# Distributed Service Discovery for Heterogeneous Wireless Sensor Networks

Raluca Marin-Perianu, Hans Scholten and Paul Havinga

University of Twente, Enschede, The Netherlands
{r.s.marinperianu, j.scholten, p.j.m.havinga} @ utwente.nl

**Abstract.** Service discovery in heterogeneous Wireless Sensor Networks is a challenging research objective, due to the inherent limitations of sensor nodes and their extensive and dense deployment. The protocols proposed for ad hoc networks are too heavy for sensor environments. This paper presents a resource-aware solution for the service discovery problem, which exploits the heterogeneous nature of the sensor network and alleviates the high-density problem from the flood-based approaches. The idea is to organize nodes into clusters, based on the available resources and the dynamics of nodes. The clusterhead nodes act as a distributed directory of service registrations. Service discovery messages are exchanged among the nodes in the distributed directory. The simulation results show the performance of the service discovery protocol in heterogeneous dense environments.

## 1 Introduction

In recent years, the active research in the field of Wireless Sensor Networks (WSNs) has enabled a broad range of ubiquitous computing applications. Our vision is that the functionality of WSNs will go beyond passive gathering of sensed data, in the direction of providing active assistance in industrial processes, as well as in daily life. These contexts require, however, a complex and dependable functionality, which cannot be offered by individual, resource-constrained sensor nodes. We propose to address these challenges by providing functionality according to a *service oriented approach*. Organized into groups, the sensor nodes can take advantage of their cumulated resources and achieve better accuracy and reliability, providing complex services in real-world scenarios.

An essential component for the service oriented approach is the ability to discover services in a wireless sensor network environment. This is a challenging task, due to (1) the constrained capabilities of sensor nodes, (2) the mobility of both potential service providers and consumers, (3) the extensive and dense deployment of nodes in a sensor network. Moreover, there is already a large number of sensor hardware platforms that determines a significant variety in the capabilities of individual sensor nodes.

Our aim is to exploit the heterogeneity of WSN for the purpose of service discovery. We propose a solution that relies on an overlay topology which forms a distributed directory. The overlay is a clustering structure, where the clusterhead role is assigned depending on the capabilities of the nodes. Each clusterhead node keeps a service registry of all the services in its cluster. For a fast convergence of the structure in face of mobility, nodes make decisions only based on the 1-hop neighborhood knowledge. The energy spent on keeping a consistent service registry is minimized, as changes in the network topology trigger only local reconfiguration of the distributed directory.

In the following section we give two examples of applications where nodes offer complex collaborative services. Section 1.2 presents related work regarding service discovery and clustering algorithms. The overlay clustering structure is discussed in detail in Section 2. A description of the service discovery protocol is presented in Section 3. Section 4 evaluates the clustering algorithm and service discovery protocol through simulation results. Section 5 presents a summary and future work.

## 1.1 Scenarios

*Transport and logistics.* The first scenario describes the process of transporting goods from providers to consumers [13]. The products are placed on shelves which are fitted to rolling carts. Each cart is equipped with a wireless sensor node, termed a *micronode*. The nodes on the carts communicate with each other wirelessly, creating an ad-hoc network. Smaller sensors called *piconodes* are placed on each shelf of the cart for a precise control of environmental conditions when the delicate and perishable goods (such as flowers) are being transported. The micronodes gather information from piconodes and verify the environmental conditions.

The expedition floor is a place used by the transport company for the shipment process. Loaded carts are pushed by the transport company personnel on the expedition floor and are placed depending on the shop they are assigned to. A fixed infrastructure of sensor nodes, termed *beacons*, is placed expedition floor. Their role is to assist the localization process of the loaded carts and to monitor the correct placement depending on the assigned shop.

Summing up, the services involved in this scenario are the following:

– *Providing environmental measurements.* Piconodes sense the environmental conditions (e.g. temperature, humidity, light) and provide this data as a service.
– *Monitoring of environmental conditions.* Micronodes use the services offered by piconodes for monitoring the environmental conditions.
– *Localization service.* Beacons provide the localization service for carts placed on the expedition floor.
– *Monitoring of carts placement.* Groups of beacons adjacent to the grid cells of a specific shop monitor the positions of carts assigned to that shop.

*Office.* The second scenario is related to the concept of ”Smart Surroundings” [1]. Smart devices and sensors are fitted in office buildings, as well as carried by employees and visitors, thus forming a heterogeneous WSN. The following services are provided by this WSN:

– *Climate control*. Groups of sensors and actuators can collect data and actuate heating, ventilating, and air-conditioning devices to meet specific environmental parameters.

– *Localization and guidance*. In case of events such as conferences, visitors can be guided towards the places of interest by the fixed infrastructure present in the building.
– *Contact search*. Colleagues, friends, persons with common interests can be discovered through the WSN.

### 1.2 Related work

Service discovery has been extensively studied in the fields of local area, wide area, as well as in ad-hoc networks [12]. Nevertheless, resource-awareness has not been yet a field of great interest in the context of service discovery. The basic mechanisms used by protocols designed for ad-hoc networks imply one of the following: delegation of workload, flooding, maintaining several overlays or maintaining extensive topology information. We will give examples from each of these categories.

Delegating the workload to powerful devices is suitable for networks where these devices are always available. For example, small devices in FRODO [16] only implement a part of the protocol stack and they depend on powerful devices to store and process the service registry.

Some service discovery protocols piggyback on the routing messages to issue service request and get replies. Frank and Karl [7] relies on AODV [6], Wu and Zitterbart [18] uses DSR [8]. However, these two routing protocols use flooding to construct routes to destination. Due to the substantial power consumption inherent to this procedure, flooding is an undesirable feature for scalable and resource-aware routing protocols and consequently, for service discovery protocols that rely on them.

Service discovery protocols that build a dominating set which act as a distributed directory are more appropriate to be used in the context of sensor networks. Kozat and Tassiulas [10] build a backbone to which devices register their services. Their criteria for choosing the nodes that become part of the backbone is the maximum node degree among the 1-hop neighbors, together with a stability constraint concerning the link failure frequency. Services are registered to one or more nodes from the dominating set and service discovery messages are forwarded to members of the backbone. Due to the high density of nodes in the backbone, lots of loops are generated when a service discovery message travels the backbone nodes. To overcome this drawback, a source-based multicast tree additional algorithm is proposed on top of the dominating set. However, building and maintaining two overlays for the same purpose (the dominating set and the multicast tree) is expensive and unfeasible for resource-constraint sensor nodes.

Lenders et. al [11] propose a service discovery protocol inspired by electrostatic fields from physics. Nodes in the ad-hoc network determine the *potential* of a service depending on the distance to service providers. A service request packet arrived at a node is forwarded to the neighbor with the highest potential. This proactive approach is not well suited for sensor networks, because service advertisements are propagated through the whole network and nodes have to maintain potentials for every advertised service, which is both energy and memory consuming.

Several clustering algorithms have been proposed to support scalable routing in large ad-hoc networks. McDonald and Znati [14] describe an $(\alpha, t)$ clustering algorithm, taking the node mobility as the criteria for cluster organization. The cluster internal paths are expected to be available for a period of time $t$ with a probability of at least $\alpha$. Each node is aware of the complete intra-cluster topology information, which may exceed the node capabilities.

The algorithm proposed by Amis et. al [2] uses the d-hop information for clusterhead election. Each node initiates two rounds of flooding over $d$ hops for building the cluster membership. When the election algorithm finishes, nodes are at most $d$ hops away from the clusterhead. We consider that making decisions based on the complete information over $d$ hops leads to slow convergence, high maintenance overhead and memory consumption.

The DMAC algorithm [3] constructs and maintains an independent dominating set. They achieve fewer nodes in the distributed directory, compared to Kozat and Tassiulas [10], which in turn leads to fewer loops for the discovery phase. However, DMAC suffers from the *chain reaction* phenomenon, where a single topology change in the network may trigger significant changes in dominating set. For a distributed directory composed of nodes from the dominating set, the chain reaction leads to high overhead for maintaining consistent service registries.

## 2 Clustering algorithm

The clustering structure that we propose facilitates the construction and maintenance of a distributed directory. The clusterhead nodes keep a registry of the services available in their clusters. For minimizing the energy consumption during the discovery phase, service discovery messages are flooded among the clusterhead nodes, and not in the whole network.

We consider a wireless network, where two nodes $u$ and $v$ are neighbors if there is a direct communication channel between $u$ and $v$. We assume that the lower layers (such as MAC) filter out asymmetrical links, so that we can rely on bidirectional communication. Each node is assigned (1) a unique hardware identifier, termed the *address* of the node, and (2) a weight, termed the *capability grade*, representing an estimate of the node's dynamics and the available resources. The higher the capability grade, the more suitable is the node for the clusterhead role. We assume that these weights are unique, as the node hardware identifier may be used to break ties.

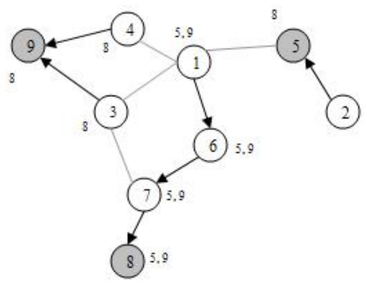
Our clustering structure is a *forest* composed of a set of *trees* or *clusters*. We assume that each node knows who its neighbors are and their capability grades from the lower network layers. We also assume that the underlying network layers provide a reliable, best-effort message delivery service. The clustering algorithm follows the idea of a greedy algorithm where nodes choose a neighbor with higher capability grade as *parent*, while other nodes that do not have such a neighbor are *roots*. The *height* of the cluster is the longest path from the root node to a leaf. We say that two trees are *adjacent* if there are two nodes, one from each tree, that are connected through a link.

### 2.1 Cluster setup

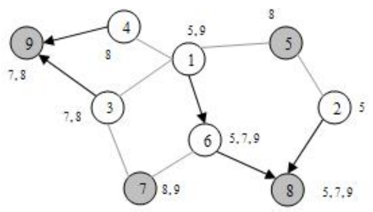
The algorithm constructs a set of trees, based on local knowledge of neighboring nodes. The protocol works as follows:

– Nodes that have the highest capability grades among their neighbors declare themselves clusterheads and broadcast a *SetRoot* message announcing their roles.
– The remaining nodes choose as parent the neighbor 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(a) shows an example network of nine nodes grouped into three clusters. The parent-child relationship is indicated by arrows. The dashed lines connect neighboring nodes that are in different trees (clusters).



(a) Network 1



(b) Network 2

## 2.2 Knowledge on adjacent clusters

Since the identity of the root node is propagated in the cluster down to the leaf nodes via the broadcast message *SetRoot*, this information also reaches nodes from adjacent clusters. These nodes store the adjacent root identity and report it to their parents. The 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 find out about all the adjacent clusters. Figure 1(a) shows the lists of adjacent clusters kept by each node in the forest.

## 2.3 Maintenance against topology changes

Nodes adjust their cluster membership according to topology changes:

– A node discovers a new neighbor with a higher capability grade than its current parent. The node then selects that neighbor as its new 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 neighborhood.

Besides reclustering, topology changes may also require modifications in the knowledge on adjacent clusters. The *UpdateInfo* message is used for transmitting the updates from children to their parents. We distinguish the following situations:

– A node $v$ detects a new neighbor from a different cluster. Consequently, $v$ adds the root of that cluster to its knowledge, together with the neighbor as the next hop for reaching the clusterhead.
– A node $v$ switches from parent $p_0$ to $p_1$. Then $v$ (1) sends the list of adjacent clusters to $p_1$ and (2) notifies $p_0$ to remove the information associated with $v$.
– A node $v$ detects the failure of the link to one of its neighbors $u$. As a result, $v$ erases the knowledge associated with $u$.
– Any change of global knowledge at node $v$ results in transmitting the message $UpdateInfo$ from $v$ to its parent.

Taking the example from Figure 1(a), node 8 moved to the neighborhood of nodes 6 and 2 (see Figure 1(b)). The clustering structure and the knowledge on adjacent clusters change accordingly.

## 3 Service discovery protocol

For efficiency and scalability reasons, we propose an integrated solution, in which the service discovery protocol uses the underlying clustering structure. The role of the clustering structure is to keep a distributed directory of service descriptions. Nodes register their services to 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.

### 3.1 Service registration

The registration works as follows:

– Every node $v$ registers its services and the services of the children to the parent node.
– In the case of a parent reselection, a node $v$ transfers the service registry to the new parent $p_1$, and notifies the old parent $p_0$ to purge the outdated service information. The process is transparent for the other nodes in the sub-tree rooted at $v$.
– If the overall service information at $p_0$ and $p_1$ changes due to the parent reselection, the modifications are propagated up in the hierarchy.

The service registration process can be easily integrated with the construction and maintenance of the clustering structure. Nodes are informed on the services provided by children through the same message $UpdateInfo$. The condition on which a node sends an $UpdateInfo$ message to its parent is when the overall knowledge on adjacent clusters or services its sub-tree has changed.

### 3.2 Service discovery

The service discovery process uses the distributed directory of service registrations. Suppose a node in the network generates a service discovery request $ServDisc$. The request is first checked against the local registrations. In the case where 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. Each node $v$ along the path checks the knowledge on adjacent clusters and picks a neighbor that has a path to the root. In the case where a link is deleted and $v$ cannot forward the $ServDisc$ message, it chooses another neighbor that provides a path to destination. If such a neighbor does not exist, $v$ informs its parent that it no longer has a route to the next cluster. The same procedure is repeated until all the paths to destination are tested. If the next cluster is not reachable, the root node erases the cluster from its knowledge.

Typically the result of a service search is the address of one or more service providers. This response can be returned by the first node that finds a match in its registry for the requested service. However, in certain situations it may be preferable that the service provider itself issues a reply for the service request. Examples include applications where service descriptions change frequently, or in cases where the reply incorporates more information than the address of the node. In these situations, the $ServDisc$ message is forwarded down the cluster until it reaches the service provider. In the case where the link to the service provider is deleted before the de-registration process, the service request is sent back to the root node who forwards it to the adjacent clusters.

The service discovery reply may follow the reverse cluster-path to the client, or any other path if a routing protocol is available. For the first case, if there is a cluster partition, the path can be reconstructed using the same search strategy as for the $ServDisc$ message, where this time the service is the address of the client.

## 4 Performance evaluation

For our experiments we use the OMNeT++ [17] simulation environment. We generate a random network, by placing $N$ nodes uniformly distributed on a square area of size $a \times a$, where $a = 500m$. We consider links to be bidirectional, so nodes have the same transmission range, $r$. There is a link between two nodes if the distance between them is less or equal to $r$. Each node chooses a capability grade from a uniform distribution. Static nodes have higher capability grades than mobile nodes.

We use a heartbeat broadcast message periodically sent by every node to maintain the neighborhood information. The heartbeat is also used for the cluster setup and maintenance, replacing the $SetRoot$ message for the propagation of root identity. The focus of our simulations is the overhead induced by the $UpdateInfo$ and $ServDisc$ messages.

### 4.1 Cluster 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 discovery protocol. The reason is that a high density of clusterheads leads to a high discovery cost.

For measuring the cluster density, we use the cyclic distance model for link formation, in order to avoid the border effects [4]. In this model, nodes at the border of the system area establish links via the borderline to the nodes located at the opposite side of the area. This setup approximates an area where nodes are distributed according to a Poisson point process [4] with constant density $\rho = N/a^2$. The expected node degree for a Poisson point process is:

$$D(\rho, r) = \rho\pi r^2 \tag{1}$$

After we randomly place the nodes on the simulation area, within a finite time interval the network converges to a valid cluster structure. We repeat the experiment for three values of the transmission range $0.1a$, $0.2a$ and $0.3a$, over at least 1000 topologies, and we count the number of clusters formed in each experiment. The mean of the samples are shown in Figure 1, with 5*th* and 95*th* percentile.

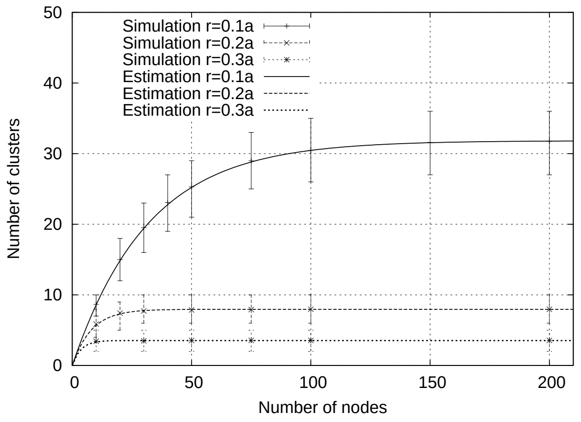


**Fig. 1.** Average number of clusters on $a \times a$ area, with 5*th* and 95*th* percentile.

In the first part of the curve the nodes are sparsely distributed on the simulation area. When the network becomes dense, the new nodes added either join the already existing clusters or they form their own cluster and force the root nodes in the neighborhood to join. We can notice in Figure 1 that from a certain number of nodes in the area, adding new nodes does not change the number of clusters. This behavior has a positive effect on the performance of the service discovery protocol, since the number of nodes in the distributed directory remains constant after reaching a certain network density.

We make the observation that the spatial distribution of the clusterheads belongs to the family of *hard-core point proccesses* [15] [4], in which the constituent points are forbidden to lie closer together than a certain minimum distance. We approximate the cluster density by using the *Matérn hard-core process*, which applies a thinning to a stationary Poisson process such as every retained node $x$ has the highest weight in the circle with radius $r$ centered at $x$. The estimated number of clusterheads for a the Matérn process is the following:

$$E_{clusters}(N, r, a) = \frac{a^2}{\pi r^2}(1 - e^{-\frac{N\pi r^2}{a^2}}) \tag{2}$$

In addition, we can compute the limit on the estimated number of clusters, as $N$ grows to infinity:

$$c = \lim_{N\to\infty} E_{clusters}(N, r, a) = \frac{a^2}{\pi r^2} \tag{3}$$

We define an area as being *saturated* if $|E_{clusters}(N, r, a) - c| \leq \epsilon$, where $0 < \epsilon \ll 1$. From Eq. 2 and 3, we compute the minimum number of nodes on a saturated area:

$$N \geq c \ln(\frac{c}{\epsilon}) \tag{4}$$

As a numerical example, for $a = 500m,\ r = 100m$, a saturated area with $\epsilon = 0.1$ is reached from $N \geq 35$. In other words, increasing the number of nodes above 35 does not influence the cluster density.

We plotted the theoretical estimations for the three values of the transmission range $r$. Figure 1 shows that the estimated values match exactly the simulation results.

## 4.2 Cluster height

In the next set of experiments we investigate what is the average cluster height for the same setting explained in Section 4.1. Figure 2 shows the results depending on the expected node degree, with the 5*th* and 95*th* percentile of the samples. We can notice that for all the three transmission ranges that we choose, the points follow the same curve. Consequently, our first conclusion is that the cluster height does not depend on the number of nodes, but it depends only on the density of the network. The second conclusion is that the average cluster height is lower than 2, and in 95% of the cases the clusters have the height less than or equal to 3. This means that we can achieve relatively small height clusters without imposing a maximal hop diameter limit, which would increase the maintenance effort and generate the chain reaction effects.

## 4.3 Role ratio

We represent in Figure 4.3 the ratio of clusterheads, parents and leaves depending on the expected node degree. We notice that the average role ratio is only a function of the expected number of neighbors. When $D(\rho, r) = 0$, the number of root nodes equals the number of nodes in the network. For dense networks, the ratio of clusterheads and parent nodes gets close to zero, while the ratio of leaf nodes grows asymptotically to 1.

## 4.4 Discovery cost

We analyze the discovery cost as a function of the number of nodes in the network. The nodes are uniformly distributed on the bounded simulation area of size $a \times a$, with

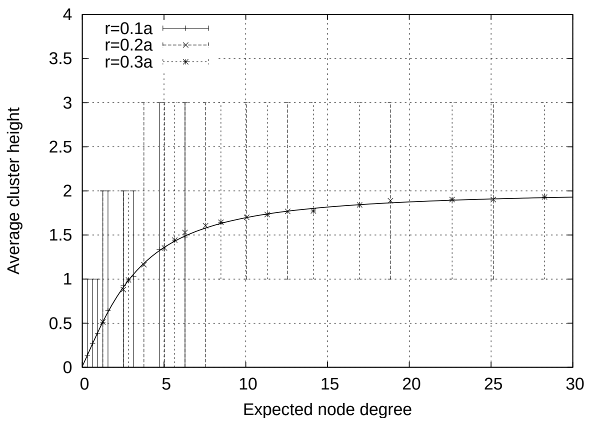


**Fig. 2.** Average cluster height with 5*th* and 95*th* percentile.

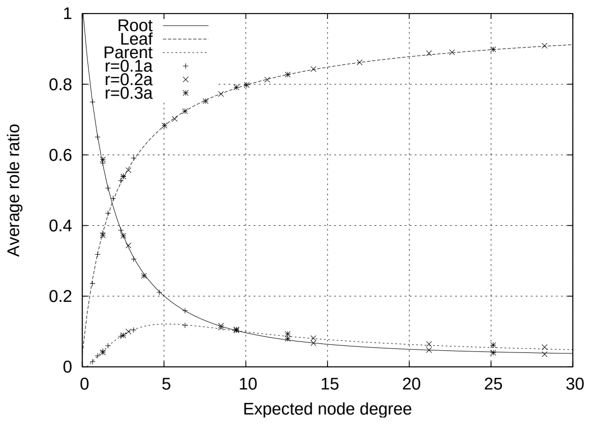


**Fig. 3.** Average ratio of roots, parents and leaves.

transmission range $r = 0.2a$. We adopt a uniform repartition of services, where a service is provided on average by five service providers. After placing the nodes on the simulation area, we wait until the network reaches consistency and we randomly pick a node. The node generates a discovery request with a randomly chosen service. We compute the total number of $ServDisc$ messages during one discovery phase. We perform experiments over at least 100 different topologies and we calculate the average values.

We notice from Figure 4 that in the first part of the curve, the average number of messages grows proportionally with the number of nodes in the network. This behavior changes for networks with high density: results indicate an asymptotically bounded discovery cost. The reason is that the discovery protocol propagates messages at the level of clusterhead nodes, while from a certain network density, the number of clusterheads remains constant (see Section 4.1).

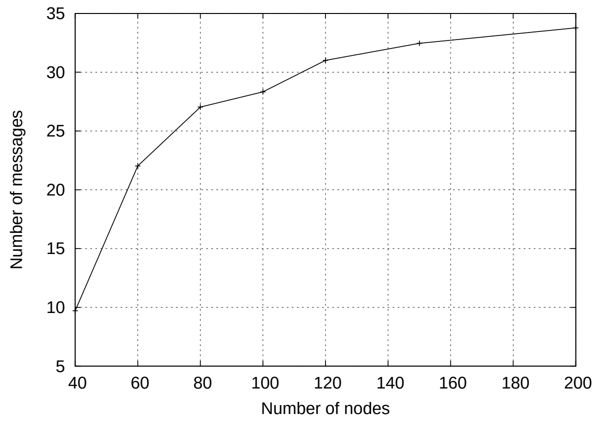


**Fig. 4.** Average number of $ServDisc$ messages exchanged during one service discovery phase.

Figure 5 gives a rough comparison on the number of messages spent during the service discovery phase, when using the proposed discovery protocol and flooding. We estimate the number of received messages during a complete flood as:

$$E_{flood}(N, \rho, r) = D(\rho, r)N \tag{5}$$

## 4.5 Maintenance overhead

We study the impact of the network density over the maintenance overhead (number of $UpdateInfo$ messages) in a dynamic network. We consider that 50% of the nodes are mobile and they move according to a simplified version of the random waypoint mobility model [9], that closely match the presented scenarios. At the beginning, nodes are randomly placed on the simulation area and they stay there for a specified period of

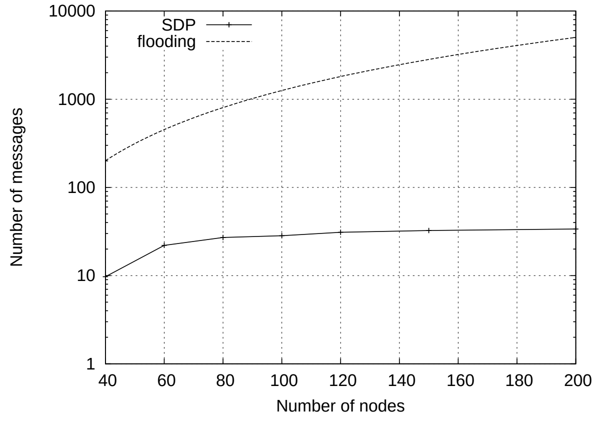


**Fig. 5.** Number of messages spent during a service discovery phase in comparison to flooding.

time. When the time expires, they choose a random destination and start moving toward the destination with a constant speed of $1m/s$ (the speed of a walking person). Upon arrival at the destination, nodes pause for 30 seconds before restarting the process. Due to the initialization problems that characterize the random waypoint mobility model [5], we discard the initial 1000 seconds of simulation time in each simulation trial and we count the number of $UpdateInfo$ messages for the next 1000 seconds.

Figure 6 shows the number of messages exchanged per node in 1000 seconds, as an average over at least 50 simulations. We are mainly interested in the maintenance overhead for dense networks. We recall from Section 4.3 that for dense networks, the large majority of nodes are leaves. Moreover, the mobile nodes are mainly leaf nodes for our simulation case, due to the fact that they have lower capability grades. Let $v$ be a static node in a dense network. A new child of $v$ is most probably a mobile leaf node, which sends an $UpdateInfo$ message containing information on adjacent clusters. However, since the network is dense, $v$ already has this information, so it does not forward the message to its parent. Consequently, in dense networks, the information received from a new neighbor node may not change the knowledge on adjacent clusters, so it does not generate the additional overhead necessary for sparse networks. We can see from Figure 6 that the average number of maintenance messages per node slowly decreases after reading a certain network density.

## 4.6 Load per capability groups

We next investigate how the capability grades influence the energy consumption during the maintenance and discovery phases. We sort the nodes in ascending order depending on the capability grades and we group them in 10 classes such that the weakest nodes belong to the first class and the most powerful nodes fit in the last class. Considering that we have 100 nodes in the network, each capability group has 10 member nodes. The first five groups of nodes are static, while nodes in the last groups are mobile.

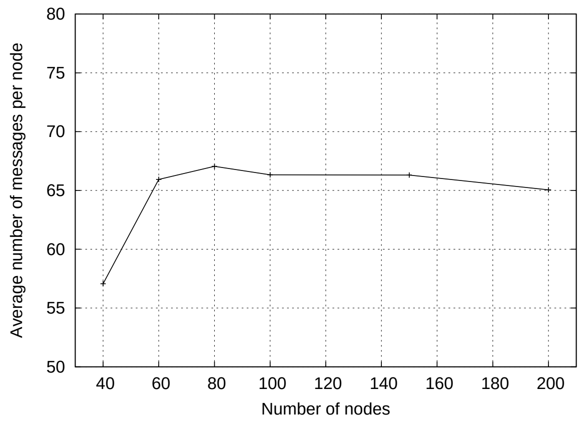


**Fig. 6.** Average number of $UpdateInfo$ messages exchanged per node during 1000 seconds of simulation time.

We are first interested in the distribution of maintenance overhead. In Figure 7 we represent the number of $UpdateInfo$ messages sent and received on average by a node in 1000 simulation seconds, depending on the capability group it belongs. We notice that the least overhead is experienced by the two weakest groups of stationary nodes, most probably small sensors which are part of the static network. The mobile nodes, have a higher maintenance overhead, due to frequent parent re-selections. The last two groups of nodes have the highest overhead, since they represent the static powerful nodes in the network.

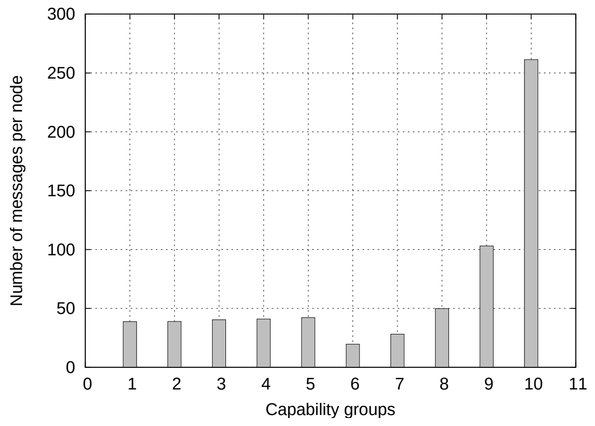


**Fig. 7.** The average number of maintenance messages per groups of capabilities.

In the following, we are interested in the division of the discovery overhead among the capability groups. During 1000 seconds of simulation time we issue 10 random service requests, with a delay of 100 seconds. Figure 8 shows the number of service discovery messages *ServDisc* send and received on average by every node from a capability group during one service discovery phase. We notice that nodes with higher capability grades spend more energy during discovery.

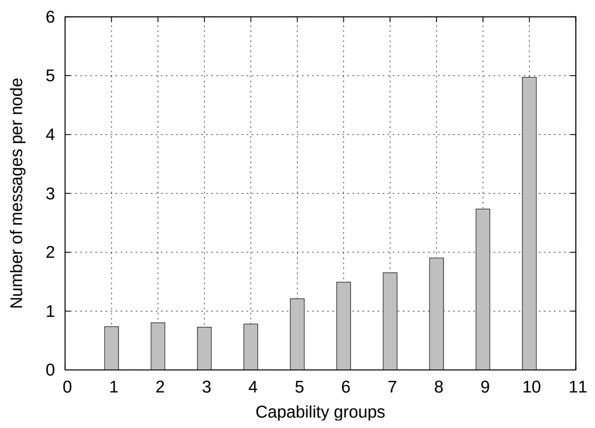


**Fig. 8.** The average number of service discovery messages per groups of capabilities.

## 5 Conclusions

This paper proposes a service discovery protocol for a heterogeneous wireless sensor network environment. The protocol relies on a clustering structure, which is a solution for the scalability problem as it achieves restricted reconfiguration, distributed knowledge and it avoids network flooding. The clusters are set up depending on the available resources and dynamics of nodes. The algorithm exploits the heterogeneous nature of the network by assigning higher tree levels to more powerful nodes. The simulation results show that the protocol performance degrades gracefully when increasing the network density. Moreover, the protocol delegates more workload to powerful nodes while relieving the mobile and stationary weak devices.

Our plan for the future is to avoid the possibility of overloading the root and parent nodes with service registrations. The idea is that nodes that reach their memory limit can decrease the capability grade. This dynamic adjustment of capabilities depending on the context is expected to improve the resource-awareness and to provide even better performance. Another research interest concerns directing the discovery messages towards the most suitable adjacent cluster. For that, we plan to integrate location within the service discovery protocol in order perform geographic routing at the cluster level.

## References

1. Smart surroundings. http://wwwes.cs.utwente.nl/smartsurroundings/.
2. Alan D. Amis, Ravi Prakash, Dung Huynh, and Thai Vuong. Max-min d-cluster formation in wireless ad hoc networks. In *INFOCOM (1)*, pages 32–41, 2000.
3. Stefano Basagni. Distributed clustering for ad hoc networks. In *ISPAN '99*, page 310, Washington, DC, USA, 1999. IEEE Computer Society.
4. C. Bettstetter. *Mobility Modeling, Connectivity, and Adaptive Clustering in Ad Hoc Networks*. PhD thesis, Technische Universität München, Germany, October 2003.
5. T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *WCMC: Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.
6. Santanu Das, Charles E. Perkins, and Elizabeth M. Royer. Ad hoc on demand distance vector (AODV) routing. Internet-Draft Version 4, IETF, October 1999.
7. Christian Frank and Holger Karl. Consistency challenges of service discovery in mobile ad hoc networks. In *MSWiM '04*, pages 105–114, New York, NY, USA, 2004. ACM Press.
8. D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison-Wesley, 2001.
9. David B Johnson and David A Maltz. Dynamic source routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.
10. 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.
11. Vincent Lenders, Martin May, and Bernhard Plattner. Service discovery in mobile ad hoc networks: A field theoretic approach. *Pervasive and Mobile Computing*, 1:343–370, 2005.
12. 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.
13. Raluca Marin-Perianu, Hans Scholten, and Paul Havinga. CODE: A description language for wireless collaborating objects. In *Intelligent Sensors, Sensor Networks & Information Processing Conference (ISSNIP)*, pages 169–174. IEEE Computer Society Press, December 2005.
14. A. McDonald and T. Znati. A mobility-based framework for adaptive clustering in wireless ad hoc networks. *IEEE JSAC*, August 1999.
15. Dietrich Stoyan, Wilfrid S. Kendall, and Joseph Mecke. *Stochastic Geometry and its Applications*. John Wiley and Sons, 1995.
16. V. Sundramoorthy, J. Scholten, P. G. Jansen, and P. H. Hartel. Service discovery at home. In *4th Int. Conf. On Information, Communications & Signal Processing and 4th IEEE Pacific-Rim Conf. On Multimedia (ICICS/PCM)*. IEEE Computer Society Press, December 2003.
17. A. Varga. The omnet++ discrete event simulation system. In *ESM'01*, Prague, Czech Republic, June 2001.
18. J. Wu and M. Zitterbart. Service awareness in mobile ad hoc networks. Boulder, Colorado, USA, March 2001. Paper Digest of the 11th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN).