Sharing of context information in pervasive computing

Waltenegus Dargie, Olaf Droegehorn, and Klaus David
Department of Electrical Engineering and Computer Science
Chair of Communication Technology (ComTec)
University of Kassel
Wilhelmshöher Allee 73
D-34121 Kassel
Email: {waltenegus.dargie, olaf.droegehorn, klaus.david. } @ comtec.e-technik.uni-kassel.de

Abstract

Provision of services that suit the setting of a pervasive user and the reduction of user side input (less obtrusiveness) are two main goals of a context aware computing. To attain these goals, applications should be able to seamlessly interact with a large number of wireless sensor nodes in order to gather relevant information regarding the user and his environment. Since, either nodes or the user or both can be mobile, applications must identify relevant sensor nodes at runtime, possibly to share them with other applications. However, the pervasive user faces two challenges: mobile devices and wireless sensors are resource-constrained, on one hand; and on the other hand, the mobile user requires timely context information. Therefore, fast and efficient search for sensor nodes is essential. In this paper we advocate dynamic context information sharing; and introduce an architecture that supports it. We will compare our approach with previously suggested protocols to show that our approach conserves processing time by introducing context dependent, hierarchical, and semantic-based search.

KEY WORDS

Pervasive Computing, Context awareness, SDP (service discovery protocol), Search result data transmission, search time

I. INTRODUCTION

Recent advances in wireless sensor networks have enabled the deployment of dense yet cheap and unobtrusive sensors almost everywhere [1]. These sensors are capable of processing data locally and use a wireless link to transmit sensed data to a remote destination.

Likewise, developments in integrated circuit technology have enabled the integration of multiple micro-sensors and -actuators as a single, intelligent node of compact geometry that can be embedded in a mobile device, such as a mobile phone or a PDA [2].

The next step is to impeccably link the physical world to digital data networks so that a pervasive user can access environmental data virtually from anywhere without imposing restriction to his mobility or attention.

Sensed data give applications active awareness regarding the user's environment so that services can be tailored to suit his setting. By active awareness we mean that the user not only has knowledge about his computing environment, but also can monitor and control it by way of virtual presence; that is, without the need to be actually present at the place where an event is taking place. Active awareness also means that applications 'know' where and how to present sensitive services without compromising the user's privacy.

One main challenge is that a pervasive user might change his environment quite often. With change in environment, his preferences potentially change. Moreover, the state of available resources changes as the user moves in time and/or space - new sensor nodes with better quality of services may arrive; existing nodes may fail, move, or, depending on their residual energy, report wrong information [3].

Consequently, context aware applications must be designed to receive data input from multiple sensors and to adapt as the available sensors change over time. This can be done if they are supported by a distributed, efficient, and fast searching mechanism that provides network resources at runtime.

To highlight our discussions, we give two brief, but practical scenarios below.

Suppose Christian arrives at the University of Kassel to attend a conference but he is not familiar with the campus. At the entrance gate he signs as a guest to download a context aware Campus Navigator application that helps him find his way through the campus. The application needs a location sensor; but Christian does not have one on any of his mobile devices. Within his Bluetooth proximity, there are three students (Janet, Peter, and Phoebe) who happen to have location sensors, and are willing to share their location context with him, as their location is his location.

Peter has a GPS receiver, which gives absolute location information; Janet uses her PDA infrared port to detect infrared beckons which are distributed inside every building, intended for indoors use; Phoebe uses short-range RF transceiver to detect RF beckons, intended for building-wise sensing.

Christian's campus navigator dynamically selects Phoebe's location sensor to reach at the right building. Such a sharing scenario is shown in Figure 1 and 2.

As Christian enters the conference room, another context aware application that monitors the status of his mobile devices begins querying in an ad hoc manner a set of sensors for environmental data to determine Christian's activity.

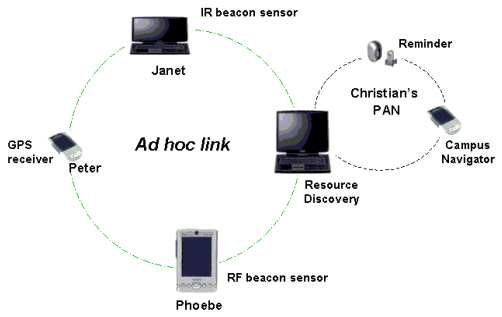

Figure1 Dynamic Context sharing

Some of the data obtained are:

Sound Level = 12 dB
Light Intensity = 1000 Lux
Temperature = 20ºC
Location = Uni-Kassel [Room-2343]
Collocated_ People = 75
Application Running = [PowerPoint]
Time = 9:00 AM

With these data, the application consults its codebook to determine the setting in which its user is found. It appears Christian is in a conference hall. Once the setting is determined, the application refers previous actions for the same setting. Then it takes the following decisions: firstly, it lowers the volume of Christian's PDA; secondly, an audio reminder is played: "It appears you are attending a conference; do you want to switch off your mobile phone?"

The two scenarios above demonstrate some typical features of context aware computing. The first scenario demonstrates the fact that in a pervasive environment either applications or the sensors they need, or both, may not be *a priori* known. There must be a dynamic discovery of resources everywhere, anytime; and these resources must be shared between multiple users.

The second scenario exhibits the amount of sensor input needed to undertake high-level context composition without obstructing the user. Composition may involve many atomic contexts that are directly sensed using a physical sensor; or it may involve high-level data processing entities, which make some form of context transformation, such as context mapping or interpretation, to a sensed data.

The location information, *Location = Uni-Kassel [Room-2343]* is a typical example. It can be a result of data first received from a GPS receiver and is then mapped to a street address in Kassel; or it can be a mapping of beacon ID that is obtained from an RF receiver within the campus.

Both scenarios show that unlike many traditional applications, context aware applications are *data-driven* [4], in that they proactively collect data from a physical environment at runtime. There is therefore an intensive data flow between applications and service discovery mechanisms to identify available resources. If the two reside on separate devices and the media of communications between them is a wireless link (as is often the case in an ad hoc scenario) this may incur considerable bandwidth consumption and processing time.

In this paper we will discuss some service discovery protocols (SDP) that are proposed for wireless sensor networks and show some of their limitations. Once we have done that, we will introduce our approach for an improved performance. We will show how our approach reduces both searching time and the amount of search result data transmission.

This paper is organised as follows: section II discusses typical challenges in resource sharing and previous approaches to tackle them; section III introduces our approach that is aimed at fast searching and minimising the amount of search-result data transmission; and finally section V gives a brief conclusion.

II SERVICE DISCOVERY PROTOCOLS

A "traditional" way of accessing sensed data uses a warehouse approach, where data are extracted from devices in a predefined way and stored in a centralised database system that is responsible for query processing [5]. Applications view the sensing network as a single unit that supports high-level query language [6]. This approach is suitable for fixed, task-specific[1] sensor networks. It has one major limitation, nevertheless: the warehouse approach consumes network resources to transfer large amount of raw data from devices to databases. Besides, for mobile sensor nodes a centralised approach may not scale.

Another approach is a distributed device database approach [5], where the query workload determines the data that are extracted from remote sites, and where possibly a portion of the query is analysed locally on the sensor node. Here nodes are capable of processing query requests and have sufficient memory to save historical sensed data. When applications are interested to query a particular region of a sensor network, they flood the entire network with their requests. In such a scenario, nodes are aware of their own location (possibly via a GPS sensor). So, nodes that cover the region in which an application shows interest cooperate with one another to make low-level data aggregation, thereby reducing the amount of data that must be transmitted to the application. This approach marginally optimises network

[1] The task type of the network is known at the time the sensor network is deployed; or the network may be reprogrammable and the task it supports may change slowly overtime [4].

resources. However, it is still suitable for task-specific networks, since all nodes must remain active, their radio turned on, listening for a request. This may cause considerable waste of node energy.

A more flexible and efficient approach uses a service discovery protocol (SDP); not only to identify eligible nodes but also to access important information such as the type of data that can be provided by a node, the mode[2] in which the node can operate, transmission power levels, and the current residual energy [7]. Such data enriches an application's knowledge regarding a node and the quality of service it delivers.

In a simple SDP, such as the Bluetooth *universally unique identifier* (UUID) [8], services register using a universally unique ID. A request query for a given service specifies the ID of the service it searches and expects to receive either a positive or negative response – no inexact matching is possible or required.

Another SDP approach, mostly used by Internet search engines (such as Google), attempts to match the pattern of keywords or attributes of a service description to a request query. The service-requesting query may contain a specific name and/or a set of one or more attributes. The SDP attempts to match the query's pattern with the pattern of the service description its database contains. If *m* out of *n* attributes match, the SPD deems searching a success. If there are *j* services whose attributes overlap, they may all be considered eligible and it is up to the application to refine the inexact search response.

More sophisticated service discovery protocols use semantic information associated with services to improve the quality of service discovery. This allows applications to set priorities, expected values of service attributes, and some index of a match's closeness [9].

The Jini [10] service discovery protocol and leasing mechanism is one example. It uses a trio of protocols called *discovery*, *join*, and *lookup.* A pair of these protocols, *discovery* and *join*, occur when a resource is plugged in; d*iscovery* occurs when a service is looking for a lookup service with which to register; j*oin* occurs when a service has located a lookup service and wishes to join it. *Lookup* occurs when a client or user needs to locate and invoke a service described by its interface type.

Jini is a heavyweight SDP, most suitable for fixed networks as opposed to wireless networks. For resource constrained wireless sensor networks, however, it is slow because access to individual sensors is object-based, affecting the potential low-level energy saving by data aggregation [7].

The Enhanced Bluetooth SDP [9] is another example that handles searching beyond pattern matching. It uses a semantic based service discovery mechanism by implementing the Darpa Agent Markup Language and Ontology Inference Layer (DAML+OIL) [12]. Here searching is made rigorously to reduce the number of inexact request matching so that application side processing is minimised, and to reduce the amount of search result data transmission. The SDP, however, imposes heavy computational and memory burden upon resource-constrained devices. Besides, search is slow. In the next section, we will discuss how we can improve the performance of this SPD by introducing context-based, hierarchical search suitable for wireless sensor networks.

[2] Some nodes can operate as cluster nodes or gateway nodes. The reader is kindly referred to [11] to get a better understanding of different modes of operations in wireless sensor networks.

III. CONTEXT BASED NODE ORGANISATION

We aim at the following three important goals:

- Ease of application development by shielding application developers from the worry of sensed data collection.
- Enhance accuracy of search result through systematic node organisation.
- Improve search time by using caching.

The choice of a specific SDP is mainly dictated by processing time and bandwidth. If processing time is the crucial issue at the side of the service discovery mechanism, a pattern-matching algorithm yields a faster response time. This is particularly useful if the discovery mechanism has to accommodate a large number of requests. But search result is inexact, and needs further application side processing. If on the contrary, bandwidth is a more important issue, and application side processing should be avoided, semantic based search is more effective since it minimises inexact matching.

The main problem with semantic based search is that it is rigorous and slow. If the SDP has to deal with a wide range of resources and should accommodate a large number of requests, it does not scale. Moreover, it causes latency to sensed data [7].

By systematically balancing the two approaches, a better performance can be attained. Consequently, we propose a hierarchical search mechanism and the caching of search results to improve the performance of a semantic based SDP.

Unlike general-purpose communication networks, resources supported by wireless sensor networks can be hierarchically and ontologically classified based on the type of context they sense and/or process. Figure 3 shows a layered architecture we propose to classify network resources.

At the lower level we have sensors; they sense the physical world directly. We classify them by the atomic context they sense, such as seismic, acoustic, visual, etc. The second layer contains nodes, which are responsible to gather raw data from child sensors for a low-level data aggregation. We classify aggregator nodes as homogeneous and heterogeneous aggregators. Homogeneous aggregators receive raw data from homogeneous sensors. For example, a temperature aggregator node receives a raw data from many temperature sensors within a certain region to calculate the average temperature. A heterogeneous aggregator

receives raw data from heterogeneous sensors. For example, to resolve the identity and orientation of a moving object a node may require raw data from motion sensors and a digital camera.
At a given time a sensor node may be found in either or both of the last two layers, though it is costly for a wireless node both to sense and to aggregate sensed data at the same time.
The third layer is a mediation layer; it facilitates node discovery. It contains at least one SDP and a number of node organisers (NO). We discuss node organisers shortly. Through the mediation layer, context aware applications identify the suitable set of sensor nodes at runtime in a dynamic and scalable fashion.
Figure 2 shows our layered architecture for wireless sensor networks.

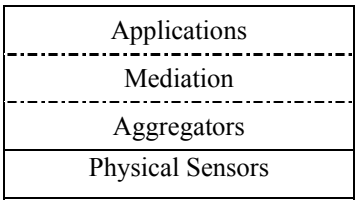

Figure 2. A layered architecture to represent wireless sensor nodes

Once we arrange resources as described above, we take a two-tier approach to the mediation layer in order to make searching more efficient and fast.
At a primary level, a lightweight pattern-matching search identifies services that closely match to a request query. If this search resolves only to a single node, no further semantic based search is necessary. Search result is immediately returned to the application. If, however, a pattern-matching result has resolved to a large number of nodes and further refinement is required, a secondary level, semantic based search is carried out by an NO per context.
We restrict a given NO to specialise in a particular resource at a given layer. A location sensor node organiser, for example, is responsible only for location sensors or aggregators, and can handle a semantic based search only for location sensor nodes. By this we make sure that search at the NO is lightweight, faster, and more accurate.
We assume reasonably that a patter-matching search can resolve in which class of resources a request query is interested. In the first scenario of section I, for example, we see the Navigator searching for a location sensor. Once the context of a node is identified, an NO that has semantic-based information for that particular context carries out the rest of the assignment.
Since we invoke an NO to carry out a rigorous search only for a particular context, searching time is by far smaller than searching through a general-purpose resource discovery database. Another shortcoming of almost all service discovery protocols we discussed so far is that a new search must be made every time a search query is received. If two applications are interested for a particular sensor node at the same time or in closer time interval, the SPD does not keep history of previously made search results. Therefore it has to process search query again and again. In our approach, all Node Organisers keep a history of previously made search results, the latency of which is determined by the nature of the wireless sensor network.
When an NO receives a pattern-matching search result from a service discovery mechanism or a request query directly from an application, it first consults its history to see if a similar query has been handled previously before it carries out a semantic search; if yes, then it pings the node for availability. If the node is still available, then no semantic search is needed. Only otherwise, will it carry out a rigorous, semantic based search.
Figure 3 shows our hierarchical service discovery approach.

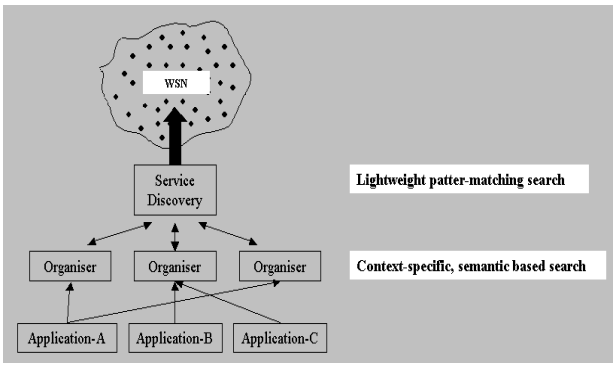

Figure 3. A hierarchical service discovery approach for a wireless sensor network

IV. CONCLUSION

Identifying the right sensor nodes at the right time to extract context data from a physical environment is a challenge of context aware applications. Consequently, a scalable and fast service discovery protocol suitable for wireless sensor networks is important.
In this paper we studied some existing distributed service discovery protocols since wireless sensor networks share many properties with traditional distributed systems. We showed that most service discovery protocols either return inexact search results to gain processing time, or impose computational and memory burden on resource-constrained mobile devices. Therefore, we proposed a context dependent, hierarchical, and semantic-based SDP (service discovery protocol) to avoid inexact search results while maintaining small processing time. We also showed how our approach improves processing time by keeping the history of previous search results.

ACKNOWLEDGMENT

We would like to acknowledge partial funding of the German “Bundesministerium für Bildung und Forschung

(BMBF)" in the framework of the Wireless Internet Projects.

REFERENCE

[1] G. J. Pottie and W. J. Kaiser, "Wireless Integrated Network Sensors," *Communication of the ACM, Vol. 43, No. 5, May 2000 pp. 51-58.*

[2] "Smart Dust training seminar," *Crossbow Technology, San Jose, CA, 2004*.

[3] Umar Saif, "Architectures for Ubiquitous Computing," *Ph. D dissertation, University of Cambridge, UK, 2002*.

[4] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *IEEE/ACM Transactions on Networking, Vol. 11 No. 1 February 2003*.

[5] P. Bonnet et al., "Querying the Physical World," *IEEE Personal Communication, October 2000, pp. 10-15*.

[6] Philip B. Gibbons et al., "Iris Net: An Architecture for a Worldwide Sensor Web," IEEE Pervasive Computing, October 2003, pp.22-33.

[7] W. Heinzelman et al., "Middleware to Support Sensor Network Applications," *IEEE Transactions on of Networking, January 2004, pp. 6-14*.

[8] http://www.bluetooth.com

[9] S. Avancha, A. Joshi, and T. Finin, "Enhanced *Service Discovery in Bluetooth," IEEE Comp., Vol 35, No. 6, June 2002 pp. 96-99*.

[10] http://www.java.sun.com

[11] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transaction on Wireless Communications, Vol. 1, No. 4, October 2002, pp. 660-670*.

[12] http://www.w3c.org