Design Automation for Embedded Systems
https://doi.org/10.1007/s10617-020-09236-4

Reducing neighbor discovery time in sensor networks with directional antennas using dynamic contention resolution

Nicolás Gammarano[1] · **Javier Schandy**[1] · **Leonardo Steinfeld**[1]

Received: 26 May 2019 / Accepted: 9 March 2020

Abstract
In this paper we present, simulate, and evaluate Dynamic Asynchronous Neighbor Discovery protocol for Directional Antennas (DANDi), a neighbor discovery protocol for Wireless Sensor Networks (WSN) with directional antennas that guarantees that every reliable 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 based on the detection of packet collisions, so no network topology information is needed in advance. We propose, implement and evaluate a mechanism to identify these collisions, both in simulations and with real nodes. DANDi is implemented in Contiki OS, an open-source operating system for WSN and the Internet of Things, and extensively tested. First using the COOJA network simulator and then with real Tmote Sky nodes equipped with 6-sectored antennas. The neighbor discovery times are analyzed and analytical expressions for these times are presented. The DANDi protocol performance is compared with Sectored-Antenna Neighbor Discovery, a state-of-the-art protocol for this kind of networks. Our experiments show that the discovery time can be reduced up to four times in average depending on the network topology, while discovering every reliable sector-to-sector link. To the best of our knowledge, DANDi is the fastest protocol that is able to discover every reliable link in a network without requiring any prior information of the network topology.

Keywords Wireless sensor networks · Neighbor discovery · Sectored antennas · Directional antennas · Collision detection

1 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 these 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 radio frequency 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. Figure 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

This work has been supported by Fondo María Viñas (Project FMV_1_2014_1_104872).

Nicolás Gammarano
ngammarano@fing.edu.uy

Javier Schandy
jschandy@fing.edu.uy

Leonardo Steinfeld
leo@fing.edu.uy

[1]Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República, Avenida Julio Herrera y Reissig 565, 11300 Montevideo, Uruguay

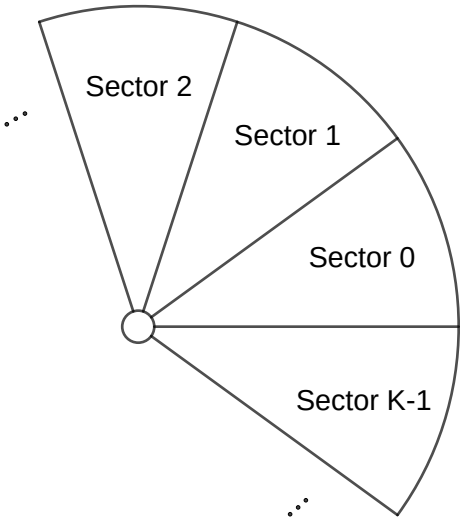


Fig. 1: K-sectored antenna

neighbor nodes, and the ones in range may reply including their network address. But with ESD antennas, there may be (and generally there are) more than one sector-to-sector (S2S) link between two nodes. Besides, the transmitter must know when its neighbor is pointing to a certain direction.

This work is an extended version of a paper presented in SBESC 2018 [1]. In that paper, Dynamic Asynchronous Neighbor Discovery protocol for Directional Antennas (DANDi), a protocol based on collision detection, was presented and evaluated through simulations. In that previous implementation, the collisions were detected by the network simulator and passed as events to the nodes.

In this work, we present a simple and effective collision detection mechanism to be implemented in the nodes. We also evaluate this mechanism and take new considerations for the DANDi protocol to work correctly in lossy networks. Finally, we present the evaluation, both in simulations and in real nodes, of the DANDi protocol including these improvements and its comparison with a state-of-the-art protocol with similar characteristics and for nodes with directional antennas.

The rest of the work is organized as follows. In Sect. 2, we present the related work. In Sect. 3, we describe the proposed protocol, its parameters, time restrictions, we find an expression for the duration of the protocol and present considerations for lossy networks. Section 4 presents some implementation details and the experimental setup. The collision detection mechanism is presented in Sect. 5 and the evaluation of the DANDi protocol in Sect. 6, including a comparison with SAND. Finally, we present conclusions in Sect. 7.

2 Related work

Many algorithms have been proposed to perform neighbor discovery when using directional antennas. Some neighbor discovery algorithms are based on a probabilistic approach. One of the simplest is to divide the time in timeslots and to send with probability p_t and to receive with probability $1 - p_t$. Vasudevan et al. [2] deduce the probability p_t that maximizes the number of nodes discovered in time. This optimal probability depends on the beamwidth of the directional antennas and the number of nodes in the network. The disadvantage of a fully probabilistic approach is that it will take much more time to discover the last links.

Other neighbor discovery algorithms rely on omnidirectional antennas to assist the neighbor discovery process [3], or assume that only the sink node is equipped with a directional antenna [4]. The most common approach is to use directional transmissions and omnidirectional receptions, eliminating the need of knowing in which direction the neighbor's antenna is pointing. The main drawbacks of this approach are that an extra omnidirectional antenna is needed for each node, neighbors that would only be reached with both directional transmissions and receptions would not be discovered, and that the spatial reuse decreases.

Other algorithms [5,6] rely on previous time synchronization among nodes in the network. This time synchronization is hard to achieve before establishing the network. Xiong et al. [5] propose SBA, an algorithm where each node transmits with probability p_t to the sector S_i and receives with probability $1 - p_t$

to the opposite of sector S_i. The synchronization is needed for every node in the network to know where to point at every moment.

There are fully directional (directional transmissions and receptions) algorithms that do not require previous time synchronization among the nodes [7,8,9,10,11], as it is the case of DANDi [1]. The first two algorithms [7,8] use scan sequences based on each node's unique ID, guaranteeing that all the sector combinations among all nodes are tested. In both works the shortcoming is that the collisions are not considered, so the discovery is not guaranteed for dense networks, or even for networks with dense areas. In SAND (Sectored-Antenna Neighbor Discovery) protocol [9] (extended in [10]), a serialized mechanism where one node is discovering its neighbors at a time guarantees that all the sector combinations among all the nodes are tested. In all the algorithms mentioned above, the respective parameters are chosen using the number of nodes in the network or the estimated number of neighbors using the network density and the coverage area of the antenna. So some knowledge of the network topology is needed in advance.

Our proposed algorithm DANDi does not need omnidirectional antennas nor time synchronization; it relies on serialization of the discovery process (like SAND) and on collision detection. DANDi guarantees that all the sectors of all the nodes in the network are discovered, taking collisions into account. Besides, no network information is needed in advance.

3 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 reliable 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 the DANDi protocol is that one single discoverer node (DN) at a time discovers all reliable 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 at 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*.

Figure 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 thoroughly explained in this section.

3.1 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 many *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. Figure 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*).

3.1.1 Probing sectors algorithm

Figure 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.

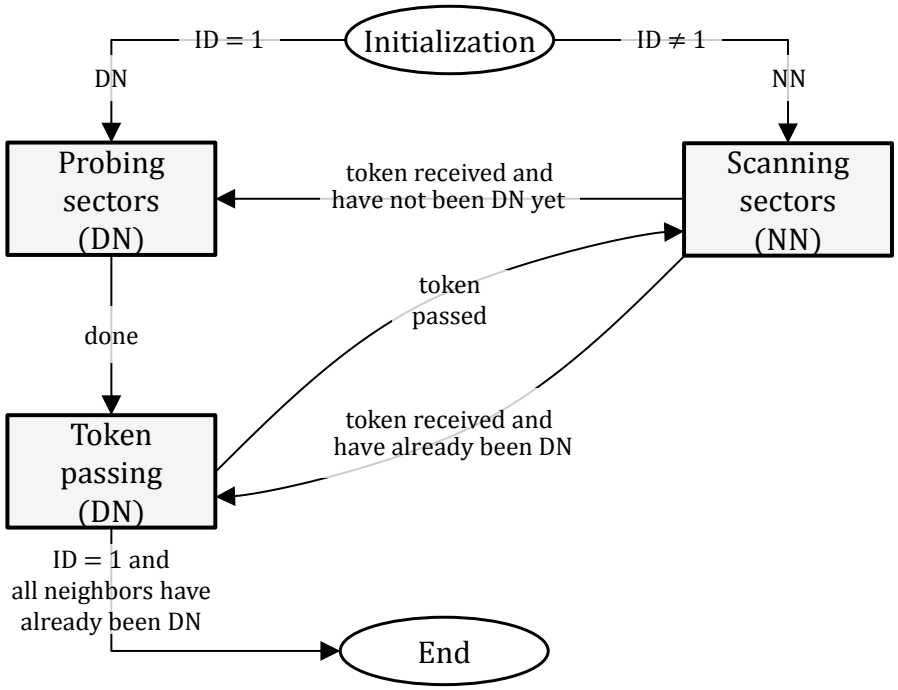


Fig. 2: State diagram: node roles and main process

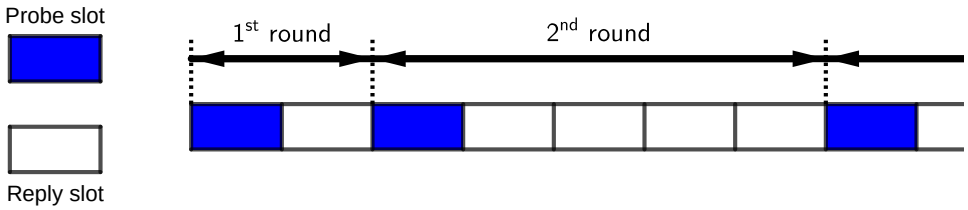


Fig. 3: Slots and rounds

(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 [12]. For the last case, in which the DN detects a collision, all the replies in that slot are lost. Summarizing, more rounds are needed to ensure that all NN are discovered correctly for every case except the first one.

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. The DN could increment the number of slots to speed up the contention resolution if there is a collision in any slot. 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* [13]. 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), the DN can pass the token to the next node only after one round with no replies from the NN. However, if there is a round with more than one reply slot for contention

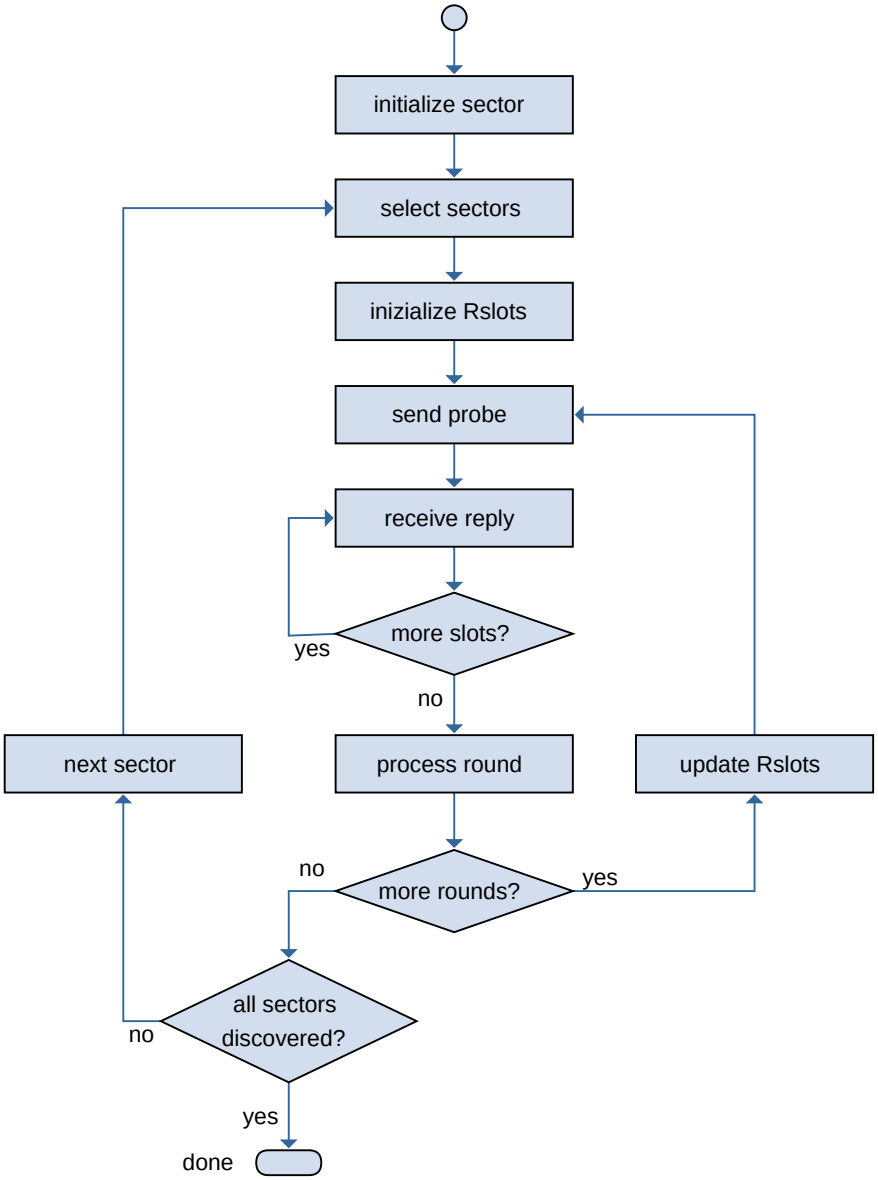


Fig. 4: Probing sectors algorithm

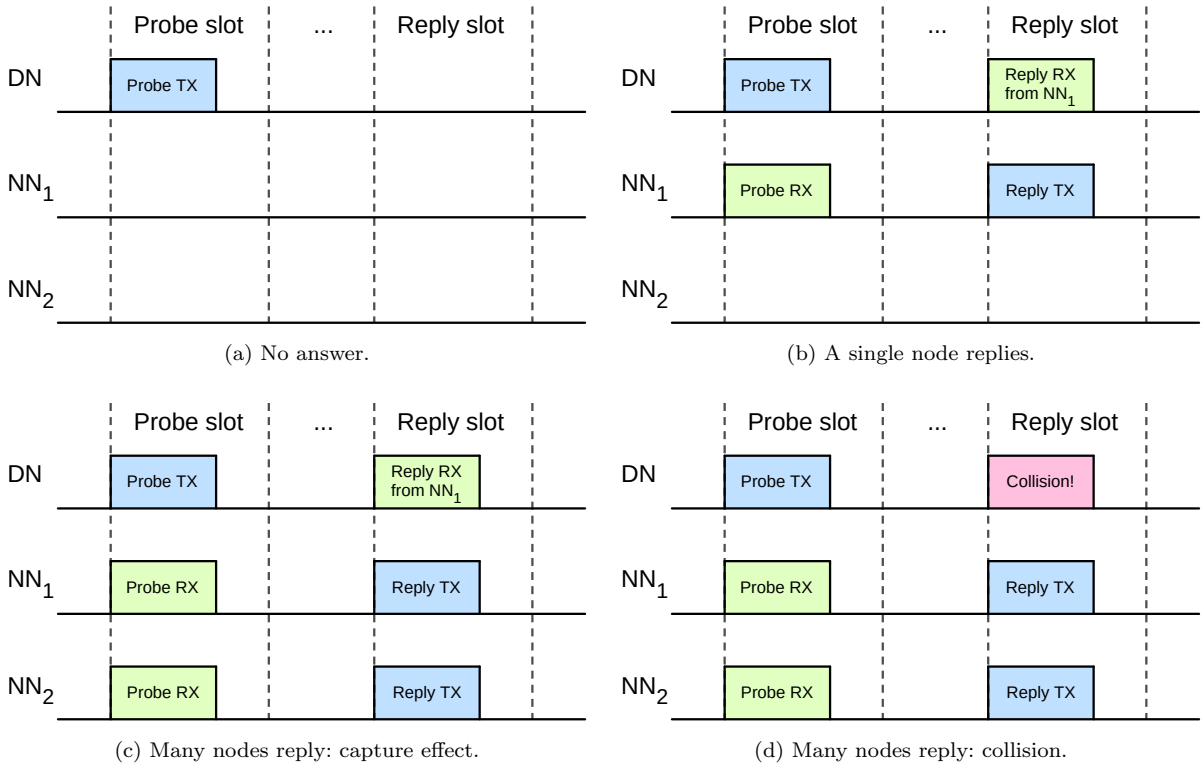
resolution, there will be no probe slots for a certain time (determined below) since the nodes are resolving the contention. To ensure that the missing slots will not lead a neighbor to miss a probe, the missing probe slots must be recovered. To achieve this, the DN must explore a certain number of rounds with a single reply slot (N_{probe}, determined in Sect. 3.3), 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.

3.1.2 Scanning sectors algorithm

Figure 6 shows the scanning sectors algorithm of the NN, where the scan performed in every sector 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 S2S link. This is why probe messages need to include the sector of the DN.



(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

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

3.2 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 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.

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

3.3 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 is 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

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

Figure 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.

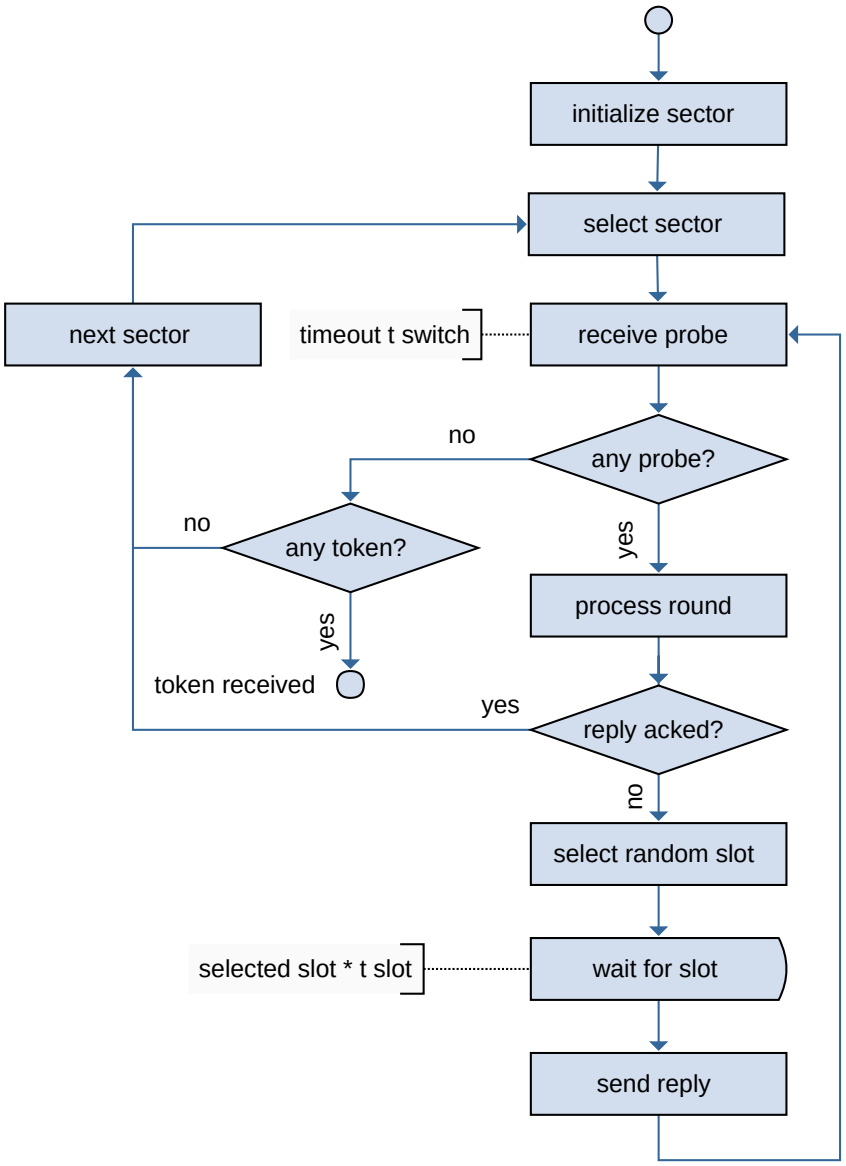


Fig. 6: Scanning sectors algorithm

NN asynchronously switch 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:

$$t_{switch} > t_{probe} + \tau_{frame} + \tau_{switch}. \tag{2}$$

Figure 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:

$$N_{probe} \cdot t_{probe} > K \cdot t_{switch}, \tag{3}$$

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.

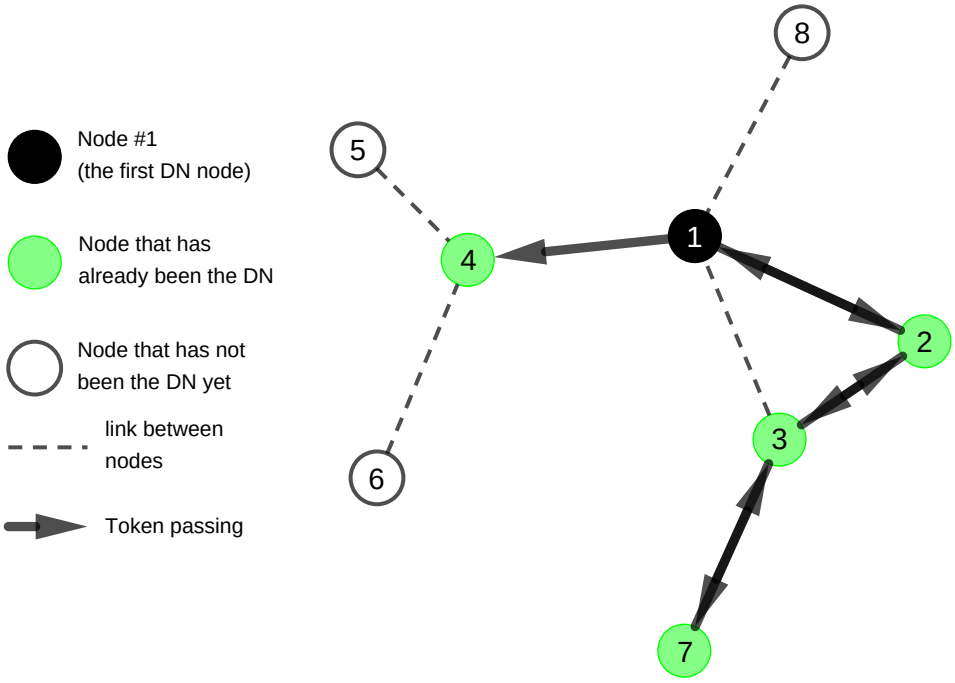


Fig. 7: Partial tree started by node #1. Nodes #2, #3 and #7 have already been the DN, while node #4 is the current DN. Nodes #5, #6 and #8 have not been the DN yet

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

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

3.4 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 tokens 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):

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

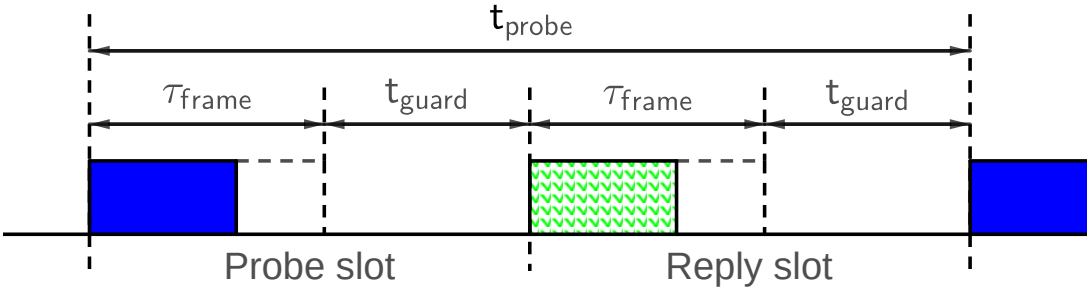
where $T_{PR}(i)$ and T_{TP} are the time taken by the probe-reply mechanism of node i and by the token passing mechanism respectively.

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

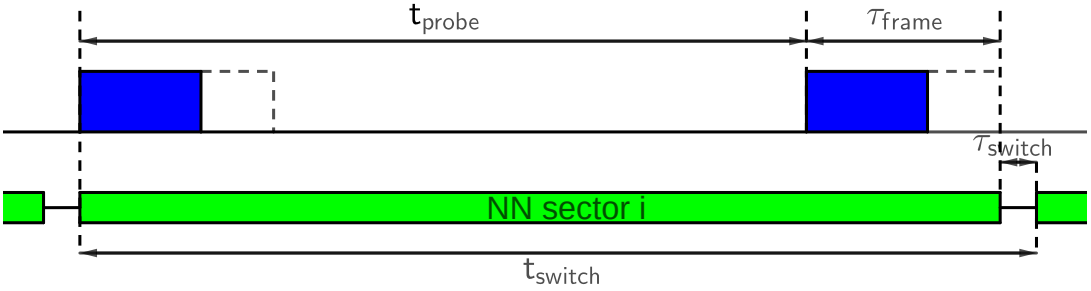
$$T_{PR}(i) = t_{slot} \sum_{j=0}^{K-1} \left\{ \sum_{k=1}^{R_{ij}} \{1 + s_{ijk}\} \right\}, \tag{6}$$

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.

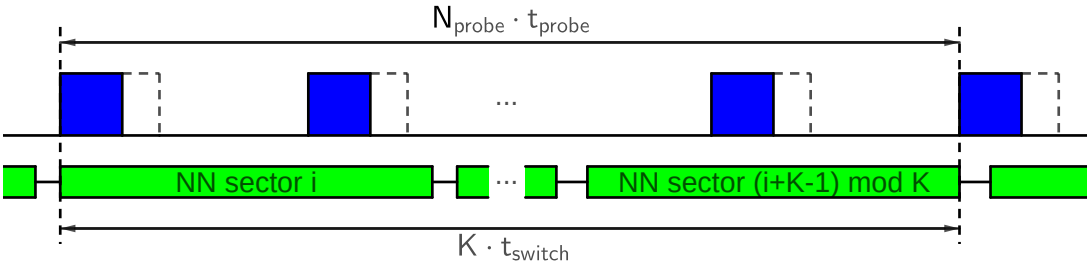
$T_{PR}(i)$ is probabilistic and not deterministic, given that the random choice of reply slot by the NN impacts on the number of suppressed messages or collisions, and then in the number of rounds needed to resolve the contention. It could happen that two NN choose the same reply slot for a lot of consecutive 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



(a) Restriction on t_{probe}.



(b) Restriction on t_{switch}.



(c) Restriction on N_{probe}.

Fig. 8: Time restrictions

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:

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

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

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

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.

3.5 Considerations for lossy networks

The originally proposed DANDi protocol, as described previously in this section, was extensively tested and assessed [1] using the COOJA network simulator [14]. The simulation setup used a radio medium with no packet losses and without interference.

A real-world wireless network faces some practical issues such as frame losses (due to interference, fading, etc.) that must be taken into account in the protocol design and implementation, in order for it to work properly.

The design of the original protocol in a network with real nodes performs poorly. In some cases not all the sectors or nodes are discovered, and in other cases the protocol does not even finish because the token messages are lost. Considering these aspects, the protocol design must include mechanisms to cope with lossy links to guarantee: i) the discovery of all reliable S2S links and ii) the passage of the token.

We analyze the effect of losing the different kind of messages in the original protocol, and we later present modifications to deal with these issues.

Probe. Losing a probe message may lead to fail to discover some S2S links. If a probe is lost, any NN that would have received the probe will continue scanning sectors. Depending on the moment the probe is lost, NNs may receive subsequent probes or fail to discover that particular S2S link if the DN changes its active sector. To reduce the probability of losing a probe message, the number of probe messages sent without answer before changing the active sector (N_{probe}) can be increased. This would also increase the discovery time per sector.

Reply. In case a NN's reply message is lost, the DN will not send the ACK message (sent in the probe message of the following round), and the NN will send again a new reply. Consequently, there is no need to introduce modifications. The effect is that more rounds may be needed to discover every S2S link.

Token. The loss of a token message would lead to stop the discovering process, leaving the protocol in an inconsistent state, so it is fundamental to make the token passage reliable. The token message is then acknowledged. A timer is used to retransmit the token in case the DN does not receive the ACK message.

Ack. If the ACK message corresponding to a token message is lost, the DN will continue to send token messages until the corresponding ACK is received (even if the NN received the token correctly). In order to increase the probability of receiving the acknowledge message, it is sent repeatedly 10 times. This number is selected to be large enough to ensure that at least one ACK arrives to destination, but as low as possible to reduce the overall time of the protocol. If the previous DN fails to receive the ACK, but receives any message that indicates that the target node received the token correctly (a probe message sent by the new DN or a reply message of any node), it assumes that the token message was correctly delivered and takes the NN role.

4 Implementation and experimental setup

We implement DANDi in Contiki OS [15], an operating system for wireless sensor nodes with constrained resources. We based the implementation on the Tmote Sky [16] platform and used the default 6LoWPAN protocol stack (IEEE 802.15.4 in the 2.4 GHz band).

To validate DANDi, we simulate a network of 16 Tmote Sky nodes with 6-sectored antennas in the COOJA network simulator. The directional radiation pattern of the antennas is modeled as a function of the angle between the transmitter and the receiver nodes, and the transmission success is determined by the signal strength of the received packets, also considering the capture effect. The radio medium does not consider packet losses or interference from external sources (e.g. WiFi, Bluetooth).

The experiments with real sensor nodes were performed in an office environment with moderate WiFi interference. We deployed a network of 10 Tmote Sky nodes with 6-sectored SPIDA antennas built in our laboratory as shown in Fig. 9. The maximum gain of these antennas is 6 dBi with a half-power beamwidth of 130° and no side lobes. Output power was set near the minimum value supported by the radio hardware (-25 dBm) in order to enable testing the network in a reduced space. In this configuration, the maximum communication range is around six meters. Ten nodes were distributed over an area of 20 by 8 meters as shown in Fig. 10.

We compare the performance of DANDi with our own implementation of SAND, developed following the protocol description available in the original paper [9]. This implementation was validated through simulations.

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 1.

Table 1: Parameters of the DANDi protocol used in the simulations

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

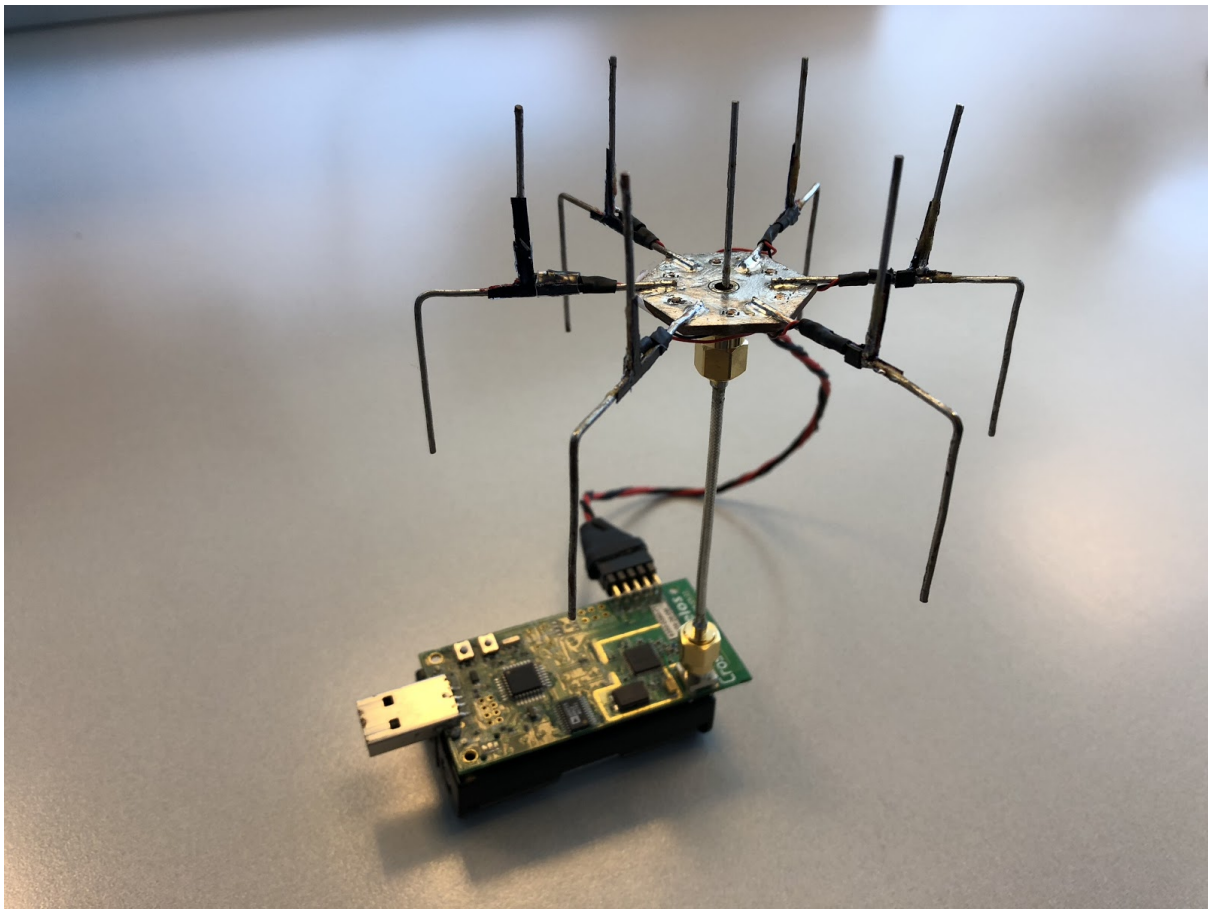


Fig. 9: Tmote Sky node with 6-sectored SPIDA antenna

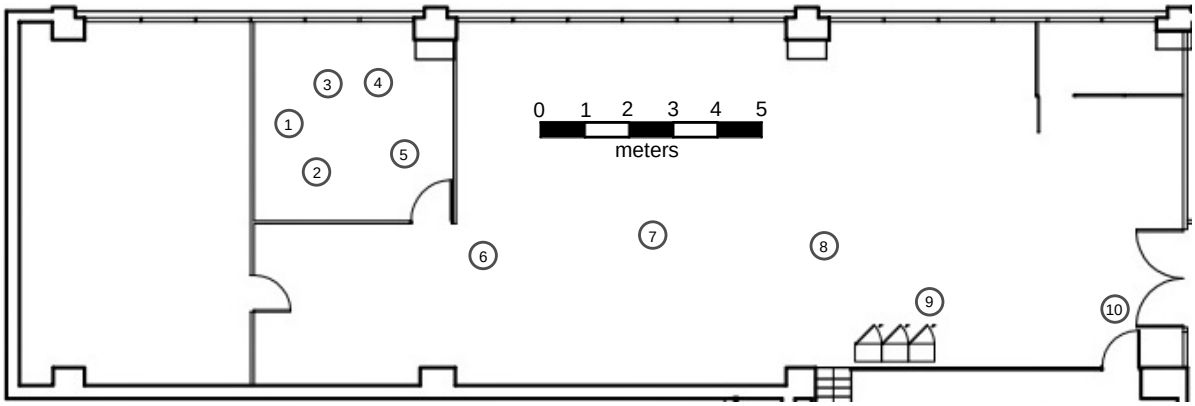


Fig. 10: Network deployed in an office environment

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 is very short (negligible) compared to the reply slots, so the time taken by the probe-reply mechanism of node i can be simplified 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:

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

We still have to choose how the number of slots of each round s_{r_i} is selected depending on the number of slots $s_{r_{i-1}}$, the number of collisions $c_{r_{i-1}}$ and the number of discovered neighbors $N_{r_{i-1}}$ 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 Sect. 3.1:

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

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.

5 Collision detection

One of the key features of DANDi that enables the dynamic contention resolution is the collision detection mechanism. The DN node must be able to detect if more than one node reply and it results in a collision. A receiver-side collision detection can be implemented using Cyclic Redundancy Check (CRC) or Clear Channel Assessment-based (CCA) techniques [17], adopted in some other protocols [18]. We chose to implement the collision detection based on received signal strength measured by the radio receiver as the RSSI (Received Signal Strength Indicator).

To determine if a collision occurs, the DN checks the RSSI level during the reply slots. If the RSSI is above a certain threshold for a certain time, it means one or many NN may have sent a reply message. If the DN does not receive a reply message it assumes a collision occurred.

The RSSI level is polled repeatedly during the reply slot. In our implementation we were able to obtain a valid RSSI measurement every 200 µs. A reply packet has 72 bytes of data and takes approximately 2.46 ms to be transmitted, so we poll the RSSI level around 12 times during the transmission of a reply packet. We set our threshold to seven consecutive checks were the RSSI is above a certain power level to determine if a packet is being transmitted. Seven consecutive checks correspond to 1.4 ms. This time restriction is very important to distinguish between packet collisions and interference. WiFi signals can make the RSSI level increase significantly for short periods of time, even though the nodes are working in the channels with lower overlap with WiFi. In the next section, we show that there are no false positive measurements due to WiFi interference if we consider that a packet transmission should be at least 1.4 ms long.

The power level above which we say there is a packet transmission must be properly selected. On the one hand, a low threshold close to the radio sensitivity (e.g. −95 dBm for the CC2420 radio of the Tmote Sky) could increase the number of false positive (FP) detections (a collision is detected and there was none) due to a high noise level (e.g. Wi-Fi interference). On the other hand, a relatively high threshold could increase the number of false negative (FN) detections (a collision occurred and it is not detected), resulting in missing detecting a real collision just because the received signal is low.

A threshold of −88 dB was selected based on empirical experiments, detailed in the next section. We used the criteria of selecting the lower threshold that resulted in 0 % of FP, even though we know that the protocol could fail to discover links with an RSSI lower than −88 dB due to FNs in the collision detection mechanism. The S2S links that we would fail to discover would be the weaker ones, and probably would not be used for further communication to avoid packet losses even if they were discovered.

5.1 Collision detection evaluation

We performed several experiments to assess the performance of the collision detection mechanism, both in simulation and with real sensor nodes.

5.1.1 Simulation results

We performed two simulated experiments as a first step in verifying our approach. In experiment #1, we placed one DN node periodically sending probe messages and one NN node that replied each probe message. This experiment corresponds to a scenario without collisions and it aims to obtain an estimation of the FP rate. In experiment #2, we placed a second NN replying the probe messages in the same slot as the other NN to force collisions. This experiment aims to obtain an estimation of the FN rate.

Table 2 shows the results of the simulation of these experiments.

Results from experiment #1 show that we have no FPs, as no collisions were detected. In experiment #2, even though there is only one slot for the NNs to reply, the DN is able to receive some packets due to the capture effect. The remainder cases are collisions correctly detected so we can conclude that we have no FNs.

Experiment	Probe messages sent	Reply messages received	Collisions detected
#1	10 000	10 000	0
#2	10 000	924	9076

Table 2: Collision detection simulation results

These results show that the logic of the implementation works fine, but to assess the real performance we have to test it with real sensor nodes.

5.1.2 Experimental results

We repeat the experiments performed in simulation with real sensor nodes in an office environment (as described in Sect. 4) adding a third experiment. For experiment #3 we placed a single DN node sending probe messages, without any NN. The main objective of this experiment was to find out if WiFi interference was detected as packet collisions. Results from these three experiments are shown in Table 3.

Experiment	Probe messages sent	Reply messages received	Collisions detected
#1	10 000	9561	125
#2	10 000	3016	6822
#3	10 000	0	0

Table 3: Collision detection experimental results

For experiment #1 (scenario without collisions), we can see that we have detected 125 collisions when we were expecting none. We can also observe that 9561 reply messages were received by the DN. The remainder cases are probe messages that did not received the corresponding reply message nor were detected as collisions.

In this kind of networks, depending on the link quality, some packets are expected to be lost. We analyzed the sequence number of each packet and found that 229 reply messages were lost. When a reply packet is lost, it is possible that it was sent by the NN but could not be received correctly by the DN due to interference. If this is the case, the DN would find that the RSSI exceeds the threshold long enough to mistake it with the collision of two or more reply messages. Then, the 125 detected collisions correspond to some of the 229 reply messages sent by the NN but not received correctly by the DN due to interference.

For experiment #2 (scenario forcing collisions), we can observe that there are 162 probe messages that did not receive a reply nor were detected as a collision ($10\,000 - 3016 - 6822 = 162$). Analyzing the sequence number of the packets, we can observe that from these probe messages, 125 were not received by either of the NN. So the remaining 37 probe messages that were replied by the NN, but not received nor detected as a collision by the DN are the candidates for FNs. We say candidates because it is also possible that the NN sent the replies but the DN did not receive any of them. But if we take a worst-case scenario, the FN ratio would be:

$$FN \leq \frac{37}{6822 + 37} = 0.54\,\%.$$

For experiment #3 we obtained no detections of collisions for 10 000 probe messages transmitted. Based on the results of the experiments, we can conclude that our implementation is robust to interference from the environment.

6 DANDi evaluation

This section presents the performance results of the DANDi protocol, and a comparison with the state-of-the-art protocol SAND. We present both simulation and experimental results with real sensor nodes.

6.1 Simulation results

To assess the protocol effectiveness, we simulate DANDi for different network topologies. In this section we show the obtained results.

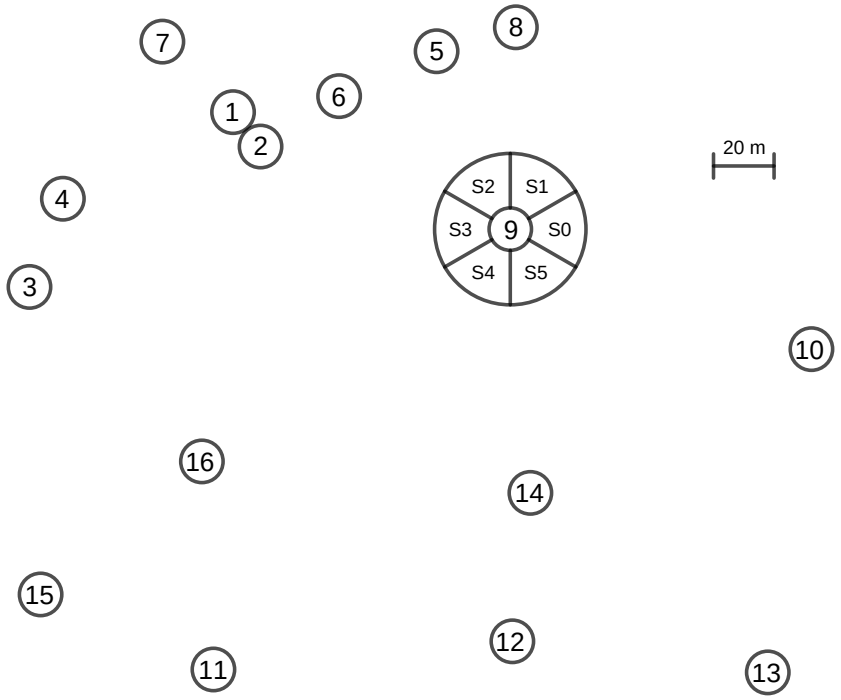


Fig. 11: Simulated network with 16 nodes. The sectors of all nodes are oriented as shown for node #9

6.1.1 Performance evaluation

In the first place, we simulate the 16-node network shown in Fig. 11 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 4: Number of S2S links per sector of each node of the network

Node	S_0	S_1	S_2	S_3	S_4	S_5	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 4 shows the number of S2S links discovered for each sector of each node in the network. Note that a node can discover more than one S2S link per neighbor. At most, a node can discover K^2 (in our case 36) S2S links per neighbor. 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 very dense zone 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 (e.g. the case of node #10 for sectors S_0, S_1, and S_5).

Table 5: Average time taken by the probe-reply mechanism of each sector of each node in the network

Node	S_0 (s)	S_1 (s)	S_2 (s)	S_3 (s)	S_4 (s)	S_5 (s)	Total (s)
#1	1.839	1.550	1.083	1.783	1.519	1.261	9.034
#2	2.543	1.340	1.415	1.438	1.421	1.201	9.358
#3	1.215	1.020	0.550	0.406	0.406	1.103	4.700
#4	1.596	1.096	0.611	0.406	0.406	0.863	4.979
#5	0.610	0.406	1.100	1.106	1.359	1.423	6.004
#6	0.888	0.785	1.299	1.759	1.414	1.968	8.111
#7	1.623	0.955	0.406	0.661	1.139	1.835	6.619
#8	0.406	0.406	0.486	1.278	1.311	0.713	4.600
#9	0.406	0.489	1.286	1.029	0.406	0.488	4.104
#10	0.406	0.406	0.406	0.504	0.481	0.406	2.610
#11	0.406	0.476	0.476	0.406	0.406	0.406	2.578
#12	0.421	0.458	0.406	0.492	0.406	0.406	2.590
#13	0.406	0.406	0.478	0.561	0.406	0.406	2.664
#14	0.609	0.428	0.406	0.439	0.406	0.490	2.778
#15	0.551	0.833	0.464	0.406	0.406	0.406	3.066
#16	0.406	0.406	0.525	0.538	0.626	0.450	2.951

Table 5 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 Tables 4 and 5 (last column in both tables) are graphically presented in Figs. 12 and 13 respectively. Figure 13 also includes the minimum and the maximum time taken of all simulations.

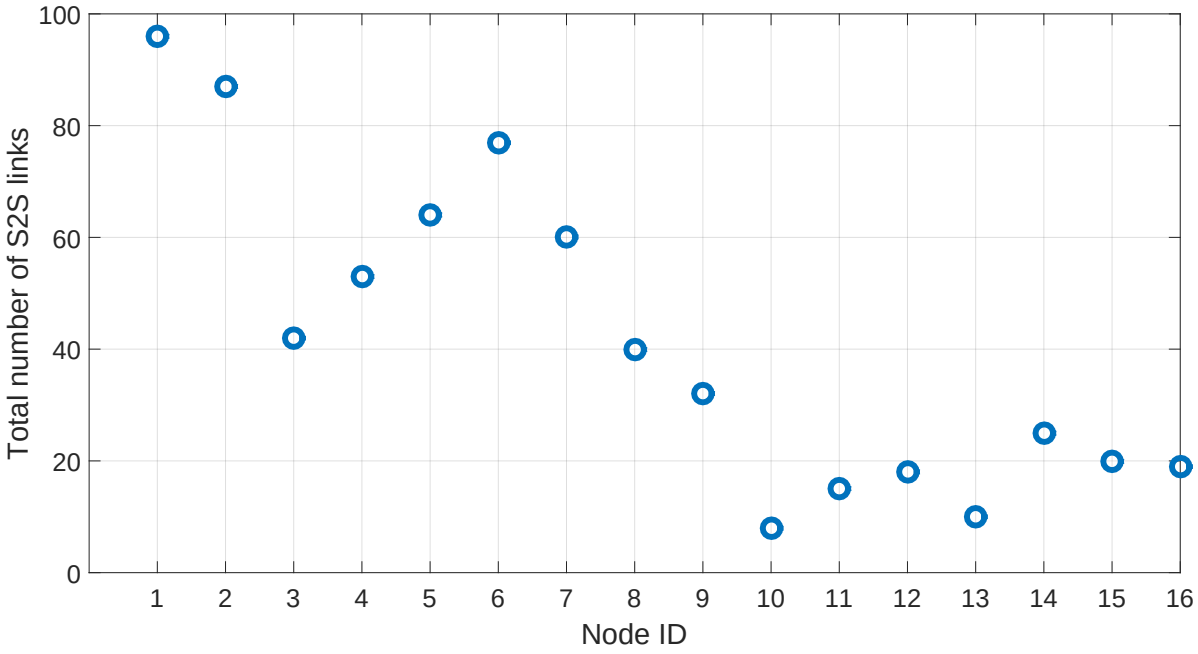


Fig. 12: Total number of S2S links for each node of the network

We see that Figs. 12 and 13 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. 13 the minimum time taken per node (in case there are no collisions between reply messages), calculated using Eq. (10): $T_{PR}(i) = t_{slot}KN_{probe} = (31.25\,\text{ms}) \times 6 \times 13 = 2.4375\,\text{s}$. We observe that nodes #10 to #16 take very close to the minimum time.

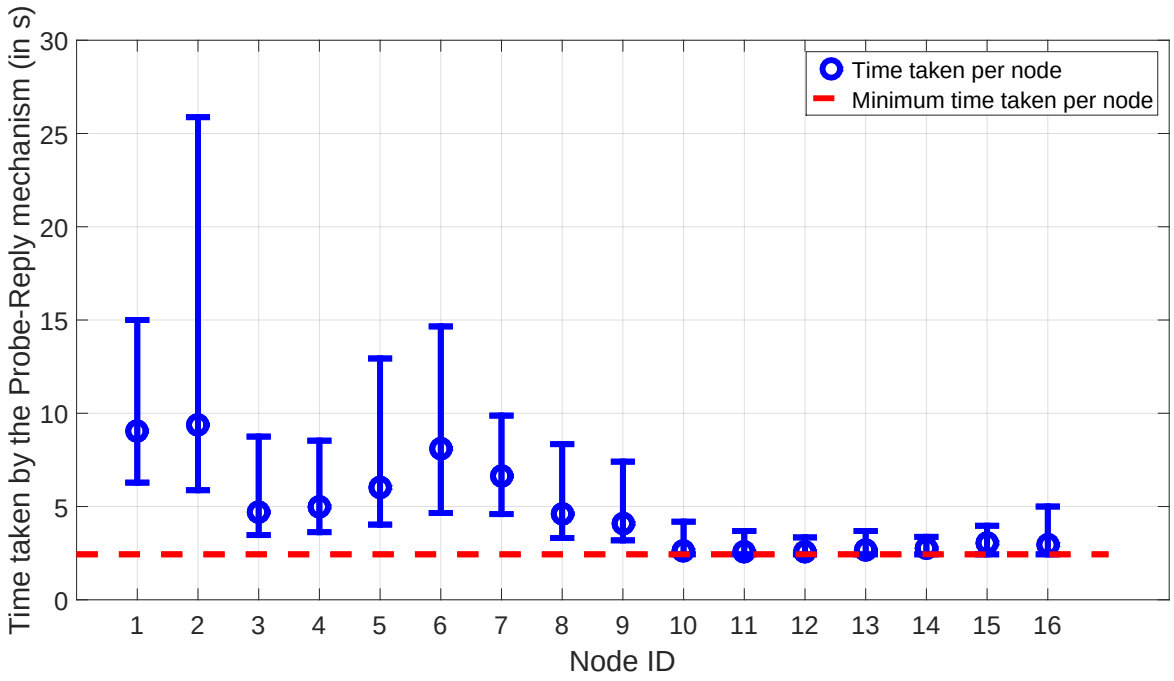


Fig. 13: Minimum, average and maximum time taken by the probe-reply mechanism for each node of the network

6.1.2 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 [19]. Like DANDi, SAND is a token-based protocol in which a single node is discovering neighbors at a time, defining slots and rounds. The number of slots s and number of rounds r in SAND have the same meaning as in DANDi. But with SAND, s and r are fixed and set in advance before running the protocol. The parameters of both DANDi and SAND protocols used in the simulations are shown in Table 6. In order for the comparison to be fair, we used the same protocol parameters and the same network topologies. We see from Table 6 that the DANDi protocol is simpler than SAND in that it has less parameters.

Table 6: 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, it allows to obtain s and r minimizing the protocol duration [9, 10]. 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. 11 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 regard 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 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 the 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. 14, 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.

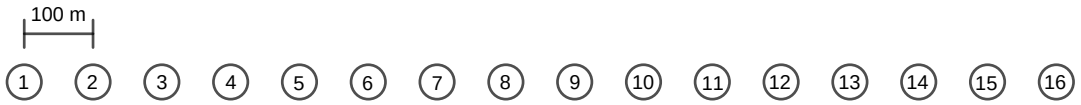


Fig. 14: Simulated network with 16 nodes and no collisions

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 Sect. 3.4. 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 6 (assuming that $t_{token\text{-}ack} \approx 0$ or negligible), we obtain $T_{DANDi} = 50.25$ s, confirming the theoretical equations deduced.

These results show that the DANDi protocol implementation works as expected. We proceeded then to test it with real sensor nodes.

6.2 Experimental results

6.2.1 Performance evaluation

We deployed the network shown in Fig. 10 composed of ten Tmote Sky nodes equipped with 6-sectored antennas. We performed the experiments running the DANDi protocol and repeated them six times. The total times required by DANDi to perform the neighbor discovery in the different experiments were 134 s, 126 s, 124 s, 113 s, 121 s and 109 s, giving an average time of 121 s.

As expected from the position of the nodes, we can see that there are two unevenly dense zones in the network: a very dense zone near nodes #1 to #6 and a lesser dense zone near nodes #7 to #10.

We observe in Table 7 that nodes in the first zone discover more S2S links in average (between 124 and 171) and, correspondingly take more time (between 9.805 s and 21.820 s). On the other hand, nodes in the second zone discover lesser S2S links (between 46 and 79), and correspondingly take lesser time (between 3.547 s and 10.852 s). This shows how the protocol dynamically adapts to the network topology, taking more time in denser zones with more S2S links to discover, and speeding up in zones with less nodes and less links to discover.

Node	S2S links	Time (s)
#1	125	10.117
#2	137	10.984
#3	132	9.805
#4	158	12.789
#5	171	14.898
#6	124	21.820
#7	75	7.602
#8	79	10.852
#9	75	5.273
#10	46	3.547
Total	1122	107.687

Table 7: Average S2S links discovered and average time taken by the probe-reply mechanism for each node using the DANDi protocol

6.2.2 Comparison with SAND

For the same network topology, we run SAND protocol with different number of slots and rounds, to compare it with DANDi. We run SAND $(s = 2, r = 2)$, $(s = 3, r = 3)$ and $(s = 5, r = 5)$, where s is the number of slots and r is the number of rounds.

Table 8 shows the total number of S2S links discovered in the network (per node and total) for the different experiments.

Node	DANDi	SAND		
		$(s = 2, r = 2)$	$(s = 3, r = 3)$	$(s = 5, r = 5)$
#1	125	96	114	129
#2	137	119	150	160
#3	132	105	126	130
#4	158	135	156	163
#5	171	138	152	178
#6	124	74	115	146
#7	75	78	106	110
#8	79	63	79	81
#9	75	69	71	79
#10	46	38	45	52
Total	1122	915	1114	1228

Table 8: Total number of S2S links discovered in the network per node for the different experiments

Table 9 shows the time taken by the probe-reply mechanism of each node.

To compare the total time taken by each protocol, besides the time taken by the probe-reply mechanism, we have to consider the time required for the token passing present in both protocols. For SAND we also have to consider the extra time taken by the mechanism needed to locally synchronize the nodes before beginning the probe-reply mechanism. These total times are shown in Table 10.

Node	DANDi (s)	SAND		
		$(s = 2, r = 2)$ (s)	$(s = 3, r = 3)$ (s)	$(s = 5, r = 5)$ (s)
1	10.117	4.500	10.125	28.125
2	10.984	4.500	10.125	28.125
3	9.805	4.500	10.125	28.125
4	12.789	4.500	10.125	28.125
5	14.898	4.500	10.125	28.125
6	21.820	4.500	10.125	28.125
7	7.602	4.500	10.125	28.125
8	10.852	4.500	10.125	28.125
9	5.273	4.500	10.125	28.125
10	3.547	4.500	10.125	28.125
Total	107.687	45.000	101.250	281.250

Table 9: Probe-reply time taken by each node of the network for the different experiments

We observe that for the DANDi protocol, the time taken by each node to perform the discovery is different and depends on the number of neighbors each node has, showing its dynamic property. For SAND, the time taken by each node to perform the discovery is fixed and depends on the number of rounds and

DANDi (s)	SAND		
	$(s = 2, r = 2)$ (s)	$(s = 3, r = 3)$ (s)	$(s = 5, r = 5)$ (s)
121	81	137	318

Table 10: Total time taken for the different experiments

slots. To ensure that a node in a dense zone discover all the S2S links, we have to force nodes in lesser dense zones to take more time to discover their neighbors.

We also see that while SAND$[s = 2, r = 2]$ takes less time than DANDi to finish, it fails to discover some S2S links in the denser zones.

SAND$[s = 3, r = 3]$ takes more time than DANDi to complete, while it discovers almost the same number of S2S links than DANDi.

If we compare DANDi with SAND$[s = 5, r = 5]$, we can see that it is 2.6 times faster but it discovers 10 % less S2S links. However, the S2S links that are not discovered by DANDi are the ones with RSSI less than the collision detection threshold (−88 dB). Those links are not considered as reliable links.

7 Conclusions

DANDi, a fully directional asynchronous and dynamic neighbor discovery protocol for wireless sensor networks is proposed, implemented and tested through extensive simulations and experiments with real nodes. The contention resolution mechanism in which DANDi is based relies on a collision detection mechanism that is also implemented and tested in simulations and experiments, achieving a FP rate of 0 % and a FN rate of 0.54 %.

The results of the simulations show that all the reliable 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.

The experimental results confirm that DANDi can effectively discover every reliable S2S link in a given network with a better performance than SAND. While DANDi may fail to discover the weaker S2S links below the collision detection threshold, SAND may fail to discover some random S2S links if the number of slots and rounds is not high enough (SAND$[s = 2, r = 2]$ and SAND$[s = 3, r = 3]$). This could lead SAND to miss reliable S2S links. If the selected number of slots and rounds is high enough (SAND$[s = 5, r = 5]$), the time taken by the protocol is 2.6 times higher than the time taken by DANDi.

To the best of our knowledge, this makes DANDi 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 every reliable communication link in a network without requiring any prior information of the network topology.

Future work could include combining the RSSI metric with additional information such as CRC checks, CCA or preamble-based methods in order to improve the collision detection mechanism, reducing false negatives.

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

References

1. N. Gammarano, J. Schandy, and L. Steinfeld, "DANDi: Dynamic Asynchronous Neighbor Discovery Protocol for Directional Antennas," in *2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC)*, Nov 2018, pp. 16–23.
2. 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.
3. J. Wang, W. Peng, and S. Liu, "Neighbor discovery algorithm in wireless local area networks using multi-beam directional antennas," *Journal of Physics: Conference Series*, vol. 910, p. 012067, oct 2017.
4. G. D. S. Sidibe, A. Surier, R. Bidaud, G. Delisle, N. Hakem, M. Servajean, B. Rmili, G. Chalhoub, and M. Misson, "Use of a switched beam antenna in a star wireless sensor network for data collection: Neighbor discovery problem," in *2019 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Oct 2019, pp. 21–26.
5. W. Xiong, B. Liu, and L. Gui, "Neighbor discovery with directional antennas in mobile ad-hoc networks," in *2011 IEEE Global Telecommunications Conference - GLOBECOM 2011*, Dec 2011, pp. 1–5.
6. B. El Khamlichi, D. H. N. Nguyen, J. El Abbadi, N. W. Rowe, and S. Kumar, "Collision-aware neighbor discovery with directional antennas," in *2018 International Conference on Computing, Networking and Communications (ICNC)*, March 2018, pp. 220–225.

7. L. Chen, Y. Li, and A. V. Vasilakos, "On oblivious neighbor discovery in distributed wireless networks with directional antennas: Theoretical foundation and algorithm design," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 1982–1993, Aug 2017.
8. Y. Wang, S. Mao, and T. S. Rappaport, "On directional neighbor discovery in mmwave networks," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 1704–1713.
9. 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.
10. 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.
11. N. Gammarano, J. Schandy, and L. Steinfeld, "Q-SAND: A Quick Neighbor Discovery Protocol for Wireless Networks with Sectored Antennas," in *2018 Ninth Argentine Symposium and Conference on Embedded Systems (CASE)*, Aug 2018, pp. 19–24.
12. 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.
13. "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.
14. F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level Sensor Network Simulation with COOJA," in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, Nov 2006, pp. 641–648.
15. 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.
16. *Tmote Sky Datasheet*, Moteiv Corporation, June 2006, rev. 1.0.2.
17. 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.
18. 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.
19. 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.