DANDi: Dynamic Asynchronous Neighbor Discovery Protocol for Directional Antennas

Nicolás Gammarano*, Javier Schandy† and Leonardo Steinfeld‡
Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay
Avenida Julio Herrera y Reissig 565, 11300, Montevideo, Uruguay
*ngammarano@fing.edu.uy, †jschandy@fing.edu.uy, ‡leo@fing.edu.uy

***Abstract*—In this paper, we propose DANDi (Dynamic Asynchronous Neighbor Discovery Protocol for Directional Antennas), a neighbor discovery protocol for Wireless Sensor Networks (WSN) with directional antennas that guarantees that every communication link in a network is discovered. DANDi is asynchronous, fully directional (supports both directional transmissions and receptions) and has a dynamic contention resolution mechanism so no network topology information is needed in advance. It was implemented in Contiki, an open-source operating system for WSN and the Internet of Things, and extensively tested using the COOJA network simulator with Tmote Sky nodes equipped with 6-sectored antennas. The neighbor discovery times are deeply analyzed and analytical expressions for these times are presented. The DANDi protocol performance is assessed through simulations and compared with SAND, the state of the art protocol for this kind of networks. Our experiments based on simulations show that the discovery time is reduced 19 % for networks with no collisions, and more than four times in average for unevenly dense networks. To the best of our knowledge, DANDi is faster than any other protocol in the state of the art with the great advantage of being able to discover every node in a network without requiring any prior information.**

***Index Terms*—Wireless sensor networks, neighbor discovery, sectored antennas, directional antennas.**

I. Introduction

Directional antennas offer several benefits when used for Wireless Sensor Networks (WSN). The increased range and the reduction of the interference with neighbor nodes make this antennas very useful for several applications. Electronically Switched Directional (ESD) antennas are one kind of sectored antennas that are the most widely used for WSN. These antennas have the capability of dynamically selecting the direction of transmission by switching an electronic circuit (generally RF switches) to concentrate the radiation in K different directions. Their simplicity, reduced size and reduced cost make them suitable for large deployments of sensor nodes. Fig. 1 shows an ideal K-sectored antenna.

The benefits of this kind of antennas for wireless communications are well known, but to take advantage of them in WSN, it is necessary to make some changes to the network protocols. One of the first problems that arises when using directional antennas is how to discover all the possible communication links with neighbor nodes. When using omnidirectional antennas, one single broadcast message is enough to query neighbor nodes, and the ones in range may reply including their network address. But with ESD antennas, there may be many sector-to-sector (S2S) links between two nodes. Besides, the transmitter must know when its neighbor is pointing to a certain direction.

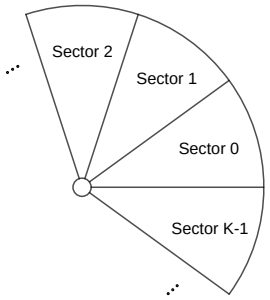


Fig. 1: K sectored-antenna.

Several neighbor discovery protocols have been proposed to solve this problem when using directional antennas. Some of them rely on omnidirectional antennas to assist the neighbor discovery process [1], [2], some of them rely on time synchronization [3], [4], some of them are based on a probabilistic approach [5], while others serialize the discovery process in time [6], [7]. The latter are convenient for WSN, as they are fully directional and do not rely on time synchronization that is difficult to achieve in this kind of networks. One example is the SAND protocol, that uses a serialized mechanism that guarantees that each node discovers sequentially all its neighbor nodes in a bounded time. One of the main disadvantages of this protocol is that it takes a considerable amount of time, and this time grows linearly with the number of nodes in the network. Another important disadvantage of SAND is that a certain knowledge of the network topology is needed in advance to optimize the parameters of the protocol.

In this work we propose a Dynamic Asynchronous Neighbor Discovery protocol for Directional Antennas (DANDi). The main contributions of this protocol are that: i) it is fully directional as it does not rely on omnidirectional antennas, ii) it guarantees every node in the network to discover all the possible communication links with its neighbor nodes, iii) it is faster than any other protocol in the state of the art, iv) it is dynamic, so no network topology information is needed beforehand to optimize the protocol and v) the neighbor information can be collected centrally to enable the sink node

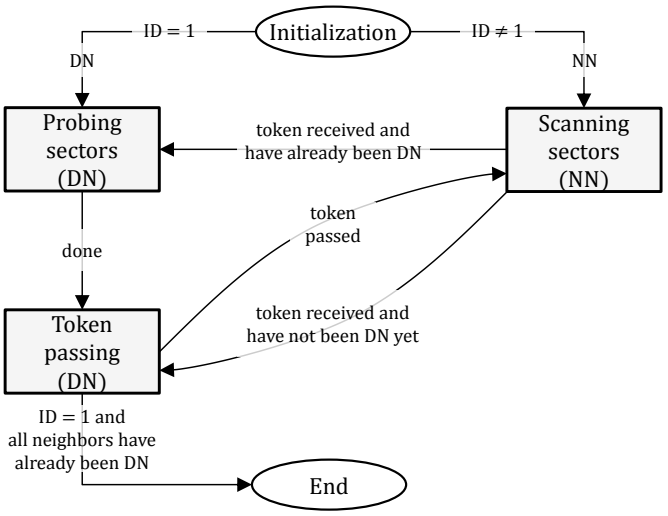


Fig. 2: State diagram: node roles and main process.

to process it and take, for example, routing decisions.

The rest of the work is organized as follows. In Section II, we describe the proposed protocol, its parameters, time restrictions and we find an expression for the duration of the protocol. Section III presents some implementation details of the protocol. The simulations results are presented in Section IV, including a comparison with SAND protocol. We conclude and state future work in Section V.

II. DANDi PROTOCOL DESCRIPTION

The main goal of a neighbor discovery protocol is to enable every node in a network to gather information about its respective neighboring nodes. If we consider nodes with sectored antennas, the essential purpose is to ensure the discovery of all S2S links between nodes in the network. Considering K-sectored antennas, there are K^2 possible sector combinations between two neighbor nodes: K possible sectors of one node and K possible sectors of the other.

The main idea of DANDi protocol is that one single discoverer node (DN) at a time discovers all S2S links with its neighbor nodes (NN) by transmitting probe messages repeatedly, while the remainder network nodes listen for incoming messages in each sector sequentially.

The DN probes one sector a time using the *probe-reply with dynamic contention resolution mechanism* described below. Once the DN finishes discovering nodes in one sector, it continues with the next one. When the DN ends discovering neighbors in its K sectors, it passes the DN role to a NN through the *token passing mechanism*.

Fig. 2 shows a state diagram representing both roles (DN and NN) and the main processes. In this case the node ID is used to select the initial role, so the first DN is the node with ID 1. The discovery process ends when the DN role is taken again by the node that started as DN, and all of its neighbors have already discovered their own neighbors. The above mechanisms are deeply explained in the next sections.

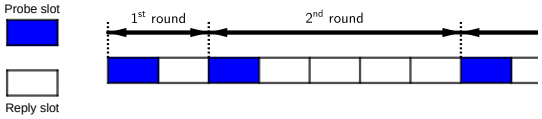


Fig. 3: Slots and rounds.

A. Probe-reply with dynamic contention resolution

The messages exchanged between DN and NN are organized in rounds. The probe message sent by the DN delimits the start of a round. A *round* is composed by a *probe slot* and one or several *reply slots*. A *probe message* is sent at the beginning of the *probe slot*. After the probe slot follows a reply round composed of one or more reply slots. The number of reply slots is specified in the leading probe message. The probe message also includes other information, such as the list of nodes discovered in the last round. Fig. 3 shows a first round with one reply slot followed by a second round with many reply slots.

The probe-reply with dynamic contention resolution algorithm involves the DN and all the NN in the network. Depending on its actual role, a node executes the DN part of the algorithm (*probing sectors*) or the NN part (*scanning sectors*).

1) Probing sectors algorithm: Fig. 4 shows the probing sectors algorithm of the DN for discovering neighbor nodes sequentially starting at sector 0.

The probing of a sector starts by selecting an antenna sector and locking it at that position until it discovers all the neighbors at that sector.

The DN initializes the number of reply slots R_{slots} equal to an initial predefined value (e.g. equal to one). Then it sends the probe message of the first round and waits for a reply in each slot during the time t_{slot}. In each reply slot, there are four possible situations shown in Fig. 5 and summarized below:

a) No answer.
b) A single node replies.
c) Many nodes reply, and one frame prevails over the others due to the capture effect.
d) Many nodes reply, and results in a collision.

The first case implicates that no neighbor is able to receive at that sector at that time. From the point of view of the DN, the second and third cases are indistinct, and the DN needs to give the chance to any node that had a suppressed message due to the capture effect, to send its reply again. The capture effect occurs when a node receives two or more messages almost at the same time, and it is able to demodulate the message with higher signal strength, when some conditions are met [8]. The last case, in which the DN detects a collision, all the replies in that slot are lost. A receiver-side collision detection can be implemented using CRC or CCA-based techniques [9], adopted in some other protocols [10]. Summarizing, more rounds are needed to ensure that all NN are discovered correctly for every case except the first one.

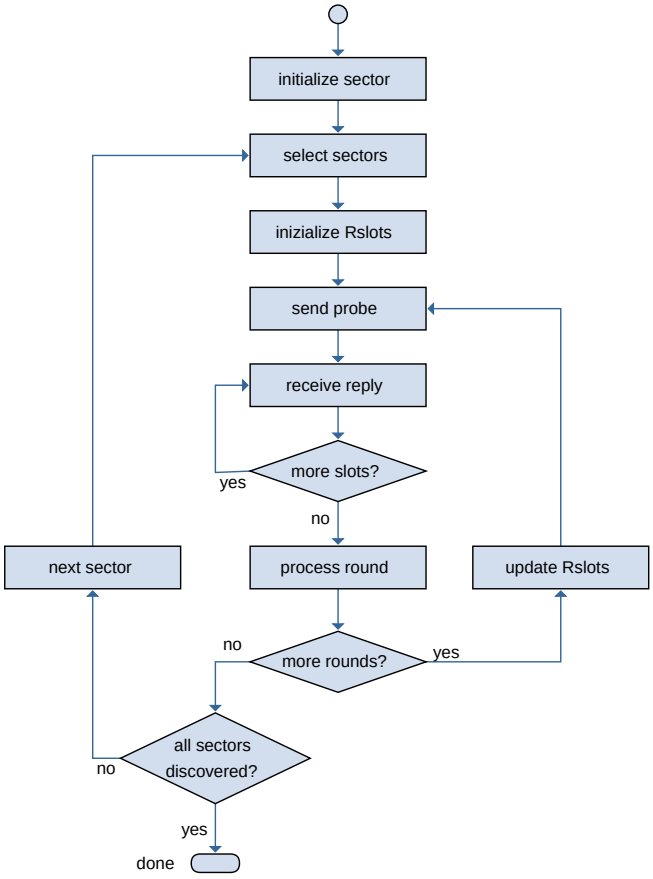
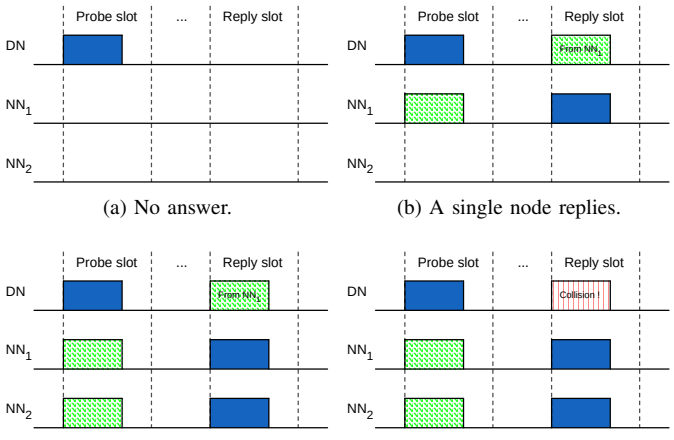


Fig. 4: Probing sectors algorithm.



(a) No answer.

(b) A single node replies.

(c) Many nodes reply: capture effect.

(d) Many nodes reply: collision.

Fig. 5: Possible results for a reply slot.

When the reply round completes, the result is processed to determine whether a new round is initiated or not. This process consists in analyzing the following cases:

1) No answer in any slot.
2) A reply or a collision happens in any slot.

The second case needs a new round to resolve either the collision/s or give the chance to any potential suppressed message to be sent again by the corresponding node. In any of these situations, the DN starts a new round. In this case the DN selects the number of reply slots of the next round R_{slots} based on the results. If there is a collision in any slot, the DN could increment the number of slots to speed up the contention resolution. If there are no collisions the number of reply slots could be kept unchanged or even reduced. In this work we adopted a simple yet effective solution that is *exponential back-off* [11]. The DN acknowledges the replies in the probe message of the following round, by including the NN identification in the list of discovered nodes.

In the first case (no answer in any slot), only after one round with no replies from the NN, can the DN pass the token to the next node. However, if there is a round with more than one reply slots for contention resolution, there will be no probe slots for a certain time (determined below) since the nodes are resolving the contention. To ensure that the missing slots will not lead a neighbor to miss a probe, the missing probe slots must be recovered. To achieve this, the DN must add a certain number of rounds with a single reply slot (N_{probe}, determined in Section II-C), and receive no answer in any of them in order to change sector.

Only after this process is completed, can the DN continue with the next sector. When the DN finishes discovering all the sectors, it passes the token to a NN that has not discovered neighbors yet.

2) Scanning sectors algorithm: Fig. 6 shows the scanning sectors algorithm of the NN, where the scan performed along all sectors can be observed. The NN selects one sector and listens for incoming messages during a certain period of time t_{switch}. If the NN does not receive anything, it continues with the next sector. If it receives a probe message from the DN, the NN processes the message. It verifies in the probe message whether it is in the list of discovered nodes or not. If it is not in the list, the NN randomly chooses a slot and sends a reply message.

If the NN is in the list, it means that the DN received the previous reply correctly and so the NN continues scanning sectors. The NN also continues with the next sector if it had already received a probe sent by this DN from the same sector. This is why probe messages need to include the sector of the DN.

If the NN receives at any time a token message instead of a probe, it stops scanning sectors and assumes the DN role.

B. Token passing mechanism

After a DN finishes discovering neighbors in all of its sectors, it passes the discoverer role to a NN that has not discovered neighbors yet. To achieve this, a token identifies

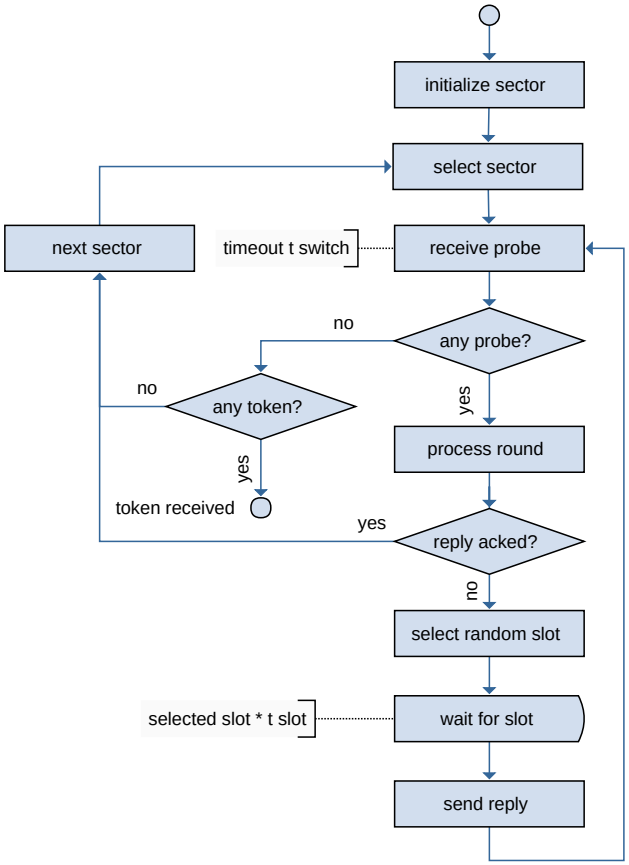


Fig. 6: Scanning sectors algorithm.

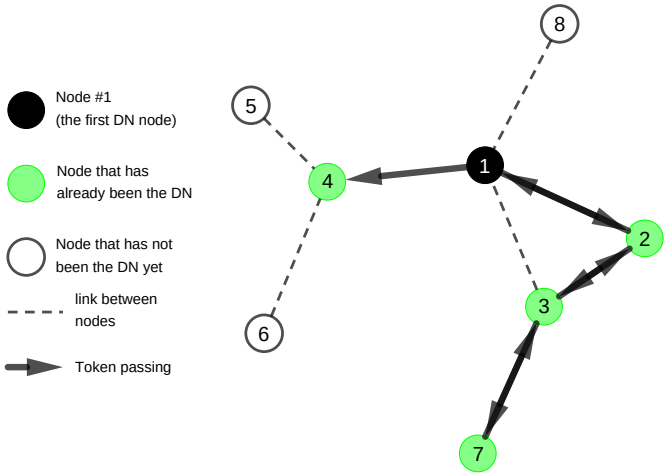
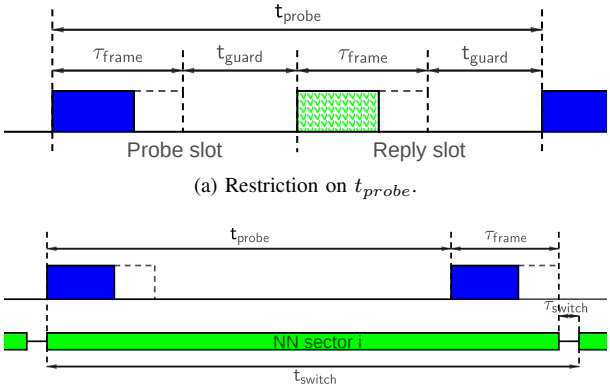


Fig. 7: Partial tree started by node #1. Nodes #2, #3 and #7 have already been the DN, while node #4 is the current DN. Nodes #5, #6 and #8 have not been the DN yet.



(a) Restriction on t_{probe}.

(b) Restriction on t_{switch}.

(c) Restriction on N_{probe}.

Fig. 8: Time restrictions.

the node with the DN role, and a special message is used to pass the token to another node. In the case that all the neighbors of a DN have already discovered their neighbors, the DN passes the token to the node from which it received the token initially. Since the NN are scanning sectors, the DN locks its antenna in the direction of the node that will receive the token message, and sends repeatedly probe messages until it can guarantee that the neighbor node received it. For the NN to be able to receive at least one probe per sector, the restriction is the same as when probing a sector. After having sent the probe messages, the DN sends the token and waits to receive an acknowledgment.

In this way the token passing process between nodes gradually forms a directed rooted tree, where the node that initiates the discovery process is the tree root, the remainder vertexes are the nodes that have already been in the DN role, and the edges represent the token passing relation.

Fig. 7 depicts the neighbor discovery performed by each DN, token passing between nodes, and the formed tree.

C. Probe, reply and switching time restrictions

The probe slot duration has some restrictions. Considering that probe and reply messages are at most of the maximum frame length, then the duration of these messages are limited by a given time τ_{frame}. The period for sending probe messages is t_{probe}. A guard time t_{guard} between slots may be necessary for turning around and switching from transmitting to receiving or vice-versa. Then, the probe period must satisfy

$$t_{probe} > 2\left(\tau_{frame} + t_{guard}\right). \tag{1}$$

Fig. 8a shows the minimum probe period required to allocate a probe or a reply and the actual time used by the DN probe and a NN reply.

NN switch asynchronously their active sector every t_{switch}, scanning for incoming messages. Note that t_{switch} is the period for switching the active sector, but while switching,

there is a time τ_{switch} which accounts for the hardware switching time, where the node cannot receive anything.

To ensure that the DN neighbor nodes are able to receive at least one probe message, the following condition must be satisfied:

$$t_{switch} > t_{probe} + \tau_{frame} + \tau_{switch}. \quad (2)$$

Fig. 8b depicts the time involved in the probing process and shows the minimum t_{switch}.

In order to send enough probe messages so that every NN has the opportunity to receive at least one of them, the total time sending probes at one sector must be larger than the time of one "full turn" scan of the neighbors:

$$N_{probe} \cdot t_{probe} > K \cdot t_{switch}, \quad (3)$$

where N_{probe} is the number of probes sent per sector. This relationship can be observed in Fig. 8c.

Hereinafter, for the sake of simplicity, we define a generic time slot for sending probe and reply messages of duration $t_{probe} = t_{switch}/2$, and long enough to satisfy the above restrictions. Note that during the time t_{switch} there is enough time for a probe slot and a reply slot.

Finally, to minimize the duration of the overall neighbor discovery process, the number of messages N_{probe} must be also minimized. Then considering Eq. (2) and (3), we obtain

$$N_{probe} > K \left(1 + \frac{\tau_{frame} + \tau_{switch}}{t_{probe}}\right). \quad (4)$$

D. Neighbor discovery time analysis

The discovery process forms a directed rooted tree, whose edges represent the passage of the token between nodes. The network has n nodes (vertexes), and since it is a tree it has $n-1$ edges. At the end of the protocol, there have been $2(n-1)$ token passing (two token passing per edge: one upwards in the tree and one downwards).

Taking this into consideration, the total time the protocol takes to complete is the sum of the time taken to discover neighbors by each network node through the *probe-reply mechanism* and the time taken to pass the token through the *token passing mechanism* (there are $2(n-1)$ in total):

$$T_{DANDi} = \sum_{i=1}^{n} \{T_{PR}(i)\} + 2(n-1) T_{TP}, \quad (5)$$

where $T_{PR}(i)$ is the time taken by the probe-reply mechanism of node i, and T_{TP} is the time taken by the token passing mechanism.

The time taken by the probe-reply mechanism of node i can be expressed as:

$$T_{PR}(i) = t_{slot} \sum_{j=0}^{K-1} \left\{ \sum_{k=1}^{R_{ij}} \{1 + s_{ijk}\} \right\}, \quad (6)$$

where R_{ij} is the number of rounds needed to complete the discovery process for node i for its active sector j, and s_{ijk} is the number of reply slots that has the k^{th} round, when node i is the DN and its active sector is j.

$T_{PR}(i)$ is probabilistic and not deterministic, given that the random choice of reply slot by the NN impacts on the number of suppressed messages or collisions, and then in the number of rounds needed to resolve the contention. It could happen that two NN choose the same reply slot for a lot of consecutive rounds, and as a result a lot of rounds would be needed. In this case the time $T_{PR}(i)$ would be greater than if the nodes had chosen different reply slots earlier. So the time $T_{PR}(i)$ will vary with i, expecting it to be greater for nodes with more neighbors. Even more, the time of probing sectors will vary from sector to sector for a given DN, being greater for sectors in which the DN has more neighbors.

In case that a DN has at most one neighbor per sector, there will be no collisions and $T_{PR}(i)$ will be the minimum possible time of a probe-reply mechanism duration. In this case $T_{PR}(i)$ is deterministic and it is given by the following equation:

$$T_{PR}(i) = 2 \cdot t_{slot} \cdot K \cdot N_{probe}. \quad (7)$$

The time taken by the token passing mechanism can be expressed in the following way:

$$T_{TP} = (h-1) t_{probe} + t_{token-ack}, \quad (8)$$

where N_{TP} is the number of messages sent from the current DN to the NN, and $t_{token-ack}$ is the time that takes to send the token and receive an acknowledgment.

III. Implementation and simulations

To assess the proposed protocol, we implemented it in Contiki OS [12], an operating system for wireless sensors nodes with constrained resources. We based the implementation in the Tmote Sky [13] platform and used the default 6LoWPAN protocol stack. In order to be able to test it with several nodes, we simulated a network of 16 nodes with 6-sectored antennas in the COOJA network simulator.

We defined the following types of messages according to the protocol: i) probe, ii) reply, iii) token, and iv) acknowledgment. These messages are sent using link-local addresses. Probe messages are *broadcast* (since their destination is any NN), while reply messages are *unicast*, addressed to the DN.

The parameters of DANDi used in the simulations are shown in Table I.

TABLE I: Parameters of DANDi protocol used in the simulations

DANDi	
Parameter	**Value**
t_{switch}	62.5 ms
t_{slot}	31.25 ms
N_{probe}	13

In this implementation, if a NN chooses the first reply slot, it replies immediately after having received the probe message. This is done to reduce the discovery time, as well as to keep the implementation simple. As a consequence, the probe slot

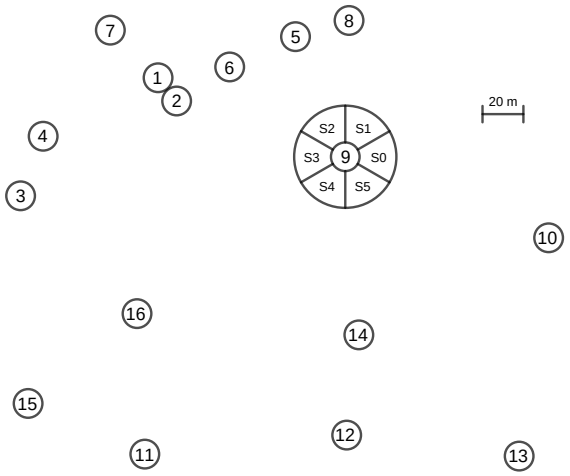


Fig. 9: Simulated network with 16 nodes. The sectors of all nodes are oriented as shown for node #9.

is very short compared to the reply slots, so the time taken by the probe-reply mechanism of node i is expressed as:

$$T_{PR}(i) = t_{slot} \sum_{j=0}^{K-1} \left\{ \sum_{k=1}^{R_{ij}} \{s_{ijk}\} \right\} \tag{9}$$

and the minimum possible time (in case of no collisions whatsoever) becomes:

$$T_{PR}(i) = t_{slot} \cdot K \cdot N_{probe}. \tag{10}$$

IV. Simulation results

To assess the protocol effectiveness, we simulated DANDi for different networks and in this section we show the obtained results.

We have to choose how the number of slots of each round evolves depending on the number of slots, the number of collisions and the number of discovered neighbors in the past round. A mathematical way to express this is through a function: $s_{r_i} = f\left(s_{r_{i-1}}, c_{r_{i-1}}, N_{r_{i-1}}\right)$. The function we used in the simulations is the following, based on exponential back-off as mentioned in Section II-A:

$$s_{r_i} = \begin{cases} 1 & \text{if } i = 0 \text{ or } c_{r_{i-1}} = 0 \\ 2s_{r_{i-1}} & \text{otherwise.} \end{cases} \tag{11}$$

This function strongly impacts in the time required by the protocol to complete. The optimization of the function used to determine the number of slots of a given round is out of the scope of this paper.

A. Performance evaluation

In the first place, we simulated the 16-node network shown in Fig. 9 with 25 different simulation seeds. In every case, there were 666 S2S links discovered in total.

The total time taken by the protocol was 1 min 31 s averaging the 25 simulations. The simulation that took the least time took 1 min 15 s, and the one that took the longest took 1 min 48 s.

TABLE II: Number of S2S links per sector of each node of the network

Node	S0	S1	S2	S3	S4	S5	Total
#1	20	18	12	16	15	15	96
#2	16	16	16	15	14	10	87
#3	12	12	6	3	2	7	42
#4	15	11	7	5	5	10	53
#5	7	5	12	13	16	11	64
#6	9	9	15	14	14	16	77
#7	16	8	0	6	13	17	60
#8	0	2	6	13	13	6	40
#9	1	6	12	8	2	3	32
#10	0	0	1	4	3	0	8
#11	1	3	5	4	2	0	15
#12	5	5	4	3	0	1	18
#13	0	1	3	5	1	0	10
#14	6	4	2	3	4	6	25
#15	6	7	3	0	1	3	20
#16	1	2	4	4	5	3	19

Table II shows the distribution of the links. As expected from the position of the nodes, we identify two unevenly dense zones in the network that generate different number of S2S links per node: a zone very dense near nodes #1 to #8 (with 40 to 87 S2S links per node) and a lesser dense zone near nodes #10 to #16 (with 8 to 25 S2S links per node). Node #9 is between both zones with 32 S2S. Another expected result is that the nodes that are on the edges of the network have no S2S links when pointing away from the network (i.e. the case of node #10 for sectors S0, S1, and S5).

TABLE III: Average time taken by the probe-reply mechanism of each sector of each node in the network.

Node	S0	S1	S2	S3	S4	S5	Total
#1	1.839 s	1.550 s	1.083 s	1.783 s	1.519 s	1.261 s	9.034 s
#2	2.543 s	1.340 s	1.415 s	1.438 s	1.421 s	1.201 s	9.358 s
#3	1.215 s	1.020 s	0.550 s	0.406 s	0.406 s	1.103 s	4.700 s
#4	1.596 s	1.096 s	0.611 s	0.406 s	0.406 s	0.863 s	4.979 s
#5	0.610 s	0.406 s	1.100 s	1.106 s	1.359 s	1.423 s	6.004 s
#6	0.888 s	0.785 s	1.299 s	1.759 s	1.414 s	1.968 s	8.111 s
#7	1.623 s	0.955 s	0.406 s	0.661 s	1.139 s	1.835 s	6.619 s
#8	0.406 s	0.406 s	0.486 s	1.278 s	1.311 s	0.713 s	4.600 s
#9	0.406 s	0.489 s	1.286 s	1.029 s	0.406 s	0.488 s	4.104 s
#10	0.406 s	0.406 s	0.406 s	0.504 s	0.481 s	0.406 s	2.610 s
#11	0.406 s	0.476 s	0.476 s	0.406 s	0.406 s	0.406 s	2.578 s
#12	0.421 s	0.458 s	0.406 s	0.492 s	0.406 s	0.406 s	2.590 s
#13	0.406 s	0.406 s	0.478 s	0.561 s	0.406 s	0.406 s	2.664 s
#14	0.609 s	0.428 s	0.406 s	0.439 s	0.406 s	0.490 s	2.778 s
#15	0.551 s	0.833 s	0.464 s	0.406 s	0.406 s	0.406 s	3.066 s
#16	0.406 s	0.406 s	0.525 s	0.538 s	0.626 s	0.450 s	2.951 s

Table III shows the average time taken by the probe-reply mechanism of each sector of each node in the network. We can see that for a sector with more S2S links to discover, it takes longer for the probe-reply mechanism to complete. This result was expected, since more S2S links in a sector generate more collisions which induce more rounds and thus more time. We can also see that the minimum time taken by the probe-reply mechanism per sector is 0.406 s. This corresponds with Eq. (10), since $T_{PR}(i)/K = t_{slot}N_{probe} = (31.25\,\text{ms}) \times 13 = 406.25\,\text{ms}$.

The average of the total values shown in Table II and Table III (last column in both tables) are graphically presented

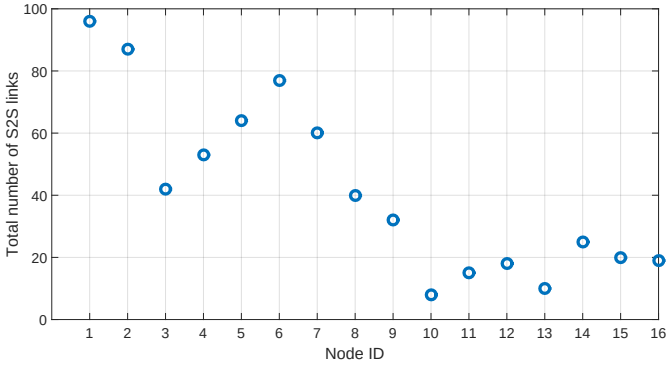


Fig. 10: Total number of S2S links per sector for each node of the network.

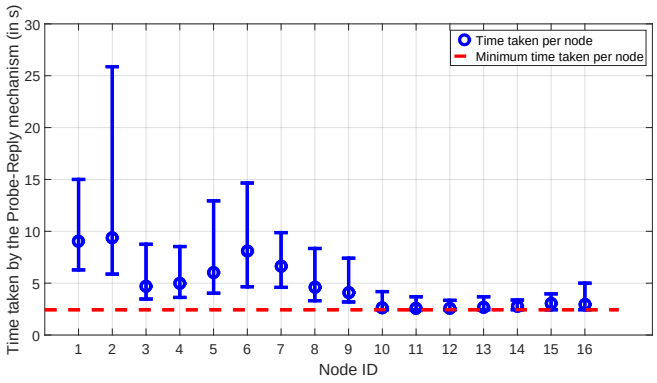


Fig. 11: Minimum, average and maximum time taken by the probe-reply mechanism for each node of the network.

in Fig. 10 and Fig. 11 respectively. Fig. 11 includes the minimum and maximum time taken along all the simulations to the average time (already present in the table).

We see that Fig. 10 and Fig. 11 have similar tendencies, confirming that the nodes that take longer to discover its neighbors are the ones that have more S2S links. We also see in Fig. 11 the minimum time taken per node (in case there are no collisions between reply messages), calculated using Eq. (10): $T_{PR}(i) = t_{slot} K N_{probe} = (31.25\,\text{ms}) \times 6 \times 13 = 2.4375\,\text{s}$. We observe that nodes #10 to #16 take very close to the minimum time.

B. Comparison with SAND

We also implemented and simulated SAND protocol for the same networks, in order to compare our proposed protocol DANDi to SAND. We chose SAND protocol given that it is a state-of-the-art fully directional neighbor discovery protocol, widely used for WSN applications [14]. Like DANDi, SAND is a token-based protocol in which a single node is discovering neighbors at a time, but with SAND, the number of slots s and rounds r are set in advance before running the protocol. The parameters of both DANDi and SAND protocols used in the simulations are shown in Table IV. In order for the comparison to be fair, we used the same protocol parameters and the same network topologies. We see from Table IV that DANDi protocol is simpler than SAND in that it has less parameters.

TABLE IV: Parameters of DANDi and SAND protocols used in the simulations

DANDi		**SAND**	
Parameter	**Value**	**Parameter**	**Value**
t_{switch}	62.5 ms	t_{switch}	62.5 ms
$t_{pretoken}$	31.25 ms	$t_{Hone\text{-}In}$	31.25 ms
N_{probe}	13	h	12
dynamic s and r		s	$s\,(N_{max}, 99\,\%)$
		r	$r\,(N_{max}, 99\,\%)$
t_{slot}	31.25 ms	t_{slot}	31.25 ms
- (not used)		$t_{GoToFastScan}$	31.25 ms

For SAND, the parameters s and r were optimized taking into account the maximum number of neighbors per sector N_{max} for each simulated network and a probability of discovering all of them of $p = 99\,\%$. The authors of SAND proposed an algorithm that given N_{max} and p allows to obtain s and r minimizing the protocol duration [6], [7]. We computed that algorithm in order to make a fair comparison between DANDi and SAND.

In the first place, we simulated the network shown in Fig. 9 with SAND for $s = 5$ slots and $r = 4$ rounds. If s or r were smaller, SAND would take less time to finish, but it would fail to discover all the 666 S2S links with a probability greater than 1 %. The total time taken by SAND was 6 min 46 s. This time is approximately 4.46 times more than the average time taken by DANDi. In the case of SAND, once the parameters s and r are fixed, the time the protocol takes is deterministic, and all the nodes in the network take the same amount of time. Even more, each sector of every node in the network takes the same amount of time to discover its neighbors.

It is clear from the results that for a random network with different number of neighbors in each active sector of each network node, DANDi outperforms SAND in regards to network discovery time. This is an expected result because, on one hand, if we choose for SAND a small number of slots s and rounds r (suitable for sectors with very few neighbors), the sectors that have more neighbors will very likely fail to discover all of them. On the other hand, if we choose a number of slots s and rounds r large enough (suitable for sectors with the largest number of neighbors in the network), we can make the probability of discovering all of those neighbors arbitrarily high, but the time taken by the protocol would be unnecessarily high for the sectors with few neighbors. In the case of DANDi, r and s are dynamic, in such a way that the time taken to discover neighbors on a sector is according to the neighbors of that sector. The only time overhead in DANDi is due to the added probe messages to recover probe gaps in case of contention (round with number of reply slots greater than one).

Besides reducing considerably the neighbor discovery time, DANDi does not require a previous knowledge of the network. Note that we compared DANDi to the better optimized version of SAND ($s = 5$ and $r = 4$), which was chosen considering the network topology. Since we want to discover all S2S links, even for the sectors with more neighbors in the network, it

Fig. 12: Simulated network with 16 nodes and no collisions.

is not sufficient to simply estimate the number of neighbors based on the network density as if the nodes were evenly distributed, but we need the topology information. In a real deployment, it is not very likely to have such information beforehand. Additionally, if the network changes (i.e. nodes are added, moved or removed), no changes are required for DANDi protocol. However, for SAND protocol, a change in the parameters s and r might be needed to re-optimize SAND, and all the nodes in the network would have to be reprogrammed with the new parameters.

Next, we proceeded to compare DANDi to SAND for the case of a network with no collisions whatsoever: every sector must have at most one neighbor. We created the network depicted in Fig 12, where each node is in the range of the consecutive nodes only. As there are no collisions, the better optimized version of SAND is with $s = 1$ and $r = 1$. We used those parameters of SAND for this network simulation.

For both protocols, the 30 S2S links were discovered. We obtained the same results (milliseconds of difference) for different simulation seeds, both for DANDi and for SAND. This indicates that in the case of a network with no collisions, DANDi duration is deterministic, as analyzed in Section II-D. DANDi took 51.26 s to discover all nodes, while SAND took 63.29 s. We can see that even in the case where DANDi performs worse, it outperforms SAND in terms of discovery time taking 19 % less.

Using Eq. (5), (8) and (10), and the parameters of DANDi in Table IV (assuming that $t_{token\text{-}ack} \approx 0$ or negligible), we obtain $T_{DANDi} = 50.25\,\text{s}$, confirming the theoretical equations deduced.

V. Conclusions and Future work

DANDi, a fully directional asynchronous and dynamic neighbor discovery protocol for wireless sensor networks is proposed, implemented and tested through extensive simulations. The results of the simulations show that all the S2S links in the network are discovered independently of the network topology, and that the sectors that have few neighbors take less time to discover than the sectors that have more neighbors. The comparison made with SAND, the state of the art protocol for this kind of networks, showed that DANDi takes from 19 % to 78 % less time to discover the network, without having to set any parameter depending on the network topology. This makes DANDi, to the best of our knowledge, the fastest neighbor discovery protocol in the state of the art for WSN with directional antennas, with the additional advantage of being able to discover 100 % of the communication links in a network without requiring any prior information.

Future work should include testing the protocol in a real network with real antennas to validate the design and compare with the simulations.

Acknowledgment

The authors would like to thank Fondo María Viñas for supporting this project (FMV_1_2014_1_104872).

References

[1] R. A. Santosa, B.-S. Lee, C. K. Yeo, and T. M. Lim, "Distributed neighbor discovery in ad hoc networks using directional antennas," in *Computer and Information Technology, 2006. CIT'06. The Sixth IEEE International Conference on*. IEEE, 2006, pp. 97–97.

[2] S. Zhang and A. Datta, "A directional-antenna based MAC protocol for wireless sensor networks," in *International Conference on Computational Science and Its Applications*. Springer, 2005, pp. 686–695.

[3] G. Jakllari, W. Luo, and S. V. Krishnamurthy, "An integrated neighbor discovery and MAC protocol for ad hoc networks using directional antennas," *IEEE Transactions on Wireless Communications*, vol. 6, no. 3, 2007.

[4] Z. Zhang and B. Li, "Neighbor discovery in mobile ad hoc self-configuring networks with directional antennas: algorithms and comparisons," *IEEE Transactions on Wireless Communications*, vol. 7, no. 5, 2008.

[5] S. Vasudevan, J. Kurose, and D. Towsley, "On neighbor discovery in wireless networks with directional antennas," in *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, vol. 4. IEEE, 2005, pp. 2502–2512.

[6] R. Murawski, E. Felemban, E. Ekici, S. Park, S. Yoo, K. Lee, J. Park, and Z. Hameed Mir, "Neighbor discovery in wireless networks with sectored antennas," *Ad Hoc Networks*, vol. 10, no. 1, pp. 1 – 18, 2012.

[7] E. Felemban, R. Murawski, E. Ekici, S. Park, K. Lee, J. Park, and Z. Hameed, "SAND: Sectored-Antenna Neighbor Discovery Protocol for Wireless Networks," in *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, June 2010, pp. 1–9.

[8] K. Whitehouse, A. Woo, F. Jiang, J. Polastre, and D. Culler, "Exploiting the capture effect for collision detection and recovery," in *Embedded Networked Sensors, 2005. EmNetS-II. The Second IEEE Workshop on*. IEEE, 2005, pp. 45–52.

[9] M. Demirbas, O. Soysal, and M. Hussain, "A singlehop collaborative feedback primitive for wireless sensor networks," in *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008, pp. 2047–2055.

[10] X. Ji, Y. He, J. Wang, W. Dong, X. Wu, and Y. Liu, "Walking down the STAIRS: Efficient collision resolution for wireless sensor networks," in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Apr. 2014, pp. 961–969. [Online]. Available: http://dx.doi.org/10.1109/INFOCOM.2014.6848025

[11] "IEEE Standard for Low-Rate Wireless Networks," *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)*, pp. 1–709, April 2016. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.2016.7460875

[12] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in *29th Annual IEEE International Conference on Local Computer Networks*, Nov 2004, pp. 455–462.

[13] *Tmote Sky Datasheet*, Moteiv Corporation, June 2006, rev. 1.0.2.

[14] A. Varshney, L. Mottola, M. Carlsson, and T. Voigt, "Directional transmissions and receptions for high-throughput bulk forwarding in wireless sensor networks," in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 351–364.