Prototyping Service Discovery and Usage in Wireless Sensor Networks

Raluca Marin-Perianu, Hans Scholten and Paul Havinga
University of Twente
Enschede, The Netherlands
Email: {r.s.marinperianu, j.scholten, p.j.m.havinga}@utwente.nl

***Abstract*—Heterogeneous Wireless Sensor Networks (WSNs) are envisioned to provide different types of services in an open and dynamic environment. This paper presents the design, implementation and evaluation of a service discovery and usage solution for heterogeneous WSNs. The users have the possibility to discover and use the services available in the WSN, while sensor nodes can search for the existing gateways to the outside world in order to signal important events. The WSN self-organizes in a clustered structure that acts as a distributed directory of service registrations. The clustering offers the necessary support to achieve energy-efficient discovery within the WSN. We implement the combined solution on resource-constrained sensor nodes, and we analyse the performance characteristics. The results show that the clustering algorithm has low communication overhead and the service discovery protocol scales with the number of nodes and network density. In addition, we show that the solution is lightweight (both code and data memory footprint) and the the interaction user-WSN is straightforward and intuitive.**

I. Introduction

Wireless Sensor Networks (WSNs) can make real the vision of smart surrounding spaces, such as houses, offices or public places. The potential ubiquitous usage has led to a rather heterogeneous market, offering today a broad spectrum of sensor nodes, ranging from tiny devices operating with limited hardware resources, to powerful nodes approaching the capabilities of an embedded computer [9]. WSNs are typically static networks of sensor nodes, homogeneous both in hardware and software, mainly deployed to collect sensor readings and route them to the sink [4]. However, we see nowadays a new trend, where heterogeneous and mobile WSNs are able to assist multiple applications in dynamic environments [3]. Compared to using one platform that imposes a set of compromises, a heterogeneous collection of devices benefits from the functionality and flexibility provided by the resource-lean nodes, in conjunction with the enhanced capabilities offered by the more endowed nodes [6]. We envision that large-scale heterogeneous sensor infrastructures will eventually be integrated into offices, public places and the environment, where multiple clients will configure, discover and use a variety of services.

We see therefore a strong motivation for having a uniform abstraction to expose the functionality of the WSN to the user. In this sense, service-based solutions have become popular [7], as they provide an easy-to-use interaction paradigm and decouple the application from the low-level technologies. In addition, the mobility and ad-hoc nature of the WSN applications can be handled through dynamic mechanisms, such as service discovery. As a first step towards service-oriented WSNs, we proposed an energy-efficient, distributed service discovery mechanism SD4WSN [14], which allows for automatic detection and usage of services in a WSN. The algorithm is simple enough to run on resource-constraint devices and is shown to be energy-efficient especially in large-scale, dense sensor networks.

In this paper, we focus on practical issues with designing, implementing and evaluating the service discovery and usage mechanisms on resource-lean sensor nodes. The resulting service-oriented WSN has to fulfil the following three objectives:

1) To self-organize in a clustered structure and to detect and react to topology changes due to mobility or communication errors.
2) To offer the users the possibility to discover and use the services available in the network. The discovery protocol should exploit the underlying clustered topology for a scalable and energy-efficient search.
3) To discover and use the available gateways to the outside world (mobile phones, PDAs, laptops) that can be used for signalling relevant events, for example alarms in the case of safety-critical situations.

From the practical point of view, we show that the solution is lightweight (both code and data memory footprint), the interaction user-WSN is straightforward and intuitive and the alarming from WSN to the world is simplified by using the same discovery and communication protocol. From the theoretical point of view, the performance measurements show that the clustering algorithm has low communication overhead and the service discovery protocol scales with the number of nodes and network density.

The remaining part of the paper is organized as follows: we give an overview of the related work in the fields of service-oriented WSN and service discovery in Section II. We briefly describe the main modules of our solution in Section III. In addition, Section IV presents a series of optimizations and extended functionality. Sections V and VI give the implementation details and describe the demonstration setting. The experiments and performance evaluation are addressed in Section VII. Section VIII describes the practical challenges that we faced during the implementation process. Finally, Section IX presents the conclusions.

II. Related work

Heterogeneous WSNs are envisioned to provide different types of services in an open and dynamic environment. Tilak et al. [15] identify the resource and service discovery as the first step towards enabling interoperability of sensor networks. However, the fact that different sensor networks are deployed by different organizations introduce unique challenges for achieving interoperability. Therefore, researchers have been investigating service-oriented solutions for WSNs, which can offer a well-defined set of interfaces for a standardized network operation and easy access to the WSN functionality. A possible approach is to have a uniform abstraction layer at the level of sink or gateway nodes, which can act as a bridge between different WSNs. Isomura et al. [10] propose that WSNs are connected through a UPnP bridge, which offers a way to discover and access the services from various sensor networks. This method solves the problem of exposing the functionality of the WSN, but is not a solution for dynamic sensor networks, where there is a need to actively update, configure and discover the available services inside the network.

The Atlas service-oriented platform [11] offers a hardware and software solution for connecting heterogeneous devices within a service-oriented architecture. The service registry resides on a centralized server, running the OSGi middleware. Due to the centralized architecture and the lack of ad-hoc networking support, Atlas is more suited for small-scale settings such as smart houses, rather than multi-hop dynamic WSNs.

Delicato et al. [7] propose a flexible Web Services approach for the design of a sensor network, where the sensor nodes act as service providers for the external users. The communication within the sensor network is accomplished by using directed diffusion and formatted SOAP messages. Sink nodes have a double role, acting both as registries for the services inside the sensor network and as service providers to the external environment. Every sink node has the complete knowledge of all the services in the network. However, this approach cannot meet the scalability and energy-efficiency requirements for large-scale dynamic WSNs.

Service discovery is a topic of great interest in the field of ad-hoc networks [13]. However, the protocols proposed are not directly applicable to WSNs, as they build complex overlay structures that require a high maintenance effort. For example, Kozat and Tassiulas [12] build a dominating set, or backbone, to which devices register their services. Due to the high density of nodes in the backbone, many loops are generated during service discovery. To overcome this drawback, the backbone organizes in a source-based multicast tree. However, building and maintaining two overlay networks is expensive for resource-constraint sensor nodes.

Our discovery method relies on a lightweight clustering structure which allows for low maintenance overhead and low discovery cost even in highly dense sensor networks. We show the feasibility and the effectiveness of the solution by implementation on real sensor nodes.

III. Solution overview

At a high level, a service-oriented solution is composed of three core pieces: service providers, service consumers and service registries, or directories. The directories facilitate the discovery of services in the network, acting as intermediaries between providers and consumers: providers need to publish and register their services, while consumers are interested in finding the service providers. This process is regulated by a service discovery protocol, which offers the service search capability needed by the service consumers.

The prototype service-oriented WSN that we present in this paper uses SD4WSN [14], a service discovery protocol for heterogeneous WSNs. SD4WSN is designed for energy-efficiency and scalability by taking advantage of the network heterogeneity. The idea is that the network is organized into clusters, where the most capable nodes are the clusterheads. The service-discovery messages travel among the clusterhead nodes, which form a distributed directory of service registrations.

A. Clustering algorithm

The clustering structure facilitates the construction and maintenance of a distributed directory. Among the features of the algorithm we mention the following: (1) decisions are made based on the 1-hop neighbourhood information, which leads to a fast convergence in face of topology changes, (2) the chain reactions [5] are avoided, such that local topology changes determine only local modifications of the directory structure, (3) the algorithm constructs small-height clusters without imposing a maximal height limit [14].

For building the distributed directory, each node is assigned a weight, termed the *capability grade*, representing an estimate of the node's dynamics and available resources. These weights are assumed to be unique, as the node hardware identifier may be used to break ties.

1) Cluster setup: The algorithm constructs a set of trees, based on local knowledge about neighbouring nodes. The protocol works as follows:

- Nodes that have the highest capability grades among their neighbours declare themselves clusterheads and broadcast a *SetRoot* message announcing their roles.
- The remaining nodes choose as parent the neighbour with the highest capability grade.
- When a node receives a *SetRoot* message from its parent, it learns the cluster membership and rebroadcasts the *SetRoot* message.

Figure 1 shows an example network of nine nodes, where each node is assigned a capability grade. Based on these capability grades, the nodes organize into three trees (clusters), having as roots the nodes with the highest capability grades in their neighbourhood (7, 8 and 9). The straight arrows indicate the parent-child relationship.

2) Knowledge of adjacent clusters: The identity of the root node is propagated in the cluster down to the leaf nodes via the broadcast message *SetRoot*. Thus, the message also reaches

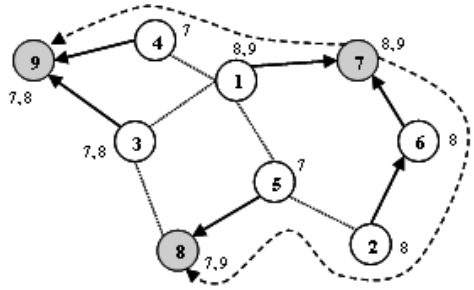


Fig. 1. Example of a clustered network.

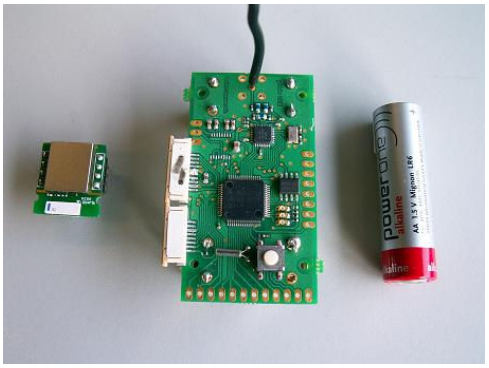


Fig. 2. The hardware.

nodes from adjacent clusters, which store the adjacent root identity. This information is then propagated up in the tree until it reaches the root node, by using a message which we term *UpdateInfo*. Through this message, nodes learn which are the clusters adjacent to their sub-trees and the next hops on the paths leading to their clusterheads. In particular, the root nodes learn the identity of all the root nodes from the adjacent clusters.

In Figure 1, the local knowledge on the adjacent clusters is displayed next to each node. This information is aggregated at the higher levels in the hierarchy, such that the root nodes inherit the information acquired by all the nodes in their cluster.

3) Maintenance in face of topology changes: Nodes adjust their cluster membership when the network topology changes in the following way:

- A node discovers a new neighbour with a higher capability grade than its current parent. The node then selects that neighbour as its new parent.
- A root node discovers a neighbour with a higher capability grade. As a result, the root node gives up its role and chooses that neighbour as its parent.
- A node detects the failure of the link to its parent. The node then chooses as new parent the node with the highest capability grade in its neighbourhood.

The knowledge of adjacent clusters changes according to the modifications of the clustering structure.

An in-depth description of the clustering algorithm can be found in [14].

B. Service discovery protocol

The service discovery protocol uses the clustering structure to maintain a distributed directory of service descriptions. In this way, the communication cost is reduced, since the service discovery messages are exchanged only among the directory nodes.

Nodes register their services with their parents and thus every node in the hierarchy maintains information on the services offered by the nodes in its sub-tree. The root nodes have a complete view of the services in their clusters. The service registration process is integrated with the construction and maintenance of the clustering structure, by using the same message *UpdateInfo*. The service discovery protocol consists of two phases: the discovery phase and the reply phase. We describe each of them in turn.

1) The discovery phase: During the service discovery phase, the discovery message travels among the nodes which are part of the distributed directory. Suppose a node in the network generates a service discovery request *ServDisc*. The request is first checked against the local registrations. If no match is found, the message is forwarded to the parent. This process is repeated until the *ServDisc* message reaches the root of the cluster. When a root node receives a service discovery request message and it does not find any match in the local registry, the *ServDisc* message is forwarded to the roots of the adjacent clusters. The next hop on the path leading to the adjacent cluster is decided by every node that acts as forwarder of the *ServDisc* message, by using the local knowledge of adjacent clusters. Figure 1 shows an example of how service discovery messages travel from root node 7 to the adjacent clusters 8 and 9. The paths followed by the two messages is indicated with dashed arrows.

2) The reply phase: The result of a service search is typically the address of one or more service providers, information which is included in the reply message. The reply message may follow the reverse cluster-path to the client, or any other path if a routing protocol is available. In the first case, if there is a cluster partition, the path can be reconstructed by using the same search strategy as for the *ServDisc* message, where this time the service is the address of the client.

IV. Optimizations and extended functionality

In addition to the clustering and service discovery modules, our solution integrates a series of optimizations for further reducing the energy consumption. Moreover, the functionality is extended by allowing mobile gateway nodes to join the WSN in an ad-hoc manner.

A. Cross-layer integration with MAC

During cluster setup and maintenance, each node broadcasts *SetRoot* messages, announcing its role. Via this message, the root identity is disseminated to the whole cluster and to the borderline nodes belonging to adjacent clusters. A cross-layer approach with a TDMA-based MAC protocol allows for

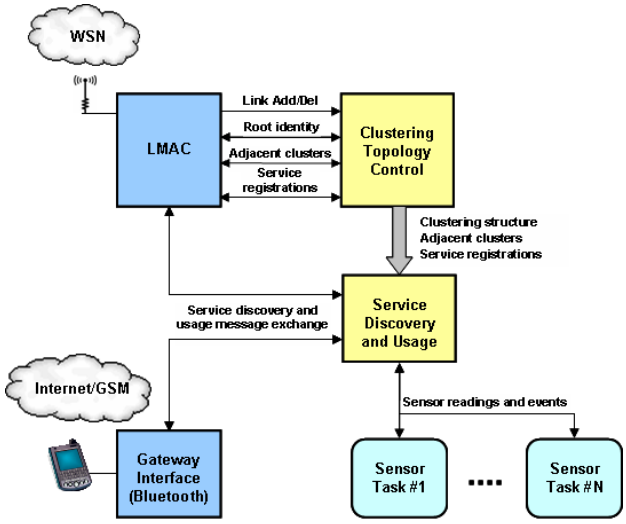


Fig. 3. Implementation overview.

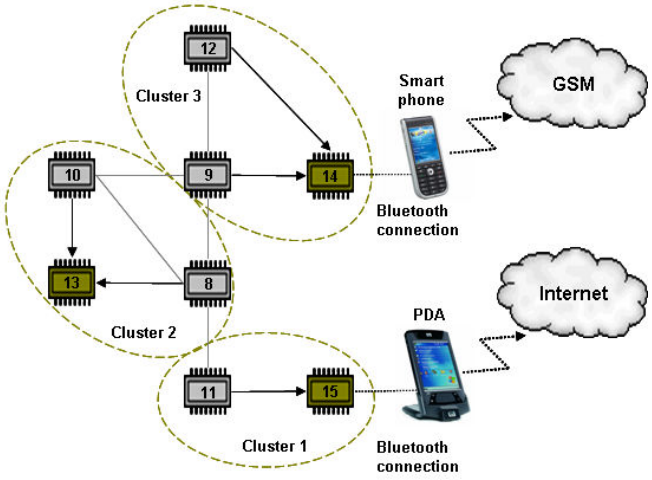


Fig. 4. Demonstration setting.

integration of the *SetRoot* message with the control message exchanged periodically by neighbours. This method reduces the communication overhead and thus increases the network lifetime (see Section V-B for a description of the cross-layer integration with LMAC).

B. Integration of discovery and usage

After service discovery, the consumer is interested to make use of the desired service, by selecting and establishing a connection with the appropriate service provider. In the case where providing a service is equivalent to sending a short piece of information to the consumer, extensive message exchange can be avoided by integrating the service usage in the service discovery process. We identify the following cases:

1) *Service discovery and usage integration during the reply phase.* The service reply may incorporate partially or totally the information exchanged during service usage. For example, suppose a service consumer issues a discovery message which searches for the location of the nodes that are sensing a particular measure or detecting an event. Upon receiving this message, the nodes that match the service description issue a reply message that already includes the location coordinates. In this way, subsequent service usage dialogue is avoided.
2) *Service discovery and usage integration during the discovery phase.* In a similar manner, the discovery message may include an event, such as an alarm about an undesired situation detected in the network (for example a temperature exceeding a threshold value). In this case, the service discovery message tries to discover gateway nodes capable of announcing the event outside the network. The discovery message contains the service usage information (e.g. the sensor readings), so the service reply and usage phases may be no longer be required.

C. Integration with mobile platforms

The traditional model of a WSN requires one sink node that collects the sensor data and acts as a gateway to other networks such as Internet [4]. Our environment is composed of a multitude of sensor platforms and mobile devices, which form a heterogeneous network. Some sensor nodes have gateway capabilities, being connected via wireless links (Bluetooth) to Smartphones or PDAs, which in turn are connected to the GSM network or to the Internet (through WLAN). In this way, the traditional model of WSN is extended to multiple, mobile gateway nodes. Using the gateway nodes, a two-way communication is possible: (1) users can interact with the WSN by discovering and using the available services, and (2) the WSN can discover and use the gateway nodes to announce relevant events, for example alarms in the case of safety-critical situations.

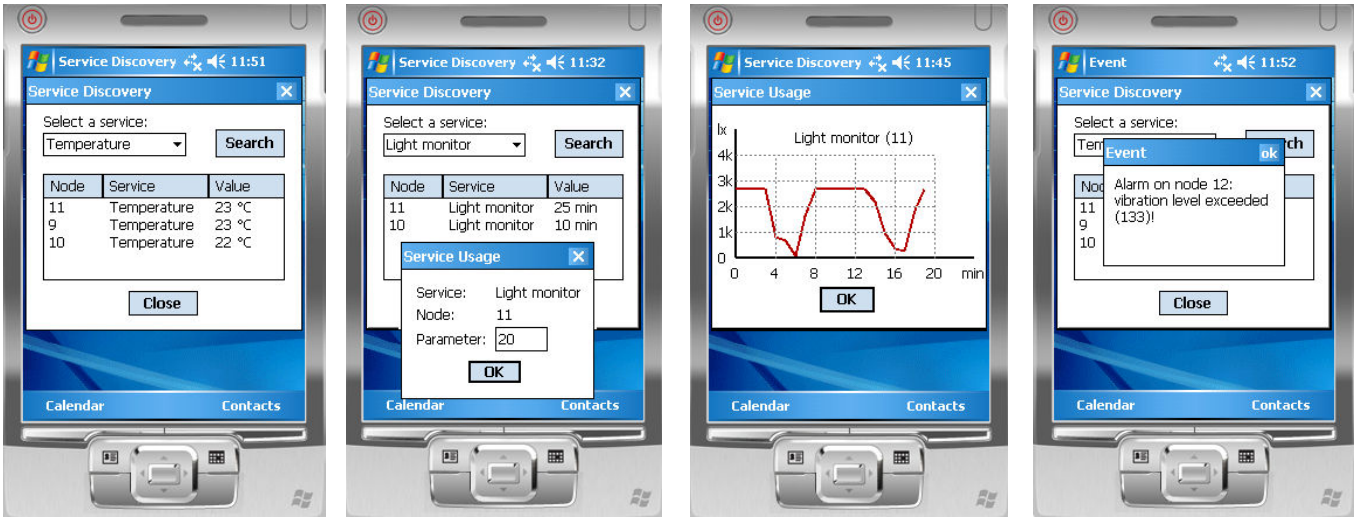
V. Implementation

We implement our service-oriented solution on sensor nodes, taking into account the optimizations mentioned in Section IV and the multiple gateway integration. In what follows, we describe the hardware and the software details of our implementation.

A. Hardware

We implement SD4WSN on the Ambient μNode 2.0 platform [1], provided with the low-power MSP430 microcontroller produced by Texas Instruments. The sensor node offers 48kB of Flash memory and 10kB of RAM. The radio operates in the 868MHz and 915MHz band and has a maximum data rate of 100kbps.

The connection of sensor nodes to mobile devices is done by using the Parani-ESD200 Bluetooth module [2]. This module can communicate through its on-board antenna with other Bluetooth devices in the range of 30m. It can be connected to the Ambient μNodes via standard UART interface, and



(a) Sensor reading service usage (b) Monitoring service - message exchange (c) Result of monitoring service usage (d) Gateway discovery for announcing events

Fig. 5. Service discovery and usage console.

configured and controlled by typical AT command set. Figure 2 shows the Ambient sensor node platform and the Parani Bluetooth module.

The node provided with a Bluetooth interface can connect to any other Bluetooth-enabled device that supports the Serial Port Profile. We use Smartphones and PDAs carried by people as mobile gateways that can connect to the GSM network and the Internet.

B. Software

Figure 3 shows the main functional modules of our prototype implementation. For regulating the access of the sensor nodes to the wireless medium, we use LMAC [8], an energy-efficient TDMA-based MAC protocol designed for WSNs. LMAC divides time into slots, each node being assigned one slot when it can transmit the data in a collision-free wireless medium. During its time slot, a node transmits a control packet (as a heartbeat), followed by the actual data (if any). The higher-layer protocols can use the control packet to piggyback a small piece of information for an energy-efficient cross-layer integration. We use this facility to disseminate and locally update the root identity, thus replacing the *SetRoot* message of our clustering algorithm (see Section IV).

By analysing the control messages received from the neighbours, LMAC constructs a neighbour table which is constantly maintained up-to-date. In this way, LMAC can inform the higher layer protocols of the changes in the network topology (links added or deleted). The *Clustering Topology Control* module analyses the neighbourhood information provided by LMAC and constructs the clustering structure, by choosing the appropriate parent node. Using the cross-layer optimization, this module is informed about the root identity from the control message exchanged periodically by LMAC. Through LMAC, the *Clustering Topology Control* module receives from the children nodes and transmits to the parent node the knowledge on adjacent clusters and the service registrations.

The *Service Discovery and Usage* module receives either from the neighbours, through LMAC, or from the mobile gateway, through Bluetooth, the service discovery and usage unicast messages. In order to decide the next hop for the service discovery messages, this module uses the clustering structure, the information on adjacent clusters and the service registrations provided by the *Clustering Topology Control* module. The received message is then forwarded either to a neighbour from the WSN through LMAC, or to the gateway, through Bluetooth. The *Service Discovery and Usage* module also receives the sampled values from the attached sensors, so that it can include the sensor readings in the service usage message exchange.

The SD4WSN modules, (e.g. *Clustering Topology Control Service Discovery and Usage*) have a total code memory footprint of 3KB and require 342B of RAM, considering an average network density of 32 neighbours per node.

VI. Demonstration Setting

We build a demonstration setting as a proof of concept for our service-oriented solution. Figure 4 shows a set of sensor nodes organized into three clusters and connected to gateway devices via Bluetooth interfaces. The capability grades of the nodes determine the formation of clusters. The gateway nodes have the two highest capability grades (14 and 15), as they are connected to the Smartphones and PDAs. Consequently, they are chosen as clusterheads of clusters 3 and 1. The clustering structure dynamically reorganizes in case of mobility or node addition/removal.

The sensor nodes are equipped with light, temperature and movement (tilt switch) sensors (see Table I). The following services are available in this setting:

1) *Sensor reading service* - provides a simple sensor reading at the moment of invocation (e.g. temperature reading).
2) *Monitoring service* - provides the history of a specified measure (e.g. light monitoring over the last 20 minutes). This service necessitates the establishment of a communication session between the provider and the consumer. First, the service parameters (e.g. time history) are established between the two parties. Then, the service provider delivers the desired history measurements to the user.
3) *Alert service* - enables the sending of SMS or Email through the mobile devices when an an abnormal situation occurs in the network. The WSN searches for these services provided by gateway nodes in order to announce the event to the GSM network or Internet.

Figure 5 shows the service discovery and usage console which runs on the mobile devices. The user can select the desired service from a list and launch the search command which initiates a service discovery message. The message is transmitted via the Bluetooth interface to the gateway node and then forwarded according to the SD4WSN protocol. When the service is found, a service reply message will announce the consumer about the result of the search.

The result of discovering a simple temperature reading service is shown in Figure 5(a). Besides the address of the service provider, the service reply message also incorporates the temperature value (see Section IV for the integration of service discovery and usage). The user can browse the list of sensor nodes that offer the temperature service, having also access to the sensor readings.

In contrast to the sensor reading service, the monitoring services require a more complex dialogue between provider and consumer. First, the user launches a discovery message to find out the providers of the monitoring service. Upon receiving the list of providers, the user selects one of them, establishes the service parameters and invokes the service. Figure 5(b) shows an example of a light monitoring service. The user selects node 11 from the list of providers and specifies the time history (20 minutes) over which to retrieve the sampled data (the maximum time history is embedded in the service reply messages: 25 minutes for node 11). The result of the service invocation is illustrated in Figure 5(c), where the history of the sampled data is plotted on the mobile device.

In the case where a sensor node registers an abnormal sensor value (temperature or light level exceeded, vibrations), the node issues a service discovery message which incorporates the sensor reading. The discovery message travels the clustered WSN in search for an SMS or an Email service. Upon receiving the message, the mobile device displays an alarm and announces the event (see Figure 5(d)).

TABLE I
SENSOR TYPES

Sensor	Manufacturer	Model
Light	Texas Advanced Optoelectronic Solutions	TSL2550
Temperature	National Semiconductor	LM92
Tilt switch	Assemtech Europe	CW1300-1

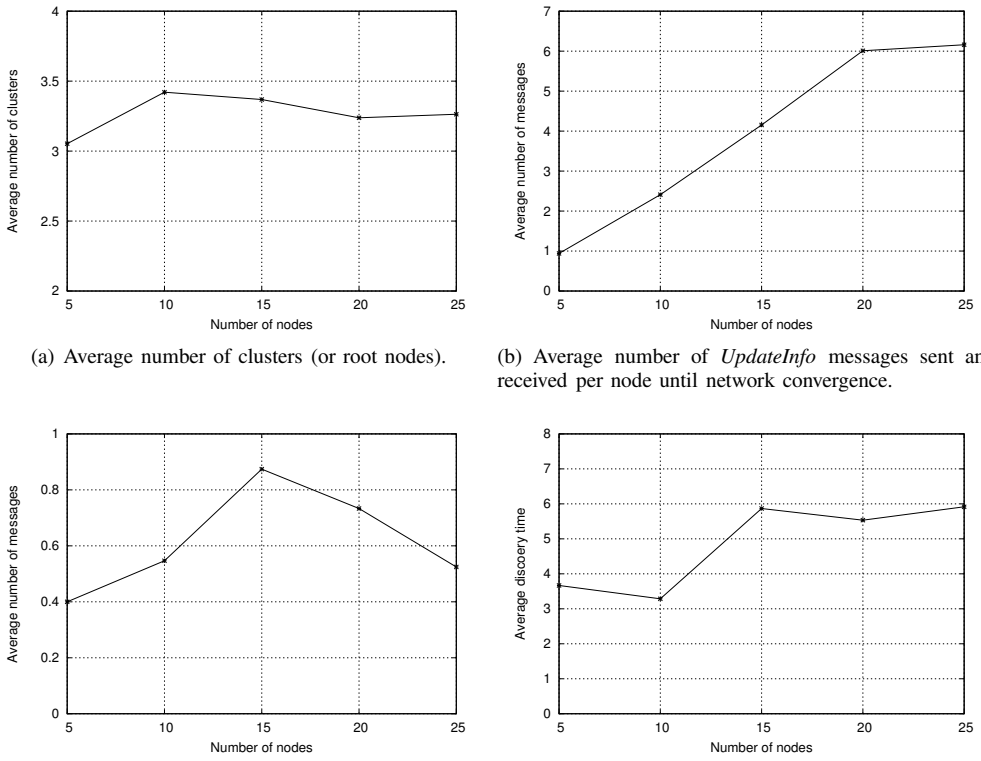
Fig. 6. Announcing events.

Figure 6 shows a detail of the demonstration setting, namely two sensor nodes, a mobile phone and a Smartphone. The first sensor node is equipped with a tilt switch sensor and is thus capable of detecting vibrations. The second node is provided with a Bluetooth interface and can connect to the Smartphone to send events. The situation depicted in the figure is the following: the node with a tilt switch sensor detects a high level of vibrations and issues a discovery message for an SMS service. The message travels the clustered WSN and reaches the node with the Bluetooth module, which provides the desired service and transmits the message to the Smartphone. The Smartphone in turn, displays an alarm and sends an SMS to the mobile phone.

VII. Performance measurements

We implement SD4WSN on the Ambient μNodes and we measure the performance by deploying a series of testbeds of up to 25 nodes. We analyse the number of messages exchanged until network convergence, the number of resulting clusters, the cost of service discovery and the discovery time, on an average case scenario. In order to estimate the average case, we generate a series of random topologies offline (up to 5 hops), which we disseminate to the WSN at the beginning of each test. The nodes are considered to be placed on a fixed-sized area. We vary the number of nodes from 5 to 25 and thus the network density (e.g. average node degree) increases from 1.4 to 7. For each network density in turn we run 20 experiments, each consisting of the following steps:

1) Disseminate the network topology
2) Compute the number of messages exchanged until network convergence.
3) Count the number of resulting clusters.



(a) Average number of clusters (or root nodes).

(b) Average number of *UpdateInfo* messages sent and received per node until network convergence.

(c) Average number of *ServDisc* messages sent and received per node during one service discovery phase.

(d) Average service discovery time in seconds.

Fig. 7. Performance measurements.

4) Issue a service discovery message from a random node in the network.
5) Calculate the number of service discovery messages exchanged and the time until the service is found.

Figure 7(a) shows the average number of clusters depending on the network density. The number of clusters is an important measure for the performance of a clustering algorithm that is intended to be used as a basis for a search mechanism. A high density of clusters leads to a large number of loops that occur during the discovery process. In the first part of the curve, the nodes are sparsely distributed on the area and generally form clusters with only one member. When the network becomes denser, the new nodes added either join the already existing clusters or they form their own cluster and force the root nodes in the neighbourhood to join. We notice the general trend that adding new nodes to the network does not significantly change the number of clusters, and consequently, the number of root nodes.

Figure 7(b) shows the average number of *UpdateInfo* messages sent and received per node for updating the knowledge on adjacent clusters and the service registrations. The messages are counted starting from a newly initialized network until the network convergence. We notice that in the first part of the curve, the number of messages per node grows linearly with the network density. The reason is that when increasing the density, the network becomes more connected and thus, the root nodes receive more information about the clusters in vicinity. However, in the last part of the curve, the number of messages per node remains relatively constant. For dense networks, it is likely that the information received from a new neighbour does not change the already acquired knowledge of adjacent clusters, so it is not propagated further in the tree.

We represent in Figure 7(c) the number of service discovery messages sent and received per node during one service discovery phase. We notice that in the first part of the plot, the number of service discovery messages increases with the number of nodes in the network. The reason is that in comparison with a sparse network, a connected network allows for messages to travel among all the clusters which are formed as a result of the clustering algorithm. This results in a higher discovery cost. Once the network is connected, increasing the density does not increase total discovery cost, because the service discovery messages travel only among the clusterhead nodes, which remain constant in number. Therefore, in the second part of the plot, the total number of discovery messages remains on average approximately constant, which means that the average number of discovery messages sent and received per node decreases.

The average discovery time is represented in Figure 7(d). We notice the same effect as in the case of discovery cost: once the network is connected, the average time it takes for a service discovery message to reach the service provider

remains approximatively constant.

We summarize the most important observations that validate from a practical point of view the theoretical results presented in [14]:

- After a certain network density, increasing the number of nodes in the network does not further increase the number of clusters.
- The average message overhead per node for cluster setup and maintenance is limited, since there is no need to transmit the knowledge on adjacent clusters that is already known in the higher levels of the cluster hierarchy.
- The discovery cost per node decreases with the network density, as the discovery messages travel only among clusterhead nodes.

VIII. Practical challenges

We faced a number of practical challenges while implementing the service discovery and usage mechanisms on sensor nodes:

- *Debugging:* Debugging proves to be very difficult in a WSN environment. We implemented the following mechanisms which helped us understand the behaviour of the nodes: (1) a report status message, sent periodically by every node to the base station, (2) a network packet sniffer that captures the traffic within the WSN, and (3) a GUI that shows online the functional nodes and the clustering structure. However, debug support for a real-life, large-scale multihop deployment is still an open problem.
- *Resource limitations:* The limited computational power of the sensor nodes makes it difficult to maintain the precise synchronization of the TDMA-based MAC protocol, while also running the other tasks, such as sampling, clustering, service discovery and Bluetooth communication. Especially the nodes equipped with Bluetooth had sometimes problems in establishing both wireless connections. Therefore, the timer values and scheduling scheme have to be carefully chosen.
- *Latency and uncertainty:* Latency can be an issue if a low duty-cycle MAC protocol is used. The clients of the service discovery protocol cannot expect fast answers to their queries, and also they do not have feedback on the progress of the discovery process within the network.

IX. Conclusions

Heterogeneous WSNs are envisioned to provide different types of services in an open and dynamic environment. In this context, a necessary ingredient to achieve full functionality and interoperability is the ability to discover and use the services available in the network. This paper presents the design, implementation and evaluation of SD4WSN, which solves the problem of energy-efficient service discovery and usage in heterogeneous WSNs. For achieving low communication costs during searching for services, the discovery protocol exploits a cluster overlay, where the clusterhead nodes form a distributed service registry. A service lookup results in visiting only the clusterhead nodes.

For evaluating the feasibility and the effectiveness of our approach, we implement the protocol on sensor nodes. As a first step, we build a demonstration setting, where the user can discover and use a set of services inside the WSN. Moreover, nodes from WSN can discover and use the available gateways to the outside world (mobile phones, PDAs, laptops) for signalling relevant events. We show that the solution is light-weight (both code and memory footprints) and alarming from WSN to the world is simplified by using the same discovery and communication protocol. As a second step, we build a testbed to evaluate the performance under different network conditions. The results show that our solution achieves low communication overhead and good scalability properties with respect to the number of nodes and network density.

References

[1] Ambient systems. http://www.ambient-systems.net.

[2] Parani-ESD200 Bluetooth module. http://www.sena.com/download/datasheet/ds_parani_esd.pdf.

[3] Smart surroundings. http://wwwes.cs.utwente.nl/smartsurroundings/.

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

[5] C. Bettstetter. *Mobility Modeling, Connectivity, and Adaptive Clustering in Ad Hoc Networks*. PhD thesis, Technische Universität München, Germany, October 2003.

[6] Alberto Cerpa, Jeremy Elson, Deborah Estrin, Lewis Girod, Michael Hamilton, and Jerry Zhao. Habitat monitoring: application driver for wireless communications technology. *SIGCOMM Computer Communication Review*, 31(2 supplement):20–41, 2001.

[7] Flavia Coimbra Delicato, Paulo F. Pires, Luci Pirmez, and Luiz Fernando Carmo. A service approach for architecting application independent wireless sensor networks. *Cluster Computing*, 8(2-3):211–221, 2005.

[8] L. Van Hoesel, T. Nieberg, J. Wu, and P.J.M. Havinga. Prolonging the lifetime of wireless sensor networks by cross-layer interaction. *IEEE Wireless Communications*, 11(6):78–86, 2004.

[9] Mike Horton and John Suh. A vision for wireless sensor networks. In *IEEE MTT-S International Microwave Symposium Digest*, pages 361–364. IEEE Computer Society, June 2005.

[10] Manabu Isomura, Till Riedel, Christian Decker, Michael Beigl, and Hiroki Horiuchi. Sharing sensor networks. In *ICDCSW '06: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems*, page 61, Washington, DC, USA, 2006. IEEE Computer Society.

[11] Jeffrey King, Raja Bose, Hen-I Yang, Steven Pickles, and Abdelsalam Helal. Atlas: A service-oriented sensor platform: Hardware and middleware to enable programmable pervasive spaces. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 630–638. IEEE Computer Society, November 2006.

[12] Ulas C. Kozat and Leandros Tassiulas. Service discovery in mobile ad hoc networks: An overall perspective on architectural choices and network layer support issues. *Ad Hoc Networks*, 2(1):23–44, June 2003.

[13] R. Marin-Perianu, P. H. Hartel, and J. Scholten. A classification of service discovery protocols. Technical Report TR-CTIT-05-25, Centre for Telematics and Information Technology, Univ. of Twente, The Netherlands, 2005.

[14] R. S. Marin-Perianu, J. Scholten, P. J. M. Havinga, and P. H. Hartel. Energy-efficient cluster-based service discovery in wireless sensor networks. In *Proceedings of the 31st IEEE Conference on Local Computer Networks*, pages 931–938, November 2006.

[15] Sameer Tilak, Kenneth Chiu, Nael B. Abu-Ghazaleh, and Tony Fountain. Dynamic resource discovery for sensor networks. In *EUC Workshops*, pages 785–796, 2005.