Mobile Sensing Towards Context Awareness

Susana Bulas Cruz
Instituto de Telecomunicações
Faculdade de Engenharia da Universidade do Porto
Rua Dr. Roberto Frias, s/n 4200-465 Porto, Portugal
susana.bulas.cruz@fe.up.pt

Abstract

In recent years, mobile technologies suffered a great development and became increasingly widespread, creating new opportunities and challenges to a large variety of areas, namely mobile sensing. Mobile sensing is an interesting and recent research topic, as nowadays mobile devices are used not only for communicating, but also for accessing the Internet, purchasing goods, exchanging information, and organizing our everyday life. In addition to continuously providing connectivity through multiple wireless interfaces (e.g. 3G, Wi-Fi, and Bluetooth), these devices also integrate various types of sensors, from microphones and cameras to accelerometers and positioning systems (GPS), allowing to capture useful real-time information about the user's context. Therefore, the main goal of this work is to use that data and develop advanced algorithms for context estimation, inference and prediction, in order to provide the user with content tailored to his context, current activity, and preferences.

1. Introduction

Mobile device users want to be able to access and manipulate information and services specific to their location, time, and environment. To provide the users with adequate applications and services, the devices should be aware and automatically adapt to their changing contexts, what is called context-awareness [10]. It is important to formally define context in mobile computing. According to Dey and Abowd, "context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves" [4]. Context usually consists at each time instant of multiple partial descriptions of a situation, called context atoms. Although context atoms may already be useful without any further processing, it is possible to use many context atoms together, at a certain time instant, to form a single description of a higher level context, e.g. an event [9].

As referred to in [5], there are great challenges in building context-aware systems, such as developing accurate sensors to acquire information from the physical environment, building software infrastructure and tools to interpret and process the sensed information, creating data management systems to manage and store contextual data for later retrieval, and developing frameworks to address security and privacy issues associated with the context-aware systems. In this paper we will address some of those topics, presenting proposed solutions discussed in the literature.

The rest of the paper is organized as follows. Section 2 characterizes sources for context-awareness, namely on current mobile devices. In Section 3, we discuss the architecture of context-aware systems, presenting a general framework. Section 4 addresses decentralized inference and compares centralized and distributed learning. Section 5 describes applications in context-aware telecommunication services. We conclude in Section 6 with some directions for future work.

2. Source Models for Context Awareness

The information we require to infer context can be obtained from a variety of sources. Some examples are hardware sensors, network information, device status, browsing user profiles [2]. In [7], context instances are divided in physical and logical context (some authors also call them external and internal dimensions [2]). The first refers to context that can be measured by hardware sensors, such as location, light, sound, movement, touch, temperature, and air pressure. The logical context is mostly characterized by the user or obtained by monitoring user interactions, namely the user's goals, tasks, work context, business processes, and emotional state [2].

Nowadays there exists a wide variety of hardware sensors capable of capturing almost any physical data. In this

work, we denote by sensor not only these hardware devices, but also any data source that can provide useful context information. Using the classification in [8], sensors can be divided in three groups.

- **Physical sensors** are the most commonly used type. Table 2 presents some available sensors of this type for sensing different contexts.
- **Virtual sensors** are software applications or services monitoring events in virtual space that provide context information.
- **Logical sensors** combine information from a physical or virtual sensor with additional information from some other source, such as a database, in order to solve higher tasks.

Type of context	Available sensors
Light	Photodiodes, color sensors, IR and UV sensors etc.
Visual context	Various cameras
Audio	Microphones
Temperature	Thermometers
Motion, acceleration	Mercury switches, angular sensors, accelerometers, motion detectors, magnetic fields
Location	Outdoor: Global Positioning System (GPS), Global System for Mobile Communications (GSM); Indoor: Active Badge system, etc.
Touch	Touch sensors implemented in mobile devices
Physical attributes	Biosensors to measure skin resistance, blood pressure

Table 1. Commonly used physical sensors [2]

Mobile devices can acquire context information from many possible sources. Current smart phones incorporate assisted GPS, digital compass, three-axis gyroscope, accelerometer, proximity sensor, ambient light sensor, camera and microphones. In addition to this large number of physical sensors that has been successfully integrated in mobile devices, they offer a lot of applications and services that can be used to extract useful context information. Mobile devices' internal information, such as applications currently running, internal device processes, and time, are some examples. They provide explicit information on device application use and users' scheduled tasks, preferences, and social networks. This cognitive context information is key towards providing the user with personalized context-aware computing services. Time information is used to associate certain events with others and form event sequences that might relate to a current higher-level context or predict a future event. Also, mobile devices that support various wireless networks (e.g. GSM and GPRS), and short-range ad hoc networks (e.g. Bluetooth), provide continuous and ad hoc connectivity to local and remote data [10].

The objective is to infer more context information from a group of sensors than the sum of context derived from individual sensors. However, there are some challenges as the data supplied by diverse sensors can be very different, ranging from slow sensors that supply scalars (e.g. temperature) to fast and complex sensors that provide larger volume data (e.g. camera). The update time can also vary significantly from sensor to sensor [14]. Moreover, the gathered information has to be reliable, even if it comes from multiple heterogeneous sources, which acquire rapidly changing data with a level of uncertainty. Combining data from several sensors to extract relevant context information is challenging due to some data acquisition degradation factors, such as noise, faulty connections, drift, and miscalibration [10]. Although context sensed from different sensors may conflict to each other due to all those factors, it is possible to use sensor-fusion techniques to solve such conflicts and improve context accuracy and completeness [4]. Yet, inferred contexts yield only partially reliable approximations. Furthermore, this large amount of available data and data sources requires not only to integrate them, but also to filter the relevant parts of data. Providing the appropriately tailored set of data is important in order to meet the devices constraints on resources (power supply, memory and computing power) and operate on a manageable amount of data [3].

3. Context-Aware Systems Architecture

In [15] the authors present a literature review and a classification of context-aware systems from 2000 to 2007. A general abstract layer architecture of context-aware systems is presented, consisting in four layers:

1. **Network Layer** involves a network supporting context-aware systems and sensor collecting low-level of context information;
2. **Middleware Layer** manages processes and stores context information;
3. **Application Layer** provides users with appropriate service;
4. **User Infrastructure Layer** manages the interface of context-aware systems to offer suitable interface to users.

The classification framework, illustrated in Fig. 1, developed for classifying the literature related with context-

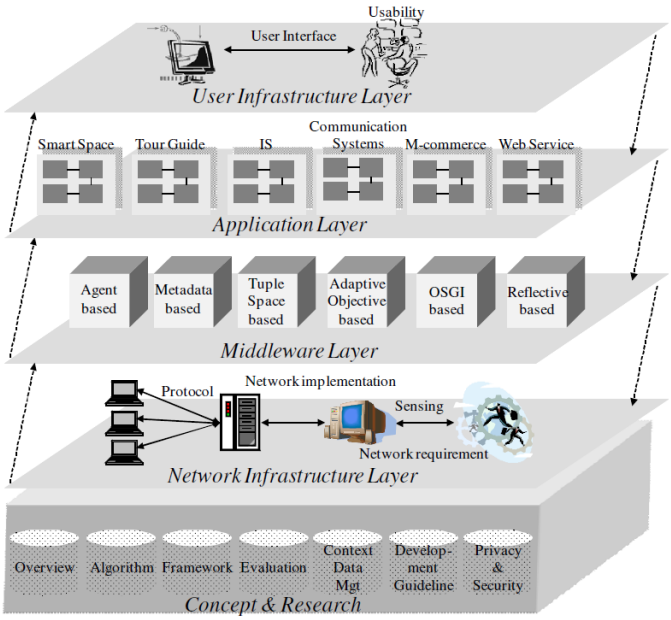

Figure 1. Classification framework [15].

aware systems is based on this abstract architecture. It consists of the following five layers: concept and research layer, network layer, middleware layer, application layer and user infrastructure layer. Each layer has detailed categories for dividing the literature. Concept and research layer, which consists in theories and foundation to construct context-aware systems, includes overview, algorithm, development guideline, framework, context data management, evaluation and privacy and security categories. Network layer involves protocol, sensing, network requirement and network implementation. Middleware layer is categorized into agent-based middleware, metadata based middleware, tuple space based middleware, OSGI based middleware, reflective middleware and sensor selection middleware. Application and service layer is divided in smart space, tour guide, information systems, communication systems, M-commerce and web services. User infrastructure layer consists in interface and usability categories.

An alternative framework devised from the common architecture in various design approaches in presented in [2]. This conceptual framework is divided in the following five layers: sensors, raw data retrieval, preprocessing, storage and management, and application.

The first layer deals with the different sensors that are sources for context awareness, already discussed in Section 2. The second layer, responsible for the retrieval of raw context data, makes the low-level details of hardware access transparent by providing abstract methods. For physical sensors it uses appropriate drivers. For virtual and logical sensors employs APIs. Thus, by using interfaces, it is possible to exchange components responsible for equal types of context with only minor adjustments in the current and upper layers. The preprocessing layer involves the interpretation of contextual information, raising the results of the previous layer to a higher abstraction level. The sensors frequently return technical data that is not suitable to use by application designers. This layer is not implemented in every context-aware system but may offer useful information if the raw data is too coarse grained. The transformations include extraction and quantization operations. In context-aware systems consisting of diverse context data sources, an aggregation (also called composition) process can be carried out in this layer. It consists in combining different context atoms to obtain high-level information. To make this analysis work correctly, several statistical methods are employed and often a training phase is necessary. The problem of sensing conflicts that might occur when using several data sources has to be solved in this layer as well. Store and management, the fourth layer, organizes the gathered data and, via a public interface, makes the data available to the client, either in a synchronous or asynchronous way. In the synchronous mode the client is polling the server for changes via remote method calls, i.e. it sends a message requesting some sort of accessible data and pauses until it receives the server's answer. The asynchronous approach works via subscriptions, i.e. each client subscribes to specific events it is interested in, and on occurrence of one of them, the client is either simply notified or a client's method is directly involved using a call back. In most situations the asynchronous mode is more appropriate as a result of the rapid changes in the underlying context. The application layer implements the actual reaction on different events and context-instances, realizing the client.

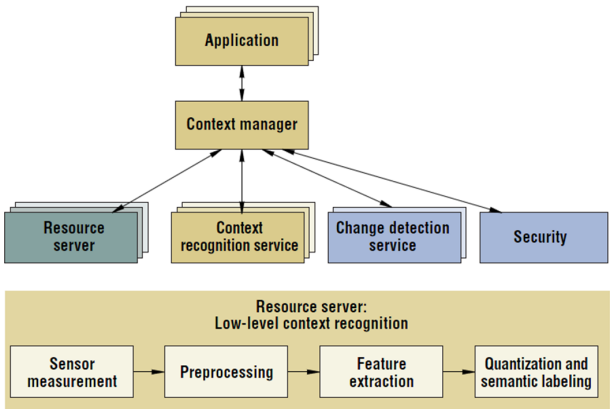

Figure 2. Entities and information flow in context framework [15].

An example of a framework for context management in mobile devices is presented in [10] and illustrated in Fig. 2. It contains four main functional entities: context manager, resource server, context recognition service, and application. When they communicate, the context manager works as a central server while the remaining entities, with the ex-

ception of security, act as clients, using services provided by the server. The context manager, resource servers, and applications run on the mobile device itself, and the services are either distributed or local.

Concerning the architecture, [5] distinguishes three approaches in how to acquire context information. The first one is direct sensor access, often applied in devices with sensors locally build in. However, this method is not adequate for distributed systems since it is not able to manage multiple concurrent sensor accesses. The second approach, middleware infrastructure, hides low-level sensing details by introducing a layered architecture to context-aware systems. The last approach, the context server, allows multiple clients access to remote data sources. It extends the middleware based architecture by introducing an access managing remote component. This distributed approach moves the complex operations of gathering sensor data to the context server, and consequently unburden the end devices, which usually have limited resources (e.g. energy, computation power, memory). However, when designing a context-aware system based on client-server architecture, there are important factors that have to be considered, such as appropriate protocols, network performance, and quality of service.

3.1. Mobile Data Gathering

One of the challenges for context-aware applications is to gather information on their current situation. The gathering process includes the discovery of context sources (peer devices, sensors or network services), matching of context sources to system needs, and subsequent retrieval of context data [6]. Context-aware applications require connecting and transferring data using heterogeneous devices, managing different network interfaces, and dealing with the various data formats used by different context sources. The communication process of establishing a connection for data transfer has to include functionality to transport data, manage addressing, and support various message protocols. As already mentioned above, the data transport can be either synchronous or asynchronous. Synchronous transport mechanisms deliver data immediately to its recipient, i.e. do not store and forward, and some examples include distributed object technologies and streaming middlewares. Asynchronous transport mechanisms deliver messages at irregular intervals and are often forwarded with a time delay. Examples include event-based middlewares and other message passing schemes. There are also many challenges related to the task of managing the reception of data, namely providing a uniform interface to receive multiple context types (from different context source), eliminating irrelevant sources by semantically making correspondence between the types of context required by the application and the context types provided by the sensors, deciding which set of context providers to use, deciding on the need for data acquisition, trading off the cost of communicating with context providers against the cost of inferring context, and deciding on the frequency of data acquisition [6].

Considering existing protocols for data-gathering in general sensor networks, we can classify them by the way they organize the sensor nodes: hierarchically and non- hierarchically. The non-hierarchical protocols include directed diffusion, flooding, gossiping, and SPIN (Sensor Protocols for Information via Negotiation). Examples of hierarchical protocols include LEACH (Low-Energy Adaptive Clustering Hierarchy) and PEGASIS (Power-Efficient Gathering in Sensor Information Systems). For more details see [1] and references therein.

3.2. Context Models

Three context management modes for coordinating multiple processes and components are presented in [2]. The first ones, widgets, present a process-centric view and are efficient, yet not robust to component failures. Network services, which resemble context server architecture and consist in a service oriented model, are not as efficient as widgets but provide robustness. The last one, called blackboard model, provides a data centric view and requires a centralized server to host the blackboard.

To manage context data, the framework entities must have a common structure for representing information in a machine processable form. In [13], the most relevant approaches to modeling context are discussed concerning the following factors: distributed composition, partial validation, richness and quality of information, incompleteness and ambiguity, level of formality, and applicability to existing environments. The addressed context models are key-value models, markup scheme models, graphical models, object oriented models, logic based models, and ontology based models. For the complete analysis please refer to [13]. In this work, we will focus on ontology based models as they fulfill most of the mentioned requirements, indicating that they are the most expressive approaches to model context.

The authors in [9] developed an ontology for mobile device sensor-based context awareness. The design was based on the following principles: domain, simplicity, practical access, flexibility and expandability, facilitate inference, genericity, efficiency, and expressiveness. Concerning the structure of the ontology, each context is described using seven properties: context type (context category), context (symbolic value of context type), value (raw value of context type), confidence (uncertainty measure), source, timestamp, and free attributes (additional properties about details that are not included in the other properties). Any context expression must contain at least context type and either context or value. There is an example of a sensor-based context

ontology vocabulary in Table 3.2. In this example, context is divided in low-level context and high-level context, according to levels of abstraction. The most low-level context is the raw data collected directly from the sensors. High-level context is inferred from low-level context, applying algorithms like Bayesian networks, probabilistic logic, fuzzy logic, decision trees, neural networks and support vector machines [15].

Context Type	Context Value
Environment:Sound:Intensity	{Silent, Moderate, Loud}
Environment:Light:Intensity	{Dark, Normal, Bright}
Environment:Light:Type	{Artificial, Natural}
Environment:Light:SourceFrequency	{50Hz, 60Hz, NotAvailable}
Environment:Temperature	{Cold, Normal, Hot}
Environment:Humidity	{Dry, Normal, Humid}
User:Activity:PeriodicMovement	{FrequencyOfWalking, FrequencyOfRunning, NotAvailable}
Device:Activity:Stability	{Unstable, Stable}
Device:Activity:Placement	{AtHand, NotAtHand}
Device:Activity:Position	{DisplayDown, DisplayUp, AntennaDown, AntennaUp, SidewaysRight, SidewaysLeft}
Context Type (higher-level)	**Context Value**
Environment:Location:Building	{Indoors, Outdoors}

Table 2. Sensor-based context ontology vocabulary example [10]

4. Decentralized Inference

Wireless sensor networks (WSN) are designed to make inferences from the environments they are sensing. However, they are usually characterized by limited communication capabilities due to tight energy and bandwidth constraints, which restrict the sensors ability to transmit data to each other or to a fusion center for centralized processing.

Decentralized inference has often been considered in the framework of parametric models, in which the statistics of phenomena under observation are assumed known to the system designer. However, in a WSN scenario one cannot expect that the necessary data or domain knowledge will always be available to support a parametric approach. Consequently, applications of WSN provide an especially strong motivation for the study of nonparametric methods for decentralized inference [11]. We will discuss these methods, focusing on how is distributed learning in WSN different from centralized learning and what fundamental limits on learning are imposed by constraints on energy and bandwidth.

4.1. Centralized versus Distributed Learning

To separate the low-level sensor data processing from high-level applications, a middleware layer is required to collect raw sensor information, translate it to an application-understandable format, and disseminate it to interested applications [4]. There are typically two approaches, centralized and distributed.

The simplest way to decouple is to use a centralized context server, which provides contextual information to the applications. Centralized learning systems are characterized by moving the data (raw data) to a central location for model building, making the training data entirely available to a single processor. Nonparametric methods studied within machine learning, such as kernel methods, neural networks, nearest-neighbor rules, decision-trees, Bayesian and Markov networks, and boosting, have demonstrated widespread empirical success in many centralized applications. Yet, in WSN, since the aforementioned constraints prevent the necessary communication for enabling the access to the entire data set, many centralized learning strategies are unfeasible. For details on these methods, please refer to [11] and the references therein. Also, the centralized approaches usually have scalability problems.

Instead of maintaining all context information in one centralized place, a distributed architecture allows context be held at several places to avoid potential bottleneck [4]. Distributed inference in sensor networks under communication constraints is a topic of much current interest. One approach to extending classical learning rules to distributed learning, and in particular to wireless sensor networks, focuses on developing communication-efficient training algorithms [12]. In [11], ensemble methods, parallel learning, and data mining, are pointed out as learning algorithms that could possibly be relevant in giving some insights to developing distributed learning efficient algorithms.

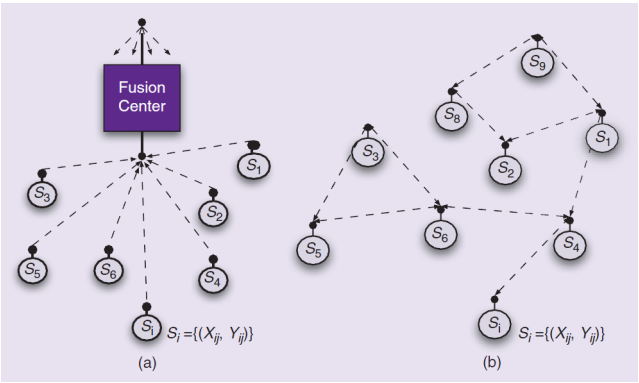

Figure 3. (a) A parallel network with a fusion center, and (b) an ad hoc network with in-network processing, [11].

In [11], the authors decompose the literature on distributed learning according to two general research themes (see Fig. 3): distributed learning in WSNs with a fusion center, where the focus is on how learning is effected when communication constraints limit access to training data; and distributed learning in WSNs with in-network processing, where the focus is on how inter-sensor communications and local processing may be exploited to enable communication-efficient collaborative learning. In the first approach, sensors would be divided into subgroups with a

”cluster head” each, responsible for collecting the data from its group. As the sensors within a cluster are close to each other and communication over short distances can be done efficiently, this may be inexpensive. Also, only a small subset of nodes is chosen to send data to the fusion center. If the nodes are wisely selected, the learning performance remains high while keeping communication costs low enough. In the second approach, the authors consider how in-network processing and local inter-sensor communication may enable communication-efficient collaborative learning. The links are usually assumed to support the exchange of simple real-valued messages, where simplicity is defined relative to the application (e.g., sensors communicate summary statistics rather than entire data sets). The key intuition is that local communication is more efficient since it requires less energy and bandwidth than communicating globally. The correlation structure of the phenomenon under observation (e.g., a temperature field) can often be represented using a graphical model (e.g., Markov networks) and since inter-sensor communications are envisioned to occur over similar graphical structures, message-passing algorithms associated with graphical models may be applied to address distributed learning in WSN. This approach is efficient due to its exploitation of local communication.

5. Context-Aware Applications

In 2000, [4] made a survey on previous research work about context-awareness focusing on applications, and describing a significant number of them. They concluded that until that moment, most of research on modeling context was limited to location information. In fact, location as position or area in space is a context of particular importance, and has received more attention in mobile computing than any other type of context [14]. However, there are examples of applications considering other types of context.

In [4], context is divided in the following four categories:

- **Computing context** includes network connectivity, communication costs, communication bandwidth, and nearby resources (e.g. printers, displays, and workstations).
- **User context** includes the user’s profile, location, people nearby, the current social situation.
- **Physical context** includes lighting, noise levels, traffic conditions, and temperature.
- **Time context** includes time of a day, week, month, and season of the year.

The computing, user and physical contexts can be recorded across a time span and used to form a context history, which is very useful for certain applications. For example, if we know the user’s calendar and the current location and time, the application may have an accurate idea of the user’s social situation, such as having a meeting or a class.

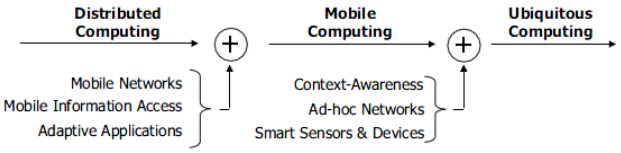

Figure 4. Evolution chain of distributed systems [13].

In [15], a large list of references (from 2000 to 2007) concerning applications and services in context-aware systems is divided in the following categories: smart space (home, hospital and classroom), tour guide, information systems, communication systems, M-commerce, and web-service. They conclude, however, that the scope of applications and services in most articles is limited to small regions (laboratory, school, hospital, smart room, etc.), and strategic alternatives or business models for gaining the revenue by using context-aware systems are very few. Nevertheless, they find context awareness to be a key factor for new applications in the area of ubiquitous computing (see Fig. 4). The authors also present the main characteristics of context-aware applications and services, based on their analysis of the literature review related with this topic.

- Context-aware applications are customizable and exploit user context and preferences. They compute devices to operate independently of human control with automatic execution within a certain context, in order to provide the user with services autonomously.
- Context-aware applications not only handle current task, situation and action but also predict future behavior, acting in anticipation of future goals or problems.
- In context-aware applications several devices are interconnected, recognizing each other on a certain distance. The distance between device and user is an important factor to offer appropriate service to the user.
- Context-aware applications provide a rich set of capabilities and services to the nomad, who moves from place to place in a transparent way.

One essential factor about context-aware applications is that traditional concerns regarding security and privacy are amplified [6]. They imply access to a wide range of sensitive data, and involve ad-hoc collaborations between unknown entities. When support for security or privacy was addressed in context-aware infrastructures, it focused on confidentiality, trust and identity. As privacy is pointed out

as the major criticism of ubiquitous computing, some authors argue that support for privacy constraints must be provided by context-aware infrastructures and not applications [6].

Concerning telecommunication services, it has been shown that context can be applied for different tasks within a mobile device. In overall device operation, context can for example be exploited for context-sensitive resource and power management. In applications run on mobile devices, context might be applied to enable adaptive applications and explicitly context-based services. In the mobile device user interface, context can be used to facilitate a shift from explicit user-driven to implicit context-driven interaction [14].

6. Conclusions

With the strong penetration of mobile devices such as notebooks, PDAs, and smart phones, ubiquitous systems are recently becoming more and more popular. By gathering context data, these systems (context-aware systems) offer innovative opportunities for application developers and for end users. They allow the adaptation of systems behavior according to the user current context, activity, and preferences. Particularly in combination with mobile devices, these mechanisms for context estimation, inference and prediction are of high value and can be used to increase usability tremendously.

For this purpose, we intend to build software modules capable of (a) classifying the information flows that reach the server, (b) selecting the most relevant information sources according to the service requirements in an automatic fashion, (c) extracting context information from the correlation among the different sources, and (d) predicting the client's behavior and changes of context from all sorts of data sources, from the most traditional ones (access points and phone characteristics) to the most recent ones (accelerometers and other sensors of the mobile devices, social networks, and intelligent vehicles). These tasks involve diverse challenges, such as extracting information from a huge amount of data from different types as well as identifying the sources models, finding adequate strategies and protocols for mobile data gathering, and developing inference techniques for context awareness by using machine learning tools. All these challenges are future directions for our research in the area of mobile sensing for context-awareness.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002. 4

[2] M. Baldauf, S. Dustdar, and F. Rosenberg. A survey on context-aware systems. *Int. J. Ad Hoc and Ubiquitous Computing*, 2(4):263277, 2007. 1, 2, 3, 4

[3] C. Bolchini, C. A. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. A data-oriented survey of context models. *SIGMOD Rec.*, 36:19–26, December 2007. 2

[4] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, 2000. 1, 2, 5, 6

[5] H. L. Chen. *An Intelligent Broker Architecture for Pervasive Context-Aware Systems*. PhD thesis, Faculty of the Graduate School of the University of Maryland, 2004. 1, 4

[6] S. L. T. Éamonn Linehan and S. Clarke. Supporting context-awareness: A taxonomic review. Technical Report TCD-CS-2008-37, Trinity College Dublin, 2008. 4, 6, 7

[7] T. Hofer, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger. Context-awareness on mobile devices the hydrogen approach. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, pages 292–302, 2002. 1

[8] J. Indulska and P. Sutton. Location management in pervasive systems. In *Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003 - Volume 21*, ACSW Frontiers '03, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc. 2

[9] P. Korpipää and J. Mäntyjärvi. An ontology for mobile device sensor-based context awareness. In *Proceedings of the 4th international and interdisciplinary conference on Modeling and using context*, CONTEXT'03, pages 451–458. Springer-Verlag, 2003. 1, 4

[10] P. Korpipää, J. Mäntyjärvi, J. Kela, H. Keranen, and E.-J. Malm. Managing context information in mobile devices. *IEEE Pervasive Computing*, 2:42–51, July 2003. 1, 2, 3, 5

[11] J. B. Predd, S. B. Kulkarni, and H. V. Poor. Distributed learning in wireless sensor networks. *IEEE Signal Processing Magazine*, 23:56–69, July 2006. 5

[12] J. B. Predd, S. R. Kulkarni, and H. V. Poor. A collaborative training algorithm for distributed learning. *IEEE Trans. Inf. Theor.*, 55:1856–1871, April 2009. 5

[13] T. Strang and C. Linnhoff-Popien. A context modeling survey, September 2004. 4, 6

[14] H. w. Gellersen, A. Schmidt, and M. Beigl. Multi-sensor context-awareness in mobile devices and smart artefacts. *Mob. Netw. Appl*, 7:341–351, 2002. 2, 6, 7

[15] J. yi Hong, E. ho Suh, and S. jin Kim. Context-aware systems: A literature review and classification. *Expert Systems With Applications*, 36:8509–8522, 2009. 2, 3, 5, 6