time at one location. Cell residence time is defined as the time between the point in time the user enters the cell and the point in time the user leaves the cell. We follow the model of Zonoozi and Dassanayake who showed that cell residence time can be described by a generalized gamma distribution [13]. For our location-based test-bed network we determined the parameters $\alpha = 2$ and $\beta = 60$.

- **Queue.** Different types of queuing systems represent the model of the network. Each strategy is represented by a different type of queuing system, e.g., using priorities or using a ring buffer. The simulation consists of the following three strategies described below: poll strategy, drop strategy, and defer & drop strategy.

- **Consumer.** The location-based service is modeled as the consumer of location-change events. The consumer processes the events and removes them from the queue immediately. For the consumer we assume zero time computation. This means that the consumer can process as many events as are generated.

The simulation measures the QoS for different strategies simulating different network bandwidth.

QoS is defined as the number of missed location-change events in relation to the total number of location changes. Thus, a QoS of 80 means that a strategy correctly processes 80 percent of all possible location changes and misses 20 percent. This percentage refers to the overall number of location changes and is not specific for a particular user. The simulation evaluates the QoS of the following three strategies:

- **Poll strategy.** The poll strategy implements the timed model by a time-triggered strategy. It represents a locating system simulating a location system. The consumer regularly checks all users to determine the set of users that changed their location. This means that it polls information about location-change events. This strategy can be configured in terms of the poll interval (e.g., poll each person each time slice, every second time slice, or every n-th time slice).

  The QoS of this strategy is defined as the number of successfully detected location changes in relation to the number of total location changes. For example, if person A switches location once per time slice, this strategy will detect every change given a poll interval of one. However, if the poll interval is two, the same strategy will only detect every other location change. The poll interval is determined by the available network bandwidth and the number of users:

$$\text{poll interval} = floor\left(\frac{\text{number of users}}{\text{network bandwidth}}\right).$$

- **Drop strategy.** The drop strategy implements the synchronous model with an event-triggered process. The system detects location-change events and delivers them to the consumer. The network transports as many location-change events as the bandwidth allows to the consumer. If the number of events within one time slice exceeds the maximum number, the remaining events are dropped. The consumer consumes all events that it receives. This strategy can be configured in terms of the network bandwidth.

  The QoS of this strategy is defined as the number of successfully delivered location-change events in relation to the number of total events. Thus, each dropped event is missed and decreases the QoS.

- **Defer & Drop strategy.** The defer & drop strategy manifests the synchronous model. It is implemented using an event-triggered strategy. The network transports the maximum number of location-change events to the consumer. If the number of events within one time slice exceeds the maximum number that the network bandwidth allows to transport, the remaining events are queued and deferred to the next time slice: their transport is *deferred* to the next time slice. If the age of the deferred events exceeds a specified threshold measured in the number of time slices, they are dropped. The defer algorithm uses first in, first out mechanism; This means that deferred events are delivered to the consumer before new events. The consumer consumes all events that it receives. This strategy can be configured in terms of network bandwidth and defer-time threshold. The higher the defer time, the longer an event can be deferred.

  The QoS of this strategy is defined similar to the drop strategy: The number of delivered events in relation to the total number of events. We do not take the defer time into account when calculating the QoS. If deferred events decrease the quality, the defer time needs to be reduced. However, a lower defer time entails a higher drop rate and therefore affects quality.

Beyond these three strategies there are many other possible strategies; especially, using more complex priority queues. However, we settled with these three strategies as they demonstrate the main issues of this research article.

4. Results

The simulation of the three strategies uses the same data: the user residence time was generated and stored to be reused in each run. For each run, we plugged into the simulation model different strategies. We simulated 10^4 users for a period of 10^7 time cycles.

Figure 5 shows the comparison of the QoS in relation to the network bandwidth of all three tested strategies and with different defer-time thresholds for the defer & drop strategy. The x-axis shows the maximum network bandwidth in messages per time slice. The y-axis shows the quality in percent calculated by relating the number of detected and processed location changes to the total number of location changes.

The first point of interest (POI 1) is around (32,43): at this point, the poll strategy splits off from the defer & drop strategy. This means that below 43 percent QoS the two strategies, poll and defer & drop, deliver the same QoS. Above 43 percent the defer & drop strategy delivers a much better QoS than the poll

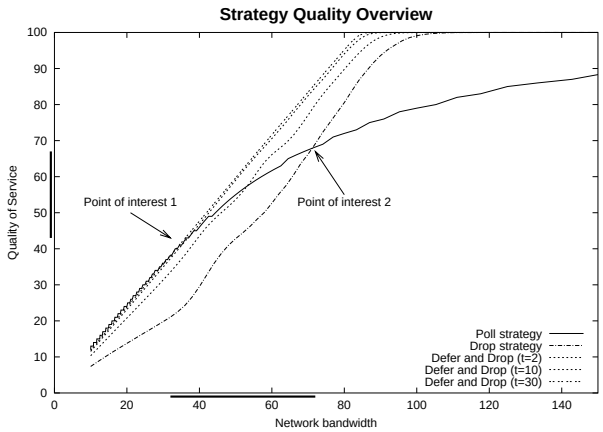

Figure 5. Overview of the QoS of the three strategies.

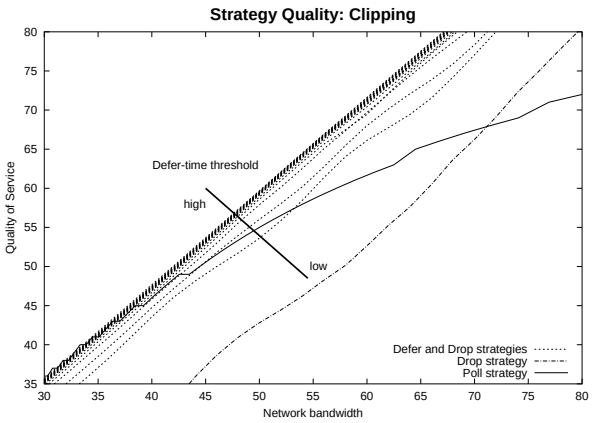

Figure 6. Detailed clipping of the 85 to 98 percent area of QoS.

strategy; at this level, event-triggered strategies (drop and defer & drop) still have a steep increase in quality while the time-triggered strategy (poll) starts to flatten down. The reason is that at this point the overhead of polling and of simulating a location system with a locating system starts to show a strong effect. However, the defer & drop strategy is not immediately superior above 43 percent QoS. The quality of this strategy depends largely on the defer-time threshold. POI 2 at (71,68) shows that it takes a while until the drop (defer & drop with threshold zero) outperforms the poll strategy. Thus between 43 and 68 percent QoS, it depends on the defer-time threshold whether defer & drop outperforms polling. Above 68 percent QoS, all event-triggered strategies, regardless their setup, outperform the poll strategy. The poll strategy reaches a QoS of about 100 at a bandwidth of 650 (about six times more than event-triggered strategies).

Figure 6 shows a detailed view on the QoS area between 35 and 80 percent. In this area, the defer & drop strategies start outperforming the poll strategy. The higher the defer-time threshold of the strategy, the better the QoS it delivers. However, the defer & drop does not outperform the poll strategy easily. Even high defer-time thresholds do no deliver a much better QoS than low ones. In Figure 6, we show different setups of defer & drop strategies and, while the first increases of the defer-time threshold provide a steep increase in QoS, the latter increases do not.

There are qualitative differences between the processing strategies to remark. In the previous paragraphs, we demonstrated that the event-triggered strategies (defer & drop and simple drop) can offer better QoS than the time-triggered strategies (poll). However, time-triggered strategies have advantages regarding managing and planning aspects. The poll interval directly controls network utilization. By increasing and decreasing the poll interval, for example depending on the time of day, QoS can be determined. This is important, if the network is not exclusively used for transporting location-change events but for other services as well. If other services temporarily require more bandwidth, one can immediately react upon such requests.

Furthermore the defer & drop strategies are more sophisticated than the simple drop strategy. Although defer & drop easily outperforms dropping, the simple drop strategy does not require any buffering of network packets. Table 1 presents guidelines, when to use which strategy.

5. Conclusion

The mobility of highly interconnected and communicating devices and users has implications for the QoS in a pervasive computing environment. Network bandwidth is a precious resource representing an upper limit for the QoS. However, the limit can be influenced by different processing strategies for detection of location changes. The simulation discussed in this article demonstrated the impact of three processing strategies on the QoS with respect to network bandwidth. Table 1 shows the results and represents guidelines for the application of the strategies in different QoS scenarios. We summarize that synchronous, event-triggered models can outperform the timed, time-triggered model and conclude that the class of location systems for location-change detection may be very adequate for the support of pervasive computing environments. From these re-

Quality	Strategies
below 43 %	poll strategy
between 43 and 68 %	defer & drop outperforms polling (depending on the threshold)
above 68 %	defer & drop strategy or drop (depending on threshold and level of complexity)

Table 1. Guidelines: Quality of service. We assume that the order of the strategies arranged by complexity is: poll, drop, defer & drop.

sults we can infer two main points:

- *Intuitive choices mislead—necessity to evaluate each case.* The intuitive choice would be always to use the synchronous model. However given the length of a time slice, the timed model may perform well enough. Especially, when a lower QoS is sufficient for the application.

- *Location systems will become important for pervasive computing.* Currently most producers of location-based systems use the timed model, only. However when building services that require a high QoS, location system and the synchronous model perform better and require lower infrastructure costs. Thus, location systems will play an important role for pervasive computing.

In our future work we will build on these encouraging and controversial results. We expand our model in two ways: We examine the simulation of timed models using synchronous models and add more complex strategies to produce better results for networks with low bandwidth.

References

[1] Bluetooth Special Interest Group. *Specification of the Bluetooth System*, 1.0b edition, Dec. 1999. available from `http://www.bluetooth.com`.

[2] S. Fischmeister. Mobile software agents for location-based systems. In *Proc. of Agent Technology and Software Engineering*, 2002.

[3] S. Fischmeister, G. Menkhaus, and W. Pree. MUSA-Shadow: Concepts, Implementation, and Sample Applications; A Location-Based Service Supporting Multiple Devices. In *Proceedings of Pacific TOOLS*, pages 71–79, Sydney, Australia, February 2002.

[4] M. Hauswirth and M. Jazayeri. A Component and Communication Model for Push Systems. In *Joint 7th European Software Engineering Conference and 7th ACM SIGSOFT International Symposium on the Foundation of Software Engineering.*, page 20 to 38, 1999.

[5] J. Hightower and G. Borriello. Location Sensing Techniques. Technical Report UW-CSE-01-07-01, University of Wahsington, Computer Science and Engineering, Box 352350, Seattle, WA 98195, Aug. 2001.

[6] J. Hightower and G. Borriello. Location Systems for Ubiquitous Computing. *IEEE Computer*, 34(8):50 to 56, Aug. 2001.

[7] C. M. Kirsch. Principles of Real-Time Programming. In *Proceedings of Second Inernational Conference on Embedded Software*, pages 61–75, Grenoble, France, October 2002.

[8] U. Leonhardt. *Supporting Location-Awareness in Open Distributed Systems.* PhD thesis, Imperial College of Science, Technology and Medicine, University of London, May 1998.

[9] F. Mattern. The Vision and Technical Fundations of Ubiquitous Computing. *UPGRADE*, 2(5):1–4, 2001.

[10] B. Miller and C. Bisdikian. *Bluetooth Revealed.* Prentice Hall, 2001.

[11] B. O'Hara and A. Petrick. *The IEEE 802.11 Handbook: A Designer's Companion.* Standards Information Network IEEE Press, 1999.

[12] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Persoanl Communications*, pages 10–17, August 2001.

[13] M. Zonoozi and P. Dassanayake. User Mobility Modeling and Characterization of Mobility Patterns. *IEEE Journal on Selected Areas in Communications*, 15(7):1239 to 1252, 1997.