Scalable Location Management for Context-Aware Systems*

Jadwiga Indulska[1], Ted McFadden[2], Matthias Kind[2], and Karen Henricksen[1]

[1] School of Information Technology and Electrical Engineering
The University of Queensland, St Lucia QLD 4072 Australia
{jaga,karen}@itee.uq.edu.au
[2] CRC for Enterprise Distributed Systems Technology (DSTC)
The University of Queensland, St Lucia QLD 4072 Australia
{mcfadden,mkind}@dstc.edu.au

Abstract. Location information is commonly used in context-aware applications and pervasive systems. These applications and systems may require knowledge of the location of users, devices and services. This paper presents a location management system able to gather, process and manage location information from a variety of physical and virtual location sensors. The system scales to the complexity of context-aware applications, to a variety of types and large number of location sensors and clients, and to geographical size of the system. The proposed location management system provides conflict resolution of location information and mechanisms to ensure privacy.

1 Introduction

Pervasive/ubiquitous systems require context awareness to provide both a seamless computing infrastructure and adaptive context-aware applications to mobile users. Computing devices currently take many forms, from traditional mobile devices such as mobile phones and handheld computers, to networked home appliances, wearable computers and "smart items" (objects with embedded storage, computing and communication capabilities [1] which can create communities of smart items and can interact with other entities).

Pervasive systems need to deal with mobility of users, their devices and their applications and also with users who may want to change their computing device whilst running some computing applications. A pervasive computing infrastructure should allow users to move their computational tasks easily from one computing environment to another and should allow them to take advantage of the capabilities and resources of their current environment. As a result, pervasive systems have to be context-aware, i.e., aware of the state of the computing environment and also of the requirements and current state of computing applications. One type of context information is location information (e.g., the

* The work reported in this paper has been funded in part by the Co-operative Research Centre Program through the Department of Industry, Science and Tourism of the Commonwealth Government of Australia.

J.-B. Stefani, I. Demeure, and D. Hagimont (Eds.): DAIS 2003, LNCS 2893, pp. 224–235, 2003.

location of users, devices and services) which needs to be gathered from a variety of location sensors. The location information can be physical (e.g., location provided by a GPS device and a variety of other physical sensors), or virtual (e.g., a calendar application, camera reading, or IP address).

There are already many approaches to define location information, to track the location of users and devices, to manage such location information, and to utilize location information to support mobile users. Location management systems which gather and manage location information can be used for a variety of purposes and their complexity depends upon their purpose. Such systems are often used to provide users with information which is specific to user location (e.g. tourist guides [2]) or to support the delivery of personalized and location-sensitive information as in the BlueLocator project [3]. Location information can also be used as one of the elements of complex context information which supports pervasive/ubiquitous systems [4]. As pervasive systems are very complex and can use location information from a variety of sources and for a variety of purposes, gathering and managing location information is a challenging task. There are a variety of issues which need to be addressed in such systems, including: (i) types of location information (many sources and kinds of location information), (ii) location resolution and associated errors, (iii) resolution of conflicting location information, (iv) location information access and privacy (who can access location information and how can it be used), (v) architectures of location management systems, and (vi) integration of location management system with general context management in pervasive systems in a way which ensures scalability of the whole system.

In this paper we present a scalable location management system (LMS) which can deal with location information requests of various complexity from simple location services which deliver a reading from a single user location senor (e.g. GPS in a PDA) to a complex support of infrastructures in pervasive systems. The latter requires location information from a variety of sensors and therefore conflicting location information is possible and privacy issues need to be addressed. We describe the architecture of such a system and present our solution for the difficult issue of conflict resolution. We also show how a Platform For Privacy Preferences (P3P) [5] privacy approach has been integrated into our LMS to provide a manageable solution for defining privacy rules for location information.

The structure of this paper is as follows. Section 2 presents an overview of issues related to processing of location information, describes the architecture of our location management system and discusses the scalability of this system. Section 3 describes our solution for resolution of conflicting location information and compares this solution to the existing IBM approach for aggregation of location information. Section 4 shows the application of P3P for defining privacy rules for location information access and illustrates an extension to provide a user friendly mechanism for defining privacy policies. Section 5 concludes the paper.

2 Challenges in Location Management

2.1 Sources of Location Information

Location sensors can be physical or virtual. Physical location sensors provide information about the position of a physical device. Examples of devices are GPS receivers, mobile phones, passive and active badges, cameras used for face recognistion, magnetic-stripe cards and even simple barcodes. Physical location sensors can provide either position or proximity information. Position sensors attempt to provide the coordinates of a device relative to some coordinate system. Different sensors will have different resolutions and associated errors. Proximity sensors locate an entity or device as being within a region but cannot precisely position the device within that region. Region sizes may range from tens of centimeters (e.g. RFID passive tags) to tens of meters (e.g. Bluetooth and 802.11 cells) to hundreds of meters (e.g. mobile phone cells). Proximity sensors with overlapping detection regions can form the basis of position sensors (via triangulation or trilateration).

Virtual location sensors extract location information from virtual space, i.e. software applications, operating systems and networks. Sensor processes can monitor application events (a networked calendar system, travel booking, etc.), operating system events (monitoring keyboard or mouse use by a logged-in user, file server use, etc.) or network events (IP address, etc). The location readings from virtual sensors usually have to be combined with information from other sources (e.g. a database of the locations of fixed equipment like magnetic card readers or desktop computers) to infer physical location information.

2.2 Processing of Location Information

Sources of location information are heterogenous. Location sensor agents produce fragments of location information which differ in type, spatiotemporal characteristics (location, orientation, motion, time), resolution (i.e. level of accuracy), and data format - to list the most important differences. Sensor agents may have different policies governing when location information is reported.

The location information has to be gathered from sensors and reconciled, i.e., various types of location information about particular entities (e.g. users, devices, applications, and smart items) have to be compared and evaluated in order to compute the location of the entities. Therefore, the location information represented in formats specific for particular sensor types has to be transformed to a common format. Moreover, if the location readings provide conflicting information such conflicts have to be resolved. Conflicts which stem from differences in sensor resolutions are easy to resolve. There are, however, numerous conflicts which go beyond differences between position and proximity location sensors and also differences in resolution. If a user has many devices (e.g. mobile phone, PDA, laptop) which can provide physical location information and in addition the system is able to gather some virtual location information (e.g., from software agents interacting with the user) the location management system may need to

deal with conflicting information and has to provide a means for resolution of such conflicts.

When processing location queries another important issue has to be addressed: privacy of location information. The importance of this issue is illustrated by the difficulties that many telecommunication companies currently have with providing location-based services and which now limit the provision of user location information to user's devices only.

2.3 Architecture and Scalability

Location management systems gather location information from many sources and process it to create the final representation available for a variety of clients. The location management systems developed so far for mobile distributed computing and location-based applications usually have a hierarchical architecture with the following layers: Application/Presentation Layer, Fusion Layer, Abstraction Layer, Reception Layer, and Sensor Layer [6]. The hierarchical architecture reflects the complex functionality of location management systems as shown in the following brief description of the functionality of particular layers:
Sensor Layer. The lowest level of the location management architecture is the Sensor Layer which represents the variety of physical and logical location sensor agents producing sensor-specific location information.
Reception Layer. The fragments of location information produced by sensor agents are delivered by the Reception Layer to the Abstraction Layer.
Abstraction Layer. This layer takes sensor-specific location information delivered by the Reception Layer and transforms it into a standard format. This layer needs to capture relationships between locations and their identifying features as it needs to associate attributes (eg. mobile cell identifiers) with physical places (for example, suburbs). The Abstraction Layer has to provide these transformations for both physical and virtual location information and needs to provide mappings between virtual (conceptual) and physical location information.
Fusion Layer. The Fusion Layer aggregates the location information gathered by the abstraction layer for a particular entity to provide a single, coherent location of the entity. If there are conflicts in the location information they should be resolved at this layer.
Application/Presentation Layer. This layer interacts with the variety of clients of the location management system and therefore needs to address several issues including access rights to location information (who can access the information and to what degree of accuracy), privacy of location information (how the location information can be used) and security of interactions between clients and the location management system.

Our location management system in general follows the above hierarchical architecture, however some solutions were introduced at particular layers to achieve our goal which is to provide a scalable location management system able to deal with conflicts of location information and ensure privacy of location information.

The issue of scalability of location management systems is important as these systems have to deal with the following challenges:

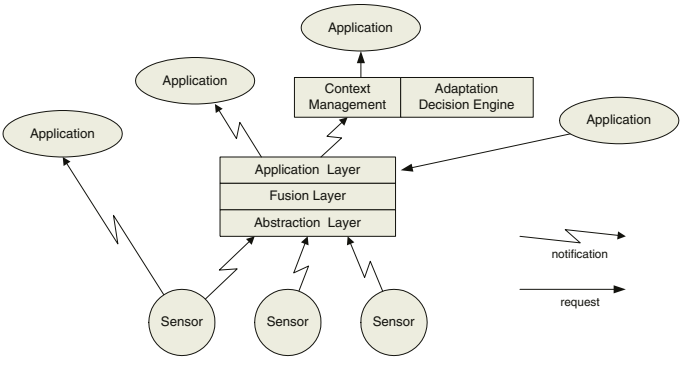


Fig. 1. Location Management Architecture.

1. Various levels of complexity of location applications from simple tourist guide applications using one location sensor (e.g. a PDA with a GPS sensor) where no aggregation of location information is needed, there is no conflict possible and no location information privacy issues are involved, to location information support for infrastructures of pervasive systems which evaluate context changes (including location changes) to make decisions about application and/or system adaptation to the changing context.
2. Potential large number of sensors and a large number of clients of the location management system.
3. Distribution of location management (users moving through domains).
4. Large number of updates of location information for moving objects.

There is a large body of research for the problem described in 4 coming from the database community, therefore we concentrate in this paper on the first three problems.

Various levels of complexity. As shown in the architecture of our LMS (Figure 1) simple applications can directly receive location information from the Sensor Layer, while other applications which require complex processing of location information are served by several layers.

Large number of sensors and clients. In our location management system we use a distributed notification service Elvin [7]. It provides the functionality of the Reception Layer as it forwards notifications between the Sensor Layer and the Abstraction Layer. The same notification system is used for delivering notifications about location changes between the Application Layer and the clients of the location management system unless a synchronous interaction is requested by an application. Elvin provides a scalable solution as it is able to cope with a large number of sensors and clients due to selective forwarding of messages. Different clients (users, applications, infrastructure of the pervasive system) may have different requirements with regard to mode of interaction (pull/push) and granularity of location information. For some clients, the pull mode (client/server) of interactions may be suitable as they need location information only when they

request it. However, for other clients (e.g. infrastructure of pervasive systems) the push model (information about location is "pushed" to the client when the location information changes) is more appropriate. Our system allows either pull or push mode to be used for client interactions. As Elvin allows entities to register for notification about location changes and the granularity of this notification, this further increases the scalability of our solution. Figure 1 illustrates both the Reception Layer (Elvin) and the clients' interactions with the system.

In addition, location information is stored in a persistent repository in the Abstraction Layer. In our LMS this repository is a relational database. This allows aggregation of location information (Fusion Layer), including conflict resolution, to be performed on demand, in response to user queries, instead of each time new sensor data is received. There could be a great difference between the frequency of location updates from sensors and the frequency of location information requests from the clients of the location management system. Moreover, once a location is determined at the Fusion Layer it is cached as a complete location description which will be valid until another update pertaining to that entity is received at the Abstraction Layer. As a result, many requests for a single entity's location can be served without significantly increasing the cost of generating the location from individual location fragments if location information updates are not very frequent.

Distribution of LMS. As pervasive systems may be geographically large and have a heterogeneous computing and networking infrastructure, one of the scalability requirements is distribution of location managers in this infrastructure. Our location management system can be easily distributed due to the layered approach and the notification system used. Wide-area coverage can be achieved by deploying a number of managers implementing the Abstraction Layer which gather location information from geographically close sensors. Multiple Abstraction Managers can update a single Fusion Layer manager. The solution allows distribution of not only the Abstraction Layer but also the Fusion Layer and the Application/Presentation Layer. On the other hand, if a tracked entity moves a large distance from its home location management system, provision can be made for the creation of a visitor location profile in the Abstraction, Fusion and Application Managers at the new destination.

3 Conflict Resolution

The conflict resolution of location information has to take into account differences in resolution, time of location readings, confidence in readings, and relevance of readings to a particular entity. There already exists an algorithm [8], developed at IBM, which aggregates location readings from several sensing devices. For n location sensors D_i, $1 \leq i \leq n$, location readings are of the form $R_i = (C_i, T_i, L_i)$, where C_i is the "associative confidence" of device D_i, T_i is the timestamp of the location reading, and L_i is the location reading reported by D_i. The associative confidence is a probability that the device reporting a location is actually at the same location as the user being tracked. The confidence adopted

for a given device depends upon whether it is moving or stationary, with the rationale being that a moving device generally has a higher probability of being located with its user than one that has been stationary for a long time. The algorithm sorts readings by timestamps (such that recent readings are viewed as more probable than those with older timestamps), then uses associative confidences to resolve conflicts between readings with similar timestamps.

The algorithm works reasonably well, if (i) all the location readings come from physical (position or proximity) sensors, (ii) sensor readings are frequent, and (iii) the readings are reasonably correct (i.e., the sensing device is actually located with the user). Let's assume that a user left a phone in a taxi (which is driven by somebody else), recently employed a swipe card to enter a room, and is now typing at a computer. There will be three location readings, originating from the swipe card, operating system and phone. Assuming that the timestamps of the readings are close, the associative confidence will be taken into account and the reading from the mobile phone will be erroneously selected.

It can be seen that the algorithm is not general enough to cope with physical and logical sensors and a variety of interaction types these sensors can have with a location management system. We argue that association confidence values should account for more than the mobility of the sensor. The keyboard is stationary but it is active which provides a high confidence in the reading. The phone is moving but it is not active (there were no recent calls), which should provide less confidence. Moreover, sensors like cameras or swipe cards provide location readings with some accuracy but the confidence of such readings is only high at time of the reading. To deal with these problems, our location management system defines different confidences for different types of sensors. The confidences for sensors are based on whether the sensor is (i) mobile and (ii) active (where active has a meaning depending on the sensor type).

Using timestamps as the main deciding factor in conflict resolution is also inappropriate in general location management systems. Some sensors produce notifications when a state changes (e.g. the operating system sends notifications when the user starts or stops using a keyboard), therefore the readings will be valid for a long time. In our approach, timestamps are only used to calculate confidence values for particular sensor readings and are not directly used to resolve the conflicts. A form of history is built into the algorithm, however, as sensors that are found to be producing incorrect readings are assigned a "diminished confidence" value. This implies that less weight is subsequently placed on their readings, until the readings are again found to be correct and the status is removed.

Our conflict resolution algorithm, shown in Figure 2, assumes that all location readings are mapped to a representation consisting of a set of coordinates and a resolution. The algorithm aggregates location readings into non-conflicting subsets, then performs conflict resolution amongst the subsets using confidence values. For each subset, a representative reading is selected; this is chosen as the reading with the smallest resolution, which is always fully contained in all of the other readings in the subset. Its confidence is calculated as the average of the

1. For each reading, compute its confidence as the product of the stationary/ mobile and active/non-active confidences of the corresponding sensor (unless the sensor has the "diminished confidence" value, in which case this value is adopted instead).
2. Compare each pair of location readings; if one reading is fully contained within the other, the readings are merged into a common subset for step 3.
3. For each subset of n consistent readings produced in step 2, compute a representative reading as follows:
   - The coordinates and resolution of the reading possessing the best (smallest) resolution are adopted.
   - The confidence for the representative reading is computed as the average of the confidence values for the group, adjusted according to the cardinality of the group. If the confidence values for the group are $c_1, ..., c_n$, this is:

$$1 - \frac{\sum_{i=1}^{n} 1 - c_i}{n^2}$$

4. The representative readings produced in step 3 are ranked by confidence, and the reading with the highest confidence selected as the result.
5. The "diminished confidence" value is set for sensors that produced readings conflicting with the reading chosen at 4, and is removed (if necessary) for sensors that produced consistent readings.

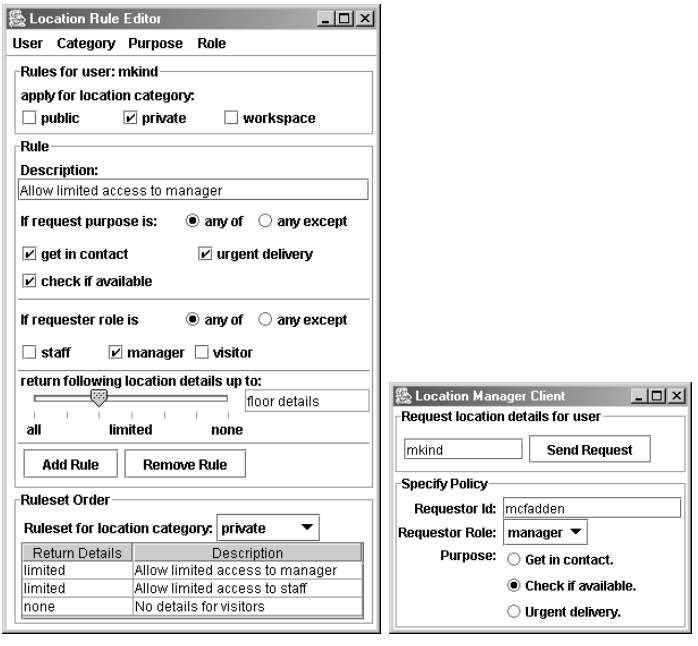
Fig. 2. Conflict resolution algorithm.

confidences for the set, adjusted to account for cardinality such that larger sets of consistent readings are assigned higher confidences. Finally, the representative readings are ordered by confidence. If there is only one subset, its representative reading is chosen as the correct reading; otherwise, the reading with the highest confidence is selected.

As the conflict resolution algorithm is applied dynamically during location queries, efficiency is crucial. Our algorithm completes in $O(n^2)$ time, where n is the number of location readings that need to be resolved. As only the most recent reading from each sensor is considered, n is bounded by the number of distinct location sensors that can be used to track a given individual, and is therefore small. When evaluated against sample data sets, the algorithm performs efficiently with high accuracy, and we aim to further improve the accuracy by exploiting historical data (past movements of the object being tracked) to resolve readings with similar confidence values.

4 Privacy

Privacy is extremely important to individuals and is reflected in the adoption of very strict privacy laws is many countries [9]. Protection of location information from unauthorized use is paramount to the success of emerging location-based services. The problem of maintaining location privacy will become greater in



(a) Location Privacy Editor (b) Query Interface

Fig. 3. User Interfaces.

future pervasive systems as the number of sensors and services that can infer a user's location increases dramatically.

Location management systems are presented with the challenge of incorporating privacy protection at the Application Layer while still allowing users to release location information to trusted service providers in a controlled and accountable fashion. As a precursor to privacy, basic security must be provided. At a minimum, location information exchange must be encrypted and service clients must be authenticated. Key policies that a location manager must address for privacy are:

- *request policy* (which allow a client to define the purposes for which requested location information will be used); and
- *access control policy* (which allow a provider to define under what context access to location information will be granted and what location detail will be provided).

There are no comprehensive solutions for location access control and privacy protection in pervasive systems as yet. The P3P initiative is a promising approach to providing privacy for users accessing web resources. P3P defines an XML based policy language allowing web services to define how potentially sensitive information collected from clients will be used, and a mechanism for user agents to retrieve these policies. The related P3P Preference Exchange Language

```
<RULESET xmlns="http://www.w3.org/2001/02/APPELv1"
         xmlns:p3p="http://www.w3.org/2000/12/P3Pv1">
  <RULE behavior="limited" description="Allow limited access to manager.">
    <p3p:POLICY><p3p:STATEMENT>
      <p3p:RECIPIENT connective="or">
        <p3p:same>manager</p3p:same></p3p:RECIPIENT>
      <p3p:PURPOSE connective="or">
        <p3p:other-purpose>get in contact</p3p:other-purpose>
        <p3p:other-purpose>urgent delivery</p3p:other-purpose>
        <p3p:other-purpose>check if available</p3p:other-purpose>
      </p3p:PURPOSE>
    </p3p:STATEMENT></p3p:POLICY>
  </RULE>
</RULESET>
```

Fig. 4. Generated APPEL Privacy Ruleset.

1.0 (APPEL1.0) [10] defines a policy language which allows users to specify privacy preferences in rule-sets. A browser or other user agent then uses the APPEL rule-sets to evaluate the acceptability of web service P3P policies.

In our location management system we use P3P (and APPEL) to provide mechanisms ensuring privacy of location information. Each location request is accompanied by a P3P policy. This policy may be static or dynamically generated by a client agent. Upon receiving a location query, the location manager evaluates the query and the accompanying P3P policy using the appropriate APPEL preferences based on the user's current context. Based on the APPEL evaluation the request is either satisfied, denied, or partially satisfied (i.e., the returned location information is generalized.) Other systems addressing privacy in the pervasive space are also based on P3P [11,12]. Our early experience with applying P3P to location information yielded mixed results:

- It was possible to use P3P to define location request policy, and APPEL rule-sets for access control.
- However, the use of generic P3P policy / XML editors [13,14] made the generation of location related policies and preferences non-intuitive and dificult for users.

To address usability issues related to the definition of privacy policy (for both request and access control) we developed domain specific user interfaces at the Application Layer that dynamically generate P3P policies and preferences.

Figure 3 (a) illustrates the user location privacy policy editor. The editor allows the definition of context sensitive privacy rules. A user may specify different privacy rules for different location zones. For example, a *public* zone would be an office lobby and a *private* zone, a restroom. For rule definition, the role of the requesting agent is considered (visitor, staff, manager). Finally, the level of location detail returned in a given situation can be adjusted. Figure 4 shows one of the compact APPEL rule-sets automatically generated from the GUI illustrated in Figure 3 (a). In the current implementation, the detail of location

```
<POLICIES xmlns="http://www.w3.org/2002/01/P3Pv1">
  <POLICY discuri="disclaimer.html" opturi="opt-in-out.html">
    <ENTITY>...</ENTITY>
    <ACCESS><nonident/></ACCESS>
    <STATEMENT>
      <PURPOSE><other-purpose>check if available</other-purpose></PURPOSE>
      <RECIPIENT><same>role=manager;user=mcfadden</same></RECIPIENT>
      <RETENTION><no-retention/></RETENTION>
    </STATEMENT>
  </POLICY>
</POLICIES>
```

Fig. 5. Generated Policy.

information returned is determined by the resultant APPEL behavior (block, limited, request) and context sensitive post-processing by the location manager. In the future this will be enhanced by defining location detail categories in the APPEL rule *persona* attribute. This will allow a more granular approach to location detail control using APPEL itself.

An application specific P3P policy generator was also implemented for a simple location client, allowing one user to query the location of another (Figure 3 (b)). The interface is simple, leaves little room for error, and dynamically generates a P3P policy (Figure 5) for each location query.

Our approach transforms a manual and cumbersome preparation of policies and preferences for location information into a dynamic generation of such policies and preferences from simple user GUIs.

5 Conclusions

The proliferation of mobile devices has created a demand for location-based services. There is a large body of research on the basic issues of location-based computing: technologies for location sensors, location determination for mobile devices, location representation formats, and early approaches to aggregation of location information (conflict resolution). In this paper we presented the architecture and functionality of a location management system, which is able to support not only location-aware applications using a limited number of sensors, but can also be used as a part of the infrastructure of large scale pervasive systems. In the latter case, the location management system feeds location information into a more comprehensive context management system. Such location management systems are very complex due to the enormous variety of physical and virtual location sensors, very large scale (in terms of sensors and entities supported by the system), high probability of conflicts in location information and complex location access and privacy rules. The location management system which we have developed and integrated with our infrastructure for pervasive systems allows either asynchronous or synchronous interactions between clients and the location management system, scales to large numbers of sensors and clients, and allows a

geographical distribution of the location management system. The algorithm for the resolution of conflicting location information developed for this system has low computational complexity and provides highly accurate results on a variety of sample location data. The system uses P3P and APPEL as mechanisms for supporting privacy of location information and provides a user friendly interface for automatic generation of privacy policies and preferences.

References

1. Beigl, M., Gellersen, H., Schmidt, A.: Mediacups: Experience with design and use of computer-augmented everyday artefacts. Computer Networks,Special Issue on Pervasive Computing **35** (2001) 401–409
2. Cheverest, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C.: Developing a context-aware electronic tourist guide: some issues and experiences. In: Proceedings of the Conference on Human Factors and Computing Systems. (2000)
3. Chen, Y., X.Chen, Ding, X., Rao, F., Liu, D.: Bluelocator: Enabling enterprise location-based services. In: Proceedings of the Third International Conference on Mobile Data Management, IEEE Computer Society (2002)
4. Henricksen, K., Indulska, J., Rakotonirainy, A.: Modeling context information in pervasive computing systems. In: Proceedings of The First International Conference on Pervasive Computing, Pervasive 2002. Volume 2414 of Lecture Notes in Computer Science., Zurich, Switzerland, Springer (2002) 169–180
5. Cranor, L., Langheinrich, M., Marchiori, M., Reagle, J.: The platform for privacy preferences 1.0 (P3P1.0) specification. `http://www.w3.org/TR/P3P/` (2002) W3C Recommendation.
6. Leonhardt, U.: Supporting Location-Awareness in Open Distributed Systems. PhD Thesis, Imperial College, London (1998)
7. Segal, B., Arnold, D., Boot, J., Henderson, M., Phelps, T.: Content based routing with elvin4. In: Proceedings of the AUUG2K Conference. (2000)
8. Myllymaki, J., Edlund, S.: Location aggregation from multiple sources. In: Proceedings of the Third International Conference on Mobile Data Management, IEEE Computer Society (2002)
9. Beinat, E.: Privacy and location-based services. Geo Informatics (2001) `http://www.geoinformatics.com/issueonline/issues/2001/09_2001/pdf_09_20%01/14_17_euro.pdf`.
10. Langheinrich, M., Cranor, L., Marchiori, M.: A P3P Preference Exchange Language (APPEL 1.0). `http://www.w3.org/TR/P3P-preferences/` (2002) W3C Working Draft.
11. Langheinrich, M.: A privacy awareness system for ubiquitous computing environments. In Borriello, G., Holmquist, L., eds.: 4th International Conference on Ubiquitous Computing (UbiComp2002). Number 2498 in LNCS, Springer-Verlag (2002) 237–245
12. Myles, G., Friday, A., Davies, N.: Preserving privacy in environments with location based applications. IEEE Pervasive Computing **2** (2003) 56–64
13. IBM Corporation: P3P Policy Editor (2002) `http://www.alphaworks.ibm.com/tech/p3peditor`.
14. Joint Research Center European Commission: JRC P3P APPEL Privacy Preference Editor (2002) `http://p3p.jrc.it/downloadP3P.php`.