A Survey of Middleware for Sensor Networks: State-of-the-Art and Future Directions

Karen Henricksen
Queensland Research Laboratory
National ICT Australia
karen.henricksen@nicta.com.au

Ricky Robinson
Queensland Research Laboratory
National ICT Australia
ricky.robinson@nicta.com.au

ABSTRACT

In future computing environments, networked sensors will play an increasingly important role in mediating between the physical and virtual worlds. However, programming sensor networks, and the applications that depend on the data they produce, is extremely challenging. The need for suitable middleware to address this problem is evident. In the last few years, various middleware solutions for sensor networks have emerged. These differ in terms of their models for querying and data aggregation, and their assumptions about the topology and other characteristics of the network. Naturally, the assumptions made for each particular middleware limit its potential applicability. Most of the current solutions provide relatively simple query abstractions, and therefore are not suitable for applications that have sophisticated requirements for processing of sensor data in the network. This paper presents a survey and analysis of the current state-of-the art in the field, highlighting the open research challenges. It also draws on the authors' experience with developing middleware for context-aware systems - that is, systems that rely on sensor-derived data to intelligently adapt their behaviour - to propose some future directions for the development of middleware for sensor networks.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

Keywords

Middleware, sensor networks, context-aware systems

1. INTRODUCTION

Deployments of sensor networks have radically increased over the past few years as wireless sensor platforms such as Berkeley motes have become widely available. This has spawned new research in the development of middleware and software engineering techniques that can simplify the programming of sensor network applications. The proposed solutions in this area facilitate high-level querying of sensor data, help to mask the distribution and heterogeneity in the sensor network, and address resource constraints by providing, for example, power-aware routing and query processing.

Much of the work in the field has taken a bottom-up approach. That is, given the available sensor network hardware, the research has looked at how to provide software engineering abstractions that assist with extracting data from the network without requiring application developers to deal with low-level hardware and networking issues. For example, several solutions have attempted to make it possible to query a sensor network in much the same way as a centralised database, without dealing with issues such as routing of the queries to appropriate nodes in the network. This paper provides a survey of the current state-of-the-art, classifying the solutions according to the types of abstraction they support. In large part, the solutions are inspired by middleware for more traditional distributed systems - for example, by work on distributed databases, tuple spaces and publish/subscribe systems.

An artefact of the current bottom-up approach is that the solutions are typically focused more on constraints of sensor network hardware than on application requirements. This implies that, while the proposed solutions may provide higher level abstractions than would be available otherwise, the abstractions may still not be suitable for many applications that could be built using sensor networks.

An alternative is the top-down approach, in which a deep understanding of application requirements is a primary driver for the design of the middleware. To some extent, this is the approach that has been taken in the field of *context-aware computing*. Context-aware systems consist of software that automatically adapts to aspects of the environment, such as the user's current location and activity, the time of day, and the presence of other people in the user's vicinity. The field of context-aware computing has been driven to a large extent by HCI researchers seeking to employ context-awareness as a technique for gathering "implicit" user input [20] from sensors, in order to reduce the amount of explicit input that is required from users. This is especially relevant for mobile/pervasive computing environments, where many tasks may simultaneously compete for the user's attention.

A commonly cited example of a context-aware application is a tourist guide that adapts its presentation of information to emphasise nearby landmarks and facilities, using a location sensor such as a GPS device to determine the user's

MidSens'06, November 27-December 1, 2006 Melbourne, Australia

current position. A more sophisticated example is a "smart home" application that assists elderly or disabled occupants of the home by monitoring their activities for unusual events such as falls, and providing reminders and assistance for tasks such as meal preparation [19]. Context-aware applications rely on sensors to observe aspects of the context. The software required to interface with the sensors and to map the sensor outputs to useful, high-level contextual information is typically very complex, so recent efforts have been made to develop middleware that handles all or part of this task. This middleware has made it possible to develop context-aware applications that build upon sensing systems that increasingly resemble sophisticated, heterogeneous sensor networks, rather than simple collections of stand-alone sensors. While almost all of the existing middleware solutions are generic enough to support a very wide variety of context-aware applications, most have been informed by application-driven requirements for sophisticated processing and interpretation of sensor data.

The authors of this paper have been involved in developing middleware for context-aware systems for several years [9, 10, 11, 13], and it is from this perspective that they analyse the current state-of-the-art in middleware for sensor networks in this paper. Although it is not the purpose of this paper to present a detailed discussion of middleware for context-aware systems (for a survey, the reader is referred to [11]), a brief discussion of the relative strengths and weaknesses of middleware for sensor networks versus middleware for context-aware systems appears in Section 3. This discussion highlights some likely future directions for middleware for sensor networks.

The paper is structured as follows. Section 2 presents an overview of current middleware solutions for sensor networks, focusing on the software engineering abstractions they provide. Section 3 provides an analysis of these solutions, highlighting a variety of open research challenges. It also proposes some future directions for the field, drawing in large part from the authors' background in the development of middleware for context-aware computing. Section 4 summarises the contributions of the paper.

2. STATE-OF-THE-ART IN MIDDLEWARE FOR SENSOR NETWORKS

The goal of middleware for (wireless) sensor networks (WSNs) is summarised as follows by Römer et al. [18]:

> The main purpose of middleware for sensor networks is to support the development, maintenance, deployment, and execution of sensing-based applications. This includes mechanisms for formulating complex high-level sensing tasks, communicating this task to the WSN, coordination of sensor nodes to split the task and distribute it to the individual sensor nodes, data fusion for merging the sensor readings of the individual sensor nodes into a high-level result, and reporting the result back to the task issuer. Moreover, appropriate abstractions and mechanisms for dealing with the heterogeneity of sensor nodes should be provided.

As the field of sensor networks is relatively new, there is no consensus yet even on fundamental design issues like the following:

- what communication paradigms are appropriate for routing information in the network; and
- what query primitives are necessary to support adequately powerful yet efficient querying of the sensor data (for example, aggregation primitives and spatial and temporal query primitives)?

Choices of routing protocols and query languages have been driven in recent solutions by strict assumptions about the requirements of the applications for which the sensor network is built, the topology and characteristics of the network, and the characteristics of the sensor nodes themselves (heterogeneity, resource poorness, etc.). Simplifying assumptions are commonly made - for example, routing issues are avoided in some solutions by assuming a single hop between the sensor node and the "sink" node that is interested in the sensor data [4], while in other solutions a fixed network topology is assumed.

According to its particular assumptions, each of the proposed middleware solutions draws on selected aspects of traditional middleware for distributed systems, such as distributed databases or publish/subscribe systems. Most solutions fit into one of the following categories:

- *database-inspired approaches*, which use SQL-like queries;
- *tuple space approaches*, which build on the tuple space abstraction made popular by Linda [7];
- *event-based approaches*, which use event correlation to aggregate sensor data; and
- *service discovery based approaches*, which use service discovery protocols to locate sensors that can meet applications' data requirements.

The remainder of this section surveys solutions in these four categories. Note that the survey does not cover "virtual machine" type solutions, such as Maté [14] and SensorWare [2], which aim purely to simplify programming without providing query processing and routing features.

2.1 Database-Inspired Solutions

The most common approach for querying sensor networks is an SQL-inspired approach. This allows a simple, declarative style of querying at the application level. Examples of solutions that adopt this approach are COUGAR [1], SINA [21] and TinyDB [16]. Some of the work in this area has been on pure sensor database systems, which essentially provide a distributed database solution appropriate for resource-constrained sensor networks, focusing on efficient query routing and processing. COUGAR and TinyDB fall into this category. SINA differs in that it uses an SQL-like language for expressing queries, but also provides other functions which are outside the scope of traditional database systems. In this section, the pure sensor database systems are presented first, and then SINA is presented for contrast.

2.1.1 COUGAR and TinyDB

The COUGAR and TinyDB sensor database systems are designed for use by relatively simple data collection applications, such as environmental monitoring applications. The main forms of data processing they support within the network are selection and aggregation based on arithmetic functions such as summation and averaging.

To some extent, both are concerned with power conservation, providing query processing strategies that aim to conserve resources. In this respect, TinyDB is more sophisticated than COUGAR - for example, it can calculate the frequency of sampling that is required to extend the battery life of a node to the requested query lifetime, and also uses a routing structure called a semantic routing tree to help sensor nodes accurately determine when queries need to be routed to their children in the routing tree.

The query languages used by the sensor database systems are extensions of SQL that support:

- temporal and data streaming concepts that allow specification of when sensor data should be sampled and for how long; and
- event-based queries - TinyDB allows queries to be triggered or terminated by events generated by other queries or by software running on a sensor node.

The following is an example TinyDB query (from [16]):

```
SELECT AVG(volume), room FROM sensors
 WHERE floor = 6
 GROUP BY room
 HAVING AVG(volume) > threshold
 SAMPLE PERIOD 30s
```

This query reports the average noise volume for rooms on the 6th floor in which the average volume for the sensors in the room exceeds a given threshold. The results of this query are delivered every 30 seconds. It is assumed that each sensor node either has hard-coded information about its position or else has positioning capabilities.

A key limitation of sensor database systems is the assumption that sensor nodes are largely homogeneous. Sensor nodes must agree in advance on the data types/relations that will be used at every node. In TinyDB, every node has an identically structured *sensors* table containing local sensor data. Each type of sensor (light, temperature, etc.) corresponds to an attribute (column) in this table. Some nodes may be missing a subset of the sensors, in which case they simply record *null* values for the corresponding attributes. This is acceptable for sensor networks in which only a small and fixed set of sensors may be present; however it is problematic for sensor networks that include many sensor types, and also networks in which graceful evolution is required. In COUGAR, it is assumed that each sensor type has a standard Abstract Data Type representation which is used at all nodes. It is not possible to insert sensing nodes with new sensing capabilities into the network in an ad hoc manner.

In addition, the sensor database systems have not been developed to support rich sensor types - for example, surveillance cameras that support sophisticated image processing functions such as face recognition. Rather, they target simple, resource-poor sensor platforms such as Berkeley motes. If some resource-rich devices were introduced as part of the sensor network, the current query processing and energy conservation strategies would most likely need to be completely reworked.

Bonnet et al. [1] point out a final limitation: sensor data capture observations, not facts. Therefore, adopting a query language that has semantics that are close to those of SQL may not be appropriate. Further research is required to address how to take uncertainty and data quality into account when formulating and processing queries on sensor networks.

2.1.2 SINA

SINA [21] can be regarded as a full middleware architecture for sensor networks, rather than a sensor database system. It provides not only support for SQL-like queries, but also for scripting using a language called SQTL (Sensor Query and Tasking Language). This language provides primitives for sensor hardware access, communication, and event handling. SQTL scripts can be sent around the network at run-time.

The SINA data model is more flexible than those of the sensor database systems described in the previous section; data is stored in an *associative spreadsheet* which uses attribute-based naming to describe cells. A small set of cells is predefined, but others can be added as required. SINA's support for changing network topologies is also more sophisticated, as it can handle problems like mobility of the querying (sink) node.

It is unclear whether SINA has been implemented.

2.2 Tuple Space Solutions

The database approaches described in the previous section provide a form of "shared memory" model, in which queries can be submitted to the sensor network as if the data was stored in a centralised repository. A similar approach, but with a different query paradigm, is provided by the TinyLIME middleware [4]. TinyLIME is based on the *tuple space* shared memory model made popular by Linda [7]. In this model, applications add and read/remove data from a common tuple space using "in" and "out" operators.

TinyLIME [4] is designed for environments in which clients typically only need to query data from local sensors. It does not provide multi-hop propagation of data through the sensor network - the only way clients can obtain data from a remote location is by obtaining it from other clients in that location. The design of TinyLIME assumes that sensor nodes are sparsely distributed, while clients move around, accessing local resources. These assumptions avoid the need for the middleware to address the complexities of query routing; however, they also severely limit the kinds of applications for which TinyLIME is suitable.

TinyLIME is an extension of the LIME (Linda In Mobile Environments) middleware developed for mobile ad hoc networks, implemented on top of TinyOS with special support for a low-power, resource constrained computing environment. The model used by both LIME and TinyLIME is that of transiently federated tuple spaces - that is, the tuple spaces of a pair of devices are temporarily federated whenever the devices are within direct networking range (i.e., a single hop) of one another.

Applications create tuple templates to subscribe for data that is of interest to them. TinyLIME subscriptions can specify the required freshness (i.e., how often data should be updated) and restrictions on values (e.g., only temperatures between 20 and 30 degrees). In order to create subscriptions, applications must know the format of tuples produced by sensor nodes. As TinyLIME is designed for motes running TinyOS, there are predefined formats for the standard sensors used by motes.

The ability of TinyLIME to support evolution in the sensor network is reasonably good (similar to that of SINA). However, as stated above, the applications for which TinyLIME is appropriate are limited. This is not a limitation of tuple space approaches in general, but of the design of TinyLIME.

However, implementing a scalable tuple space middleware capable of federating all of the connected nodes of a sensor network into a single distributed tuple space, taking into account resource considerations and mobility, would be extremely challenging. To the authors' knowledge, no implementations like this currently exist. There are, however, some implementations of publish/subscribe (event-based) systems which support a style of subscription and information delivery that has similarities with distributed tuple spaces. Event-based systems are described in the following section.

2.3 Event-Based Solutions

Advocates of event-based and publish/subscribe middleware have long argued that they are appropriate in systems in which mobility and failures are common, as they support strong decoupling of event producers and subscribers. Therefore, it is not surprising that event-based solutions are starting to be used as the basis for middleware for sensor networks. However, work in this area remains immature. Yoneki and Bacon [23] have produced a reasonably sophisticated set of event operators for describing event patterns in sensor networks. The main distinction between the event description language and the subscription languages used in previous publish/subscribe systems is that it supports not only standard operators, including conjunction, disjunction, negation, concatenation and iteration, but also spatial and temporal restrictions. This allows subscriptions such as the following (from [23]):

- $(T_{room1} + T_{room7})_{30}^{AVG}$: The average temperature for rooms 1 and 7 over 30 minutes.

A crucial limitation of this solution is the complexity that is necessarily involved in implementing it. In order to support the temporal operators, a complex timestamping scheme is required; similarly, the spatial restriction operator requires every sensor node to have accurate positioning information, which is not realistic for many sensor networks. Yoneki and Bacon report that they are working on a complete implementation, but had no implementation results to report yet in [23]. Further, the solution makes an assumption that all information reported by sensors is accurate, and that all sensors that are of the same type are interchangeable from the application's perspective (i.e., they all provide adequate levels of quality and identical data formats). Specification of the required sensor data types uses only a very simple naming scheme (for example, "T" to represent temperature in the above example).

Some work on handling uncertainty of events in sensor networks has been done by Li et al. [15] in their work on DSWare (Data Service Middleware). They introduce the notion of confidence when looking at event correlations. For a compound event made up of several sub-events, they propose using a confidence function to determine the likelihood of the compound event, according to how many of the sub-events have occurred. Li et al. also discuss reducing the confidence of events based on age to address staleness. However, they provide little detail about the scheme beyond examples, and it is unclear whether it has been implemented.

In contrast, the Mires middleware [22] is a more pragmatic publish/subscribe solution that has been designed and implemented to run on TinyOS. TinyOS provides built-in support for event handling and a message-oriented communication paradigm (Active Messages), both of which, Souto et al. argue, provide a strong basis for implementing a publish/subscribe middleware. Mires provides an architecture that allows: sensor nodes to advertise the types of sensor data they can provide; client applications to select from the advertised services; and sensor nodes to publish their data to clients in accordance with their subscriptions. Mires differs from the work discussed above in that it focuses on architectural and networking issues, rather than on subscription semantics. No information is provided by Souto et al. on the expressiveness of the Mires subscription language, so it is probable that it is either very simple or similar to the subscription languages of publish/subscribe systems used in traditional distributed computing applications.

2.4 Service Discovery Based Approaches

The MiLAN middleware [8] takes a different approach to the previously discussed solutions in that it builds on existing networking and service discovery protocols, using a plug-in mechanism to incorporate arbitrary protocols. Applications specify their sensing requirements to the middleware through a standard API, in terms of graphs describing sensor quality of service (QoS) and state-based *variable* requirements. Variables are the means used by applications to describe the types of sensor data they require. The use of a state-based variable graph means that applications can specify which subset of the variables is required in each application state (and with what QoS). The middleware uses the graphs provided by the application, together with information about the current application state, to decide how to configure the network and sensors to meet the application's requirements. Matching of sensors to variables is based on the use of a service discovery protocol to identify which sensors are available at any point in time. This approach addresses failure and evolution in the sensor network. Central considerations in sensor matching are cost (in terms of power and other resources) and QoS. Variables can be bound to *virtual sensors* that combine data from two or more sensors to provide data with better quality than a single sensor alone.

One shortcoming is that MiLAN relies on existing service discovery protocols, most of which are not suitable for use in resource-poor sensor networks. This includes the two service discovery protocols mentioned by Heinzelman et al., SDP and SLP. MiLAN appears to target a different class of sensor network (i.e., one that is richer in resources and closer to traditional heterogeneous distributed systems) than the previously described solutions.

MiLAN does not appear to have been implemented[1].

3. ANALYSIS AND FUTURE DIRECTIONS

Sensor networks clearly have the potential to revolutionise a number of industries: medicine, field biology, agriculture and defence among others. They may even change the way humans interact with the objects around them. However, before these goals can be achieved, there are many challenges that must be overcome. This section gives an overview of what the authors believe are the most important challenges in the area of middleware for sensor networks, and suggests the directions in which research in this area should proceed. Specifically, the authors argue for a convergence of middle-

[1]Heinzelman et al. stated in a 2004 publication [8] that the middleware was then under development; more recent results do not appear to be publicly available.

ware approaches in sensor networks and context-aware systems as a means to bridge the gap between the sophisticated high-level programming and query abstractions expected by context-aware applications and the simpler abstractions provided by present-day middleware for sensor networks.

A major assumption inherent in most of the middleware solutions discussed in Section 2 is that the sensors nodes in a sensor network are both resource constrained and homogeneous. This assumption is too restrictive in light of sensor network-based applications envisioned for the future. There is often a great disparity in the processing and communication capabilities of the sensor nodes required for these applications. New middleware solutions for sensor networks must be more generic and assume heterogeneous sensor hardware and diverse communication mechanisms. Already, platforms that make use of a wide range of sensors, from cameras to temperature sensors, are appearing in the field of context-aware systems [3].

In general, middleware for context-aware systems supports more sophisticated queries and data fusion techniques than sensor network middleware. With improving facial recognition techniques, for example, context-aware systems can answer queries such as "who is in the current frame?" Increasingly, they also support advanced reasoning with the aid of ontologies. Such reasoning allows new information to be abstracted from explicitly gathered information. For example, given an appropriate set of rules or machine learning algorithms, it is possible to derive, with some degree of accuracy, a person's current activity from the current time and his/her location. Middleware solutions for sensor networks, on the other hand, typically provide simpler kinds of queries and trivial data aggregation (min, max, sum, average, etc.).

Related to the use of ontologies in middleware solutions for context-aware systems is their alignment with standards. Ontology-based approaches to context-awareness build upon W3C standards such as XML, RDF and OWL, providing a platform for interoperability and a formal semantics for reasoning. In contrast, most middleware solutions for sensor networks currently are not aligned with any standards. However, the Sensor Web Enablement initiative within the Open Geospatial Consortium (OGC) is developing XML-based standards that may be used to help bridge the gap between context-aware systems and sensor networks. These standards include the Sensor Model Language (SensorML) and Observations & Measurements (O&M)[2].

There may be benefits in attempting to layer the context reasoning capabilities [17, 6] and programming abstractions/toolkits [10] of context-aware systems over the top of middleware for sensor networks. Many middleware solutions for context-aware systems currently require custom software components to be written in order to interface with each new sensor type (called widgets in the Context Toolkit [5] and receptors in the middleware of Henricksen et al. [10]). They also do not provide query routing/processing solutions suitable for extracting data from highly resource-constrained sensor nodes. Thus, layering the solutions would not only confer advantages from the research on context-aware systems to middleware for sensor networks; it would also transfer benefits in the other direction.

There are several challenges involved in harmonising middleware for sensor networks with middleware for context-aware systems. One of the crucial challenges which has already been alluded to lies in harmonising the information modelling and query abstractions. Context-aware systems involve high-level information models [10], which, as mentioned, are often aligned with standards such as OWL or RDF. Querying in these systems involves extracting information from these models and/or carrying out reasoning on the information. In contrast, current sensor network middleware solutions support simpler query mechanisms, such as SQL-like queries. One route towards harmonisation is to incorporate mark-up of sensor data and/or query results using XML formats such as O&M. However, semantic matching would still be required between high-level concepts and sensor data (e.g., between concepts such as activity and sensor observations such as movement or pressure). Indulska et al. [12] have carried out some preliminary work in this area, but further research is required.

To better support reasoning about, and use of, sensor data, improved support for information quality in sensor network middleware is required. Solutions for modelling and reasoning about information in context-aware systems assume the existence of quality metadata that characterises sensor-derived information with respect to relevant parameters such as certainty and freshness [10]. Harmonising middleware for sensor networks with middleware for context-aware systems therefore implies incorporating the notion of quality within the former (in the query model, in particular).

Work also needs to be done on supporting more advanced types of data aggregation and fusion within sensor networks. For most purposes, it will not be sufficient to adopt a solution in which data from a sensor network is aggregated in a repository located on a single sink node at which higher-level interpretation and reasoning are carried out. Data fusion, interpretation and reasoning will be required at multiple locations in the network. Query sophistication, rather than resource constraints, will become the key design consideration for sensor networks as the hardware capacity of sensor nodes increases. In future, to aid in comparing the sophistication of query support in various middleware solutions, it will be useful to develop query benchmarks for evaluating the solutions (similar to performance benchmarks already in use). These could consist of representative queries from a variety of application domains.

4. CONCLUSIONS

This paper has presented an overview of the current state-of-the-art in middleware for sensor networks, covering database-inspired, tuple space, event-based and service discovery based approaches. These solutions have several shortcomings. They offer relatively simple query abstractions, and most provide only primitive data fusion mechanisms within the network. They also largely ignore issues related to uncertainty and information quality. Finally, most solutions assume that sensor nodes are homogeneous and resource constrained, some solutions make strict assumptions about network topology, and several of the more sophisticated proposals, such as SINA and MiLAN, do not appear to have been implemented.

A significant body of related research has already been carried out in the area of context-aware systems in relation to modelling and reasoning about sensor-derived data, modelling of information quality, and detecting and resolving conflicting information. Work has also been done on developing appropriate high-level programming abstractions and

[2]http://www.opengeospatial.org/pt/06-046r2

toolkits to simplify the design and implementation of applications that rely on the sensor data. One of the contributions of this paper has been a discussion of how the work on middleware for sensor networks can be combined with research on context-aware systems, in order to leverage the relative strengths of the work in the two areas.

Acknowledgements

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology, and the Arts; the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs; and the Queensland Government.

5. REFERENCES

[1] P. Bonnet, J. E. Gehrke, and P. Seshadri. Towards sensor database systems. In *2nd International Conference on Mobile Data Management (MDM)*, volume 1987 of *Lecture Notes in Computer Science*, pages 3–14. Springer, 2001.

[2] A. Boulis, C.-C. Han, and M. B. Srivastava. Design and implementation of a framework for efficient and programmable sensor networks. In *International Conference on Mobile Systems, Applications and Services (MobiSys)*, pages 187–200. ACM Press, 2003.

[3] G. Chen and D. Kotz. Solar: An open platform for context-aware mobile applications. In *1st International Conference on Pervasive Computing (Pervasive), Short Paper Proceedings*, pages 41–47, Zurich, August 2002.

[4] C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. L. Murphy, and G. P. Picco. TinyLIME: Bridging mobile and sensor networks through middleware. In *3rd IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 61–72. IEEE Computer Society, March 2005.

[5] A. K. Dey, D. Salber, and G. D. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16(2-4):97–166, 2001.

[6] F. L. Gandon and N. M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *Web Semantics Journal*, 1(3), 2004.

[7] D. Gelernter. Generative communication in Linda. *ACM Computing Surveys*, 7(1):80–112, January 1985.

[8] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo. Middleware to support sensor network applications. *IEEE Network*, 18(1):6–14, January/February 2004.

[9] K. Henricksen and J. Indulska. A software engineering framework for context-aware pervasive computing. In *2nd IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 77–86. IEEE Computer Society, March 2004.

[10] K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: Models and approach. *Journal of Pervasive and Mobile Computing*, 2(1):37–64, Feburary 2006.

[11] K. Henricksen, J. Indulska, T. McFadden, and S. Balasubramaniam. Middleware for distributed context-aware systems. In *International Symposium on Distributed Objects and Applications (DOA)*, volume 3760 of *Lecture Notes in Computer Science*, pages 846–863. Springer, 2005.

[12] J. Indulska, K. Henricksen, and P. Hu. Towards a standards-based autonomic context management system. In *3rd International Conference on Autonomic and Trusted Computing (ATC)*, volume 4158 of *Lecture Notes in Computer Science*. Springer, September 2006.

[13] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen. Experiences in using CC/PP in context-aware systems. In *4th International Conference on Mobile Data Management (MDM)*, volume 2574 of *Lecture Notes in Computer Science*, pages 247–261. Springer, January 2003.

[14] P. Levis and D. Culler. Maté: A tiny virtual machine for sensor networks. In *10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, San Jose, October 2002.

[15] S. Li, Y. Lin, S. H. Son, J. A. Stankovic, and Y. Wei. Event detection services using data service middleware in distributed sensor networks. *Telecommunication Systems*, 26(2-4):351–368, June 2004.

[16] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems*, 30(1):122–173, 2005.

[17] A. Ranganathan and R. H. Campbell. An infrastructure for context-awareness based on first-order logic. *Personal and Ubiquitous Computing*, 7(6):353–364, December 2003.

[18] K. Römer, O. Kasten, and F. Mattern. Middleware challenges for wireless sensor networks. *ACM Mobile Computing and Communications Review (MC2R)*, 6(4):59–61, October 2002.

[19] J. Russo, A. Sukojo, A. S. Helal, R. Davenport, and W. C. Mann. SmartWave - intelligent meal preparation system to help older people live independently. In *2nd International Conference on Smart Homes and Health Telematics (ICOST)*, volume 14 of *Assistive Technology Research Series*, pages 122–135. IOS Press, September 2004.

[20] A. Schmidt. Implicit human computer interaction through context. *Personal Technologies*, 4(2&3):191–199, June 2000.

[21] C.-C. Shen, C. Srisathapornphat, and C. Jaikaeo. Sensor information networking architecture and applications. *IEEE Personal Communications*, 8(4):52–59, August 2001.

[22] E. Souto, G. Guimãraes, G. Vasconcelos, M. Vieira, N. Rosa, C. Ferraz, and J. Kelner. Mires: a publish/subscribe middleware for sensor networks. *Personal and Ubiquitous Computing*, 10(1):37–44, February 2006.

[23] E. Yoneki and J. Bacon. Unified semantics for event correlation over time and space in hybrid network environments. In *IFIP International Conference on Cooperative Information Systems (CoopIS)*, volume 3760 of *Lecture Notes in Computer Science*, pages 366–384. Springer, 2005.